Differential Privacy in Reinforcement Learning

Waël Doulazmi [*]
Master MVA
ENS Paris-Saclay
wael.doulazmi@ens-paris-saclay.fr

Ambroise Odonnat [*]
Master MVA
ENS Paris-Saclay
ambroise.odonnat@ens-paris-saclay.fr

Abstract

Reinforcement learning (RL) enables autonomous agents to learn in an interactive environment using feedbacks from their previous actions. RL algorithms are used in various fields where a personalized service is needed such as autonomous driving and recommendation systems. In these domains, it is critical to preserve sensitive information contained in user data. To this end, we design novel private RL algorithms and evaluate the impact of privacy on the learning process [1]. We compare Central and Local Differential Privacy on policy-based REINFORCE and value-based Deep Q Network algorithms. We observe in our experiments that privacy guarantees come with an increase in the complexity of the learning process and a decrease in terms of sample efficiency of the agents.

1 Introduction

Reinforcement learning (RL) is about creating autonomous agents that directly interact with the environment with little to no human supervision. In real word environments where state and action spaces are large or continuous, tabular RL algorithms become computationally intractable. To overcome this limitation, function approximation algorithms, such as Deep Neural Networks, were proposed to map states and actions to a low-dimensional space. While this holds the promise of unlocking very innovative applications, there are several complications to be addressed to make this approach applicable to real-world problems. In several domains of application such as healthcare, cybersecurity, autonomous driving or recommendation systems, data used to train RL agents are highly sensitive. Questions have been raised about the reliability and the risks associated with building autonomous agents. There has recently been a surge of interest in the reinforcement learning community around privacy. Pan et al. has shown in [14] that, unless sufficient precautions, RL agents cause information leakage about the environment. That is to say that one can infer information about the reward and states only by observing the learned policies. This is the reason why users might want to keep their data private to an observer (scenario ①) but also to an RL agent (scenario ②).

Differential Privacy (DP), first introduced in 2006 by Dwork et al. [7], is a standard mechanism to preserve data privacy. From a mathematical perspective, the (ε, δ)-definition guarantees that it is statistically hard to infer information on the environment just by observing predictions. The DP framework is thus adapted to the scenario ①. A drawback of the DP framework is that the autonomous agents directly observe the states and the rewards of the user, possibly containing sensitive information. It motivated the development of a stronger privacy-preserving mechanism called Local Differential Privacy (LDP). In this setting introduced in 2013 by Duchi et al. [6], user data is protected before the autonomous agent can explore the environment. It corresponds to scenario ②. The stronger assumptions of LDP comes with new challenges in terms of learning process. Garcelon et al. [10] established a lower bound for regret minimization in finite-horizon MDPs with LDP guarantees, showing that guaranteeing privacy has a multiplicative effect on the regret.

[*]Equal contribution
[1]Code available at `https://github.com/AmbroiseOdonnat/RL-Differential-Privacy`

Preprint. Under review.

Recently a lot of effort has been made to develop and analyze private algorithms for multi-armed bandits [5, 9], which are commonly encountered in recommendation systems. However, the application of differential privacy to deep reinforcement learning algorithms with continuous state spaces has received less attention, despite its importance for use cases such as autonomous driving. In this paper, we provide differentially private deep RL algorithms and study the impact of privacy on the learning process.

Contributions In summary, our contributions are two folds. (i) We propose central differentially private versions of REINFORCE and DQN algorithms using DP-SGD. (ii) We apply local differentially privacy to theses algorithms. With the increase in privacy, we observe that the learning process is slower and that the convergence is harder to achieve.

2 Background

2.1 Reinforcement Learning

In the RL framework introduced by Sutton and Barto in [18], the agent is a learner and a decision-maker which interacts with the environment. At each time step t, the agent observes the state of the environment $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$. As a consequence, the agent receives a reward r_t and is in a new state s_{t+1}. In this paper, we focus on non-deterministic environments with discrete and finite state and action spaces. Mathematically speaking, we consider $M = (\mathcal{S}, \mathcal{A}, R, P, S_0, \gamma)$ a Markov Decision Process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$ and discount factor γ. $P(a, s, s')$ is the transition probability of reaching state s' by taking action a in state s, $R(s)$ is the reward received by the agent in state s and S_0 is the initial state distribution over $\mathcal{S}$. Following the definition in [8], a measurable mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$ is called a policy. An agent following a policy π in an MDP takes action $a_t = \pi(s_t)$ at step t. V^π is the value function of policy π, with $V^\pi(s)$ the value of expected return starting from state s. Q^π is the action-value function or Q-function of policy π, with $Q^\pi(s, a)$ the value of taking action a in state s. The objective of an RL agent is to maximize the expected cumulative rewards.

2.2 Differential Privacy

Differential Privacy [7] is a mechanism that enables privacy by minimizing the risk of information recovery. In DP, one can define the bounds to how much information can be inferred by observing the predictions of the model. The level of privacy ensured by an algorithm M over a database D is characterized by ε the privacy budget and δ the probability of error [2]. The privacy budget ε gives an indication of the privacy loss of a DP algorithm: the higher the value of ε, the higher the privacy loss. The probability of error $\delta > 0$ takes into account bad events that could result in high privacy loss. Formally, it is the probability that the output of the model reveals the identity of a particular individual. δ has to be maintained low to minimize the risk of privacy loss.

Central Differential Privacy. Central Differential Privacy (CDP) refers to the classical DP framework introduced in [7]. In the CDP framework, the model has access to the real data and then apply a differentially private mechanism on them. Such mechanism mainly requires to add calibrated random noise on the data. In CDP, a well-known mechanism is the DP-SGD (subsection 2.3) that consists in adding Gaussian noise on the gradients during training of the model. To be able to give a formal definition of CDP, we need to introduce the notion of adjacent inputs. Typically, we say that two inputs d and d' are adjacents if they differ from only one element. In Computer Vision, d and d' could be training sets of image-label pairs which differ from a single entry. However, in our experiments, optimal policies are learned by the RL agents which explore the environment and take the obtained rewards as feedback for their actions. As we care about the privacy of the policies and rewards in CDP, we say that two reward functions d, d' are adjacent if the L2-norm of their differences is bounded by 1 [16]. CDP can be defined as follows:

Definition 1. For any $\varepsilon \geq 0$ and $\delta \geq 0$, a privacy preserving mechanism $\mathcal{M} : D \mapsto R$ with domain D and range R satisfies (ε, δ)-Differential Privacy if for any two adjacent inputs d, $d' \in D$ and for any subset of outputs $S \subseteq R$, it holds that:

$$\mathbb{P}[\mathcal{M}(d) \in S] \leq \exp(\varepsilon)\mathbb{P}[\mathcal{M}(d') \in S] + \delta \tag{1}$$

Local Differential Privacy. Local Differential Privacy (LDP) is a different yet stronger approach of privacy. In the LDP framework [2], the model does not see the raw data. Instead, information is secured locally by the user using a private randomizer M, before being sent to the RL agent [10]. We introduce several notions based on [20]) before formally defining LDP. We denote $\mathcal{U}$ the set of users. We characterize a user $u \in \mathcal{U}$ by a starting state distribution $\rho_{0,u}$, that is to say $s_1 \sim \rho_{0,u}$ and a tree of depth H describing the possible sequences of states, actions, rewards. For a user u with initial distribution $s_{1,u} \sim \rho_{0,u}$, we denote $X_u = \{s_{h,u}, a_{h,u}, r_{h,u} | h \in [H]\} \in \mathcal{X}_u$ the trajectory corresponding to user u following a policy π. Private data given by u to the RL agent correspond to $\mathcal{M}(X_u)$, where $\mathcal{M}$ is a randomizer which privatizes sensitive information while encoding sufficient information for learning. LDP can be defined as follows:

Definition 2. For any $\varepsilon \geq 0$ and $\delta \geq 0$, a privacy preserving mechanism $\mathcal{M}$ satisfies (ε, δ)-Local Differential Privacy if and only if for all users $u, u' \in \mathcal{U}$, trajectories $(X_u, X_{u'}) \in \mathcal{X}_u \times \mathcal{X}_{u'}$ and all $\mathcal{O} \in \{\mathcal{M}(\mathcal{X}_u) | u \in \mathcal{U}\}$:

$$\mathbb{P}[\mathcal{M}(X_u) \in \mathcal{O}] \leq \exp(\varepsilon)\mathbb{P}[\mathcal{M}(X_{u'}) \in \mathcal{O}] + \delta \tag{2}$$

2.3 DP-SGD

Abadi et al. introduced in [1] the differentially private stochastic gradient descent (DP-SGD), a differentially private implementation of the stochastic gradient descent [17]. DP-SGD, outlined in the Algorithm 1, consists in adding noise on the computed batched gradient during the descent of the algorithm. One of the main issue with DP-SGD is the computational efficiency. Indeed, each element of the gradient should be normalized and disturbed with noise, which counterbalance the parallel computation of SGD provided by most deep learning libraries. Thankfully, efforts were made by the authors and those libraries, notably PyTorch [15], to enable fast computation and backpropagation. One of the main challenges of our work will be to adapt this implementation to the REINFORCE and the DQN algorithms.

Privacy Guarantees. The authors provide a strong accounting method called the moments accountant that depends on N the size of the input data, L the batch size, $q = \frac{L}{N}$ the sampling ratio, E the number of epochs and $T = \frac{E}{q}$ the number of steps. They show that the DP-SGD ensures an $(q\varepsilon\sqrt{T}, \delta)-$differential privacy if one chooses $\sigma = \sqrt{2 \log \frac{1.25}{\delta}}$.

Algorithm 1: Differentially private SGD (Outline)

```
Input: Examples $\{x_1, ..., x_N\}$, loss $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$.
Parameters: learning rate $\eta_t$, noise scale $\sigma$, group size L, gradient norm bound C.
Initialize $\theta_0$ randomly
Target: $\bar{\theta} \leftarrow \theta_0$
$D \leftarrow [\,]$
Get initial state $s_0$
for $t \in [T]$ do
    Take a random sample $B_t$ with sampling probability $\frac{L}{N}$
    Compute gradient
    For each $i \in B_t$, compute $\mathbf{g}_t(x) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x)$
    Clip gradient
    $\bar{\mathbf{g}}_t(x) \leftarrow \mathbf{g}_t(x)/\max(1, \frac{||\mathbf{g}_t(x)||_2}{C})$
    Add noise
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\sum_{x\in\mathcal{B}}\left(\bar{\mathbf{g}}_t(x) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{g}_t$
end
Output $\theta_T$ and compute the overall privacy cost $(\varepsilon, \delta)$ using a privacy accounting method
```

3 DP-SGD in Reinforcement Learning

In this section, we adapt the DP-SGD mechanism to deep reinforcement learning algorithms, namely REINFORCE [19] and Deep Q-Network (DQN) [12]. We chose those two algorithms to have one example from each class of RL approaches: policy-based and value-based. We rely on the Cartpole environment [3] to analyze the impact of differential privacy on the sample efficiency of RL algorithms.

3.1 Algorithms

Policy-based. The REINFORCE algorithm, introduced in [19], approximates stochastic policies in a parameterized space:

$$\pi \in \mathcal{F}_\pi = \{\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0,1],\ \theta \in \mathbb{R}^d\}$$

We're looking for a policy that optimizes the policy-performance criterion (finite-horizon setting):

$$\max_{\theta \in \mathbb{R}^d} J(\theta), \qquad J(\theta) = \mathbb{E}\Big[\sum_{t=0}^{H-1} \gamma^t r_t \,|\, a_t \sim \pi_\theta(s_t), s_0 \sim \mu_0\Big]$$

The gradient of J is shown to satisfy [19]:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau\Big[\Big(\sum_{t=0}^{H-1} \nabla_\theta \log(\pi_\theta(s_t, a_t))\Big)\Big(\sum_{t=0}^{H-1} \gamma^t r_t\Big)\Big]$$

Where the expectation is taken w.r.t $\{\tau = (s_0, a_0, r_0, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)\}$, the possible trajectories in the environment. REINFORCE consists in using Monte-Carlo estimation to compute an estimate of this gradient, and perform gradient descent on θ. In the specific case of Cartpole, where states are independent from one another, using only one trajectory for gradient estimation is sufficient. The DP-SGD REINFORCE is described in the Algorithm 2.

Algorithm 2: DP-SGD REINFORCE

```
Initialize $\theta_0$ randomly
while $k = 0, 1, \ldots$ do
    Draw $s_0 \sim \mu_0$
    Generate trajectory
    Run policy $\pi_{\theta_k}$, and collect $\tau = (s_0, a_0, r_0, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$
    Estimate gradient
    $g_j \leftarrow \Big(\sum_{t=0}^{H-1} \nabla_\theta \log(\pi_{\theta_k}(s_t, a_t))\Big)\Big(\sum_{t=0}^{H-1} \gamma^t r_t\Big)$
    Clip gradient
    $\bar{g}_k \leftarrow g_k / \max(1, \frac{||g_k||_2}{C})$
    Add noise
    $\tilde{g_k} \leftarrow \bar{g}_k + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$
    Descent
    $\theta_{k+1} \leftarrow \theta_k + \eta_k \tilde{g}_k$
end
```

Value-based. The DQN algorithm, introduced in [12], approximates state-action functions in a parameterized space:

$$Q \in \mathcal{F}_Q = \{Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R},\ \theta \in \mathbb{R}^d\}$$

Then θ is chosen using empirical risk minimization on a loss inspired from the temporal difference introduced in Q-Learning [21]. For a valid transition (s, a, r, s'), we aim at minimizing the temporal difference:

$$\mathcal{L}(\theta) = ||Q_\theta(s, a) - (r + \gamma \max_{a'} Q_\theta(s', a'))||_2$$

Two other main ideas make DQN work. Since it is an off-policy method, all valid transitions can be used to improve our Q-function. Thus, we can leverage all the observed transitions by adding them to a memory buffer $\mathcal{D}$, and then draw mini-batches of transitions from $\mathcal{D}$ to perform the optimization. In that case, to guarantee differential privacy, noise should be added independently to each sample of the mini-batch.

The other idea is to use a target parameter $\bar{\theta}$, since we are interested in a semi-gradient of $\mathcal{L}$ w.r.t θ, to make the analogy with supervised learning, our estimator is $Q_\theta(s, a)$ and our objective is $r + \gamma \max_{a'} Q_\theta(s', a')$ which shouldn't depend on θ. Combining the two ideas, the loss function is:

$$\mathcal{L}(\theta) = \frac{1}{L} \sum_{x=(s,a,r,s') \in \mathcal{B}} \mathcal{L}(\theta, x), \quad \text{with } \mathcal{L}(\theta, x) = ||Q_\theta(s, a) - (r + \gamma \max_{a'} Q_{\bar{\theta}}(s', a'))||_2$$

The DP-SGD DQN is described in the Algorithm 2.

Algorithm 3: DP-SGD DQN

```
Initialize $\theta_0$ randomly
Target: $\bar{\theta} \leftarrow \theta_0$
$D \leftarrow$ deque()
Draw $s_0 \sim \mu_0$
while $t = 0, 1, \ldots$ do
    Exploration
    $a_t \leftarrow \varepsilon_t$-greedy$(Q_{\theta_t})(s_t)$
    Observe $s_{t+1}$, $r_t$
    Add $(s_t, a_t, r_t, s_{t+1})$ to $\mathcal{D}$
    Compute gradient
    Sample mini-batch $\mathcal{B}$ of size $L$ uniformly from $\mathcal{D}$
    For each $x \in \mathcal{B}$, compute $\mathbf{g}_t(x) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x)$
    Clip gradient
    $\bar{\mathbf{g}}_t(x) \leftarrow \mathbf{g}_t(x) / \max(1, \frac{||\mathbf{g}_t(x)||_2}{C})$
    Add noise to each per-sample gradient and aggregate
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L} \sum_{x \in \mathcal{B}} \Big( \bar{\mathbf{g}}_t(x) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}) \Big)$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{g}_t$
    Update target
    $\bar{\theta} \leftarrow \rho \bar{\theta} + (1 - \rho) \theta_{t+1}$
end
```

3.2 Numerical Experiments

Cartpole. The Cartpole environment was introduced in [3] as an example of control problem with a continuous state space $\mathcal{S}$. We chose to make experiments on this problem as both REINFORCE and DQN are able to solve it in reasonable time. We worked with the implementation of the game proposed in OpenAI's Gym API [4]. A precise description of the game can be found in the Gym documentation.

In Cartpole, the agent needs to learn how to maintain the pole on the cart upright by taking one action in $\mathcal{A} = \{0, 1\}$ (push left, push right) at each step, and obtains a reward of 1 if the pole angles goes out of the range $] - 0.2095, 0.2095[$ or if the car position goes out of the range $] - 2.4, 2.4[$ after the taken action. The agent observes the position and speed of the cart, as well as the angle and angular speed of the pole, thus $\mathcal{S} \subset \mathbb{R}^4$. The game stops if the pole or the car is out of range, or when the agent managed to maintain it for 500 consecutive steps. The initial state s_0 is drawn uniformly in $] - 0.05, +0.05[^4$.

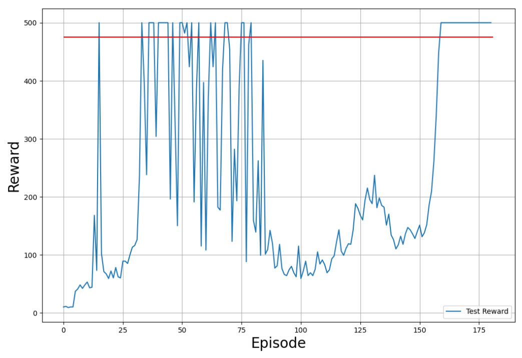
In our experiments, we consider that the agent managed to solve the problem when it achieves to get a mean reward superior to 475 in 25 consecutive trials. We are interested in the impact of the added Gaussian noise on the number of samples needed to converge. The code to reproduce these experiments is available at `https://github.com/AmbroiseOdonnat/RL-Differential-Privacy`.

REINFORCE. For this experiment, we used a two-layer MLP to parameterize the policy π_θ, with an input dimension of 4 (the number of states in $\mathcal{S}$), an hidden dimension of 128, a dropout of rate 0.5, a ReLU activation function, and an output dimension of 2 (the number of actions in $\mathcal{A}$). We first study the convergence of the vanilla REINFORCE, i.e. without DP-SGD. We can see in the Figure 1a that the agent solves the problem on the test environment after roughly 180 episodes.

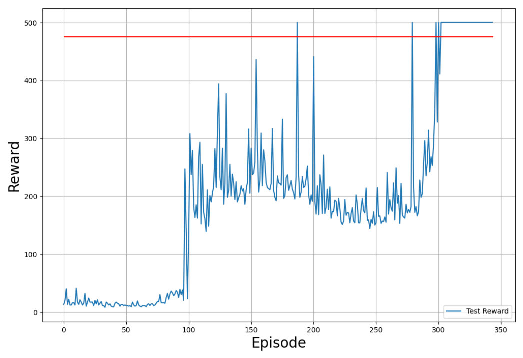
By observing the norm of the gradients during training, we chose a clipping value C value of 8 for the DP-SGD. To study the impact of the noise level σ on the algorithm sample efficiency, we repeat the same training procedure 100 times for $\sigma \in [0.0, 0.1, 0.2, 0.4, 0.6]$ and for 1000 episodes. In the Figure 2a, we plot the evolution of the averaged collected rewards at each noise level when the number of episodes increases.

DQN. For this experiment, we used a two-layer MLP to parameterize Q_θ, with an input dimension of 4 (the number of states in $\mathcal{S}$), an hidden dimension of 128, and ReLU activation, and an output dimension of 2 (the number of actions in $\mathcal{A}$). We used a batch-size of 128. We first train a policy without using DP-SGD, and observe in the Figure 1b that the agent solves the problem on the test environment after roughly 350 episodes. It should be noticed that the memory buffer need 128 episodes to be filled at batch size. Thus, the agent does not learn anything before that.

By observing the norm of the gradients, we chose a clipping value C of 0.1 for the DP-SGD. To study the impact of the noise level σ on the algorithm sample efficiency, we do the exact same experiment than for REINFORCE with 1000 episodes and $\sigma \in [0.0, 0.1, 0.2, 0.4, 0.6]$. However, for computational cost reasons, we only repeated the training at each noise level 30 times. In the Figure 2b, we plot the evolution of the averaged collected rewards at each noise level when the number of episodes increases.



(a) REINFORCE



(b) DQN

Figure 1: Evolution of the agent's reward

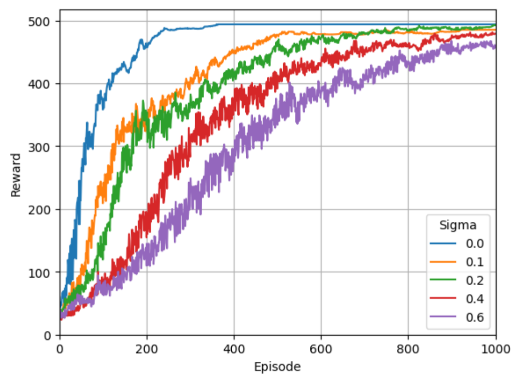
3.3 Numerical Findings

We present empirical results concerning the impact of the noise level on the training procedure. We say that the algorithm converges if it receives a reward greater than 475 for 25 consecutive episodes. Once the convergence is achieved, the reward keeps the value 500. As mentioned above, we conducted the experiments for 1000 episodes for computational cost reasons.

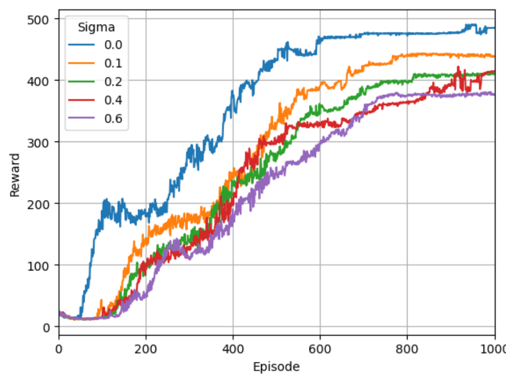
REINFORCE. We plot the learning curves on the test environment with a variety of noise levels in the Figure 2a. We see that the algorithm needs more samples to converge when the noise level increases: the greater σ, the slower it achieves convergence. Denoting N_{stop} the number of episodes before convergence, it should be noticed that the non-private algorithm only needs $N_{\text{stop}} \approx 400$ episodes to reach the stopping criterion, up to $N_{\text{stop}} \approx 1000$ for $\sigma = 0.6$. Despite that, the final return value after 1000 episodes is the same than the non-private version for all noise level. The convergence time corresponds to the episode at which the agent verifies the stopping criterion i.e. it achieved to get a mean reward superior to 475 in 25 consecutive trials. We can see in the Figure 3a the box plot of the convergence time for the 100 training procedures at different noise levels. The trend confirms what could be seen on the learning curves: the more privacy we have i.e. the more noise we add, the slower the learning process.

DQN. We plot the learning curves on the test environment with a variety of noise levels in the Figure 2b. We see that the algorithm needs more samples to converge when the noise level increases: the greater σ, the slower it achieves convergence. Contrary to REINFORCE, the convergence for the non-private version is slower. It implies that more than 1000 episodes would be required to achieve convergence. We could not go further this number of episodes for 30 times due to the heavy computational cost of the procedure. However, we can already observe that the learning curves stabilizes at some point with a sort of plateau. By abuse of notation, we denote N_{stop} the number of episodes to reach this plateau. The non-private DQN verifies $N_{\text{stop}} \approx 600$, while $N_{\text{stop}} \approx 800$ for $\sigma = 0.1$ up to $N_{\text{stop}} \approx 1000$ for $\sigma = 0.6$. We can see in the Figure 3b the box plot of the convergence time for the 30 training procedures at different noise levels. The trend confirms what could be seen on the learning curves: until $\sigma = 0.2$, the more privacy we have i.e. the more noise we add, the slower the learning process. For greater noise level, the learning is not that slower in average but the standard deviation of convergence time increases denoting a less stable process.

Analysis. It is clear that the the noise level has an impact on the learning process. The more noise we add, the more private the algorithm becomes and the lower the sample efficiency of the agents. Another thing to note is that REINFORCE is less impacted by this perturbation than DQN. As DQN relies on bootstrapping, one could argue that adding noise as a differential privacy mechanism has a stronger influence.

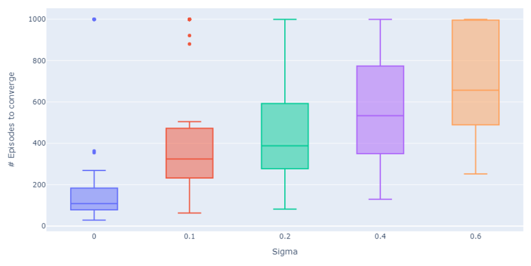


(a) REINFORCE

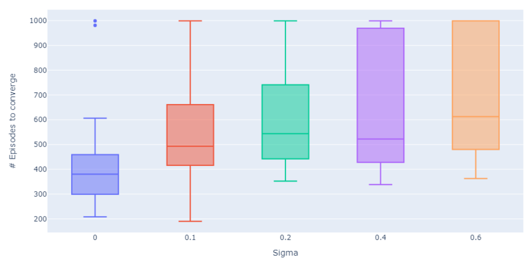


(b) DQN

Figure 2: Impact of noise level on the evolution of the average agent's reward



(a) REINFORCE



(b) DQN

Figure 3: Impact of noise level on the convergence time

4 Reinforcement Learning with Local Differential Privacy Learning

In this section, we propose and investigate a LDP approach to Reinforcement Learning. We work with the same algorithms and the same environment as in the Section 3, in order to propose a rigorous comparison of the two approaches to differential privacy.

4.1 LDP paradigm for RL

The main idea of LDP, is to induce privacy by directly injecting noise on the inputs of the model, whereas in the central differential privacy framework, the model used to see the inputs but the learning procedure was altered to make the outputs private.

In the RL framework, we want to preserve privacy on the agent's observations, namely states and rewards. One standard procedure is to make the observations private using a Gaussian Mechanism then classic RL algorithms can be used on those perturbed signals [10, 11].

For our specific study on the Cartpole environment, the reward is discrete and simply indicates if the agent failed. We decide to keep it noiseless, and focus on perturbing the states.

4.2 Algorithms

Policy-based. To adapt the REINFORCE algorithm described in the Section 3.1 and make it respect LDP, we modify the trajectory generation procedure. It is described in the Algorithm 4.

Value-based. For the DQN algorithm, using LDP is more convenient than DP-SGD, as we don't need heavy modifications to work with batches. Indeed, it is sufficient to add noise the first time we add a transition to the memory buffer, and to use the classical DQN algorithm. This procedure is described in the Algorithm 5.

Algorithm 4: LDP REINFORCE

```
Initialize $\theta_0$ randomly
while $k = 0, 1, \dots$ do
    Draw $s_0 \sim \mu_0$
    $s_0 \leftarrow s_0 + \mathcal{N}(0, \Sigma)$
    Generate trajectory
    for $t = 0, \dots, H-1$ do
        $a_t \leftarrow \pi_\theta(s_t)$
        Observe $s_{t+1}$, $r_t$
        $s_{t+1} \leftarrow s_{t+1} + \mathcal{N}(0, \Sigma)$
    end
    $\tau \leftarrow$
      $(s_0, a_o, r_0, \dots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$
    Estimate gradient
    $g_j \leftarrow$
      $\sum_{t=0}^{H-1} \nabla_\theta \log(\pi_{\theta_k}(s_t, a_t)) \sum_{t=0}^{H-1} \gamma^t r_t$
    Descent
    $\theta_{k+1} \leftarrow \theta_k + \eta_k \tilde{g}_k$
end
```

Algorithm 5: LDP DQN

```
Initialize $\theta_0$ randomly
Target: $\bar{\theta} \leftarrow \theta_0$
$D \leftarrow$ deque()
Draw $s_0 \sim \mu_0$
while $t = 0, 1, \dots$ do
    Exploration
    $a_t \leftarrow \varepsilon_t$-greedy$(Q_{\theta_t})(s_t)$
    Observe $s_{t+1}$, $r_t$
    $s_{t+1} \leftarrow s_{t+1} + \mathcal{N}(0, \Sigma)$
    Add $(s_t, a_t, r_t, s_{t+1})$ to $\mathcal{D}$
    Compute gradient
    Sample mini-batch $\mathcal{B}$ of size $L$
      uniformly from $\mathcal{D}$
    $g_t \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta)$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{g}_t$
    Update target
    $\bar{\theta} \leftarrow \rho\bar{\theta} + (1-\rho)\theta_{t+1}$
end
```

4.3 Numerical Experiments

We repeat the experiment described in the Section 3.2, with the same models and parameters, to provide a comparison of the impact of privacy on the learning efficiency for both classes of algorithms.

Noise calibration. In Cartpole, the states are 4-dimensional, $\mathcal{S} \subset \mathbb{R}^4$, and each component of the state vector is a physical quantity, with its own range. For example, $s^{(0)} \in [-4.8, 4.8]$ is the position of the center of the cart, and $s^{(2)} \in [-0.418, 0.418]$ is the angle of the pole, in radians. Because of this heterogeneity, the noise added to the states must be calibrated.

To obtain an accurate representation of the range of each component of the state vector, we collected each state value during the training of a non-private REINFORCE algorithm for 180 episodes. The box plots in the Figure 4 indicate the typical range for s on 180 episodes. We decided to take as reference the interquartile ranges (IQR) of each component shown in the Table 1. The noise injected for LDP corresponds to a proportion of this typical range, specifically, $\Sigma = \sigma D$, where D is the diagonal matrix containing the IQR of each component.

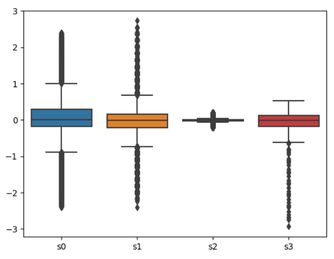


Figure 4: Values taken by the states

Implementation Details This time, we repeat the training procedure respectively 100 and 50 times for REINFORCE and DQN at each noise level $\sigma \in [0.0, 0.01, 0.1, 0.2, 0.3]$. We also test $\sigma \in \{2.0, 5.0\}$ for REINFORCE.

Component	$s^{(0)}$	$s^{(1)}$	$s^{(2)}$	$s^{(3)}$
IQR	0.55	0.39	0.05	0.32

Table 1: Reference values for noise calibration

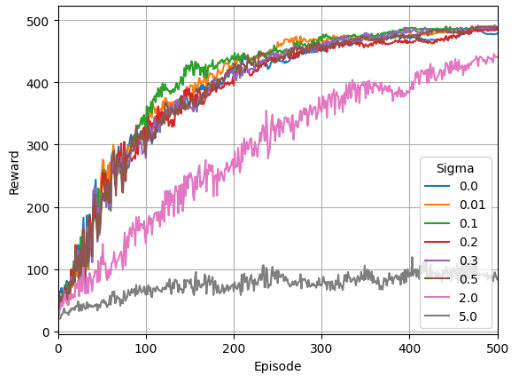
4.4 Numerical Findings

We present empirical results concerning the impact of the noise level on the training procedure. We say that the algorithm converges if it receives a reward greater than 475 for 25 consecutive episodes. Once the convergence is achieved, the reward is kept at a constant value of 500. As mentioned above, we conducted the experiments with 1000 episodes for computational cost reasons.

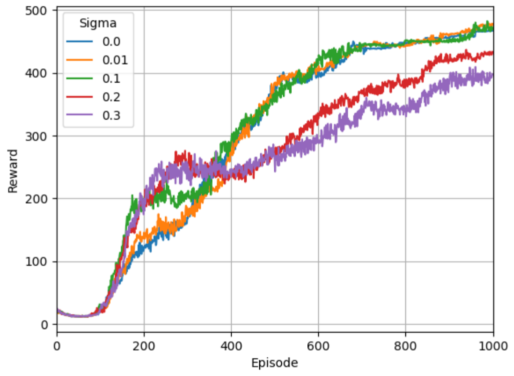
REINFORCE. We plot the learning curves on the test environment with a variety of noise levels in the Figure 5a. We zoomed on the first 500 episodes for a better visualization but the entire plot can be found in the Figure 7 of Appendix A. Denoting N_{stop} the number of episodes before convergence, it should be noticed that the non-private algorithm only needs $N_{\text{stop}} \approx 500$ episodes to reach the stopping criterion. We see that for $\sigma \in [0.0, 0.3]$, the sample efficiency of the algorithm is not changed. For $\sigma = 0.01$, convergence is even achieved before the non-private version. To test the limit of REINFORCE, we also conducted the experiment for higher noise levels. We can see that the learning process becomes slower with $\sigma = 2.0$ and that for a very high noise level of $\sigma = 5.0$, the algorithm does not converge in the 1000 episodes. The convergence time corresponds to the first episode at which the agent verifies the stopping criterion. We can see in the Figure 6a the box plots of the convergence time for the 100 training procedures at different noise levels. The trend confirms what could be seen on the learning curves: except for very large values of σ, the sample efficiency of the agents is not disturbed by the privacy increase. With $\sigma \geq 2$, we return to the situation where the increase in noise slows the learning process down. The impact of noise level on the evolution of the median reward of REINFORCE can also be seen in the Figure 8c of Appendix A.

DQN. We plot the learning curves on the test environment with a variety of noise levels in the Figure 5b. Contrary to REINFORCE, the convergence for the non-private version is slower. It implies that more than 1000 episodes would be required to achieve convergence. We could not go further this number of episodes for 50 times due to the heavy computational cost of the procedure, but we can still see that the returned reward increases toward 500. We observe two regimes: up to 350 episodes, the higher the noise level, the higher the return reward. After this regime, we see that the learning process is harder with the increase in noise: the greater σ, the slower the return increases towards 500. A very interesting phenomenon is that in this second regime, adding a small noise such as $\sigma \in \{0.01, 0.1\}$ slightly improves the learning compared to the non-private algorithm. We can see in the Figure 6b the box plots of the convergence time for the 50 training procedures at different noise levels. We observe the same trend as in the central differential privacy case: the increase in noise reduces the sample efficiency of the algorithm. The impact of noise level on the evolution of the median reward of DQN can also be seen in the Figure 8d of Appendix A.

Analysis. The two algorithms are very different in terms of behavior. While REINFORCE is robust to the noise, up to $\sigma > 2.0$, we observe that for most noise levels, the sample efficiency of DQN is deteriorated. An interesting phenomenon is that for small values ($\sigma \in 0.01, 0.1\}$), DQN converges with less samples. We can interpret this as a regularization effect. Indeed, in the same fashion that noise can be used as a regularization method in Deep Learning [13], e.g. in dropouts, one could think that adding a reasonable noise on what the DQN observes helps the learning in our experiments.

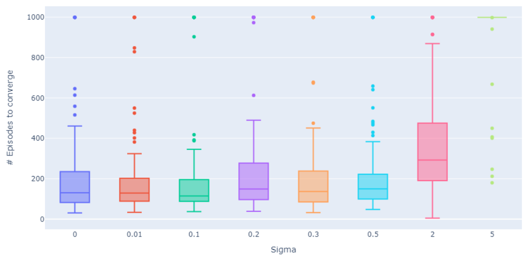


(a) REINFORCE

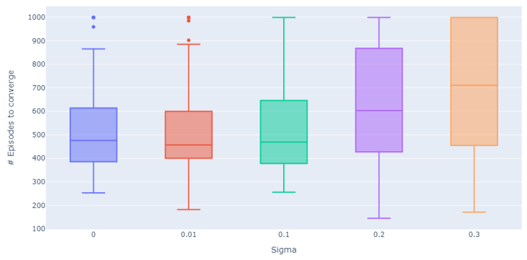


(b) DQN

Figure 5: Impact of noise level on the evolution of the average agent's reward



(a) REINFORCE



(b) DQN

Figure 6: Impact of noise level on the convergence time

5 Conclusion and Future Work

In this paper, we presented a method for incorporating differential privacy into reinforcement learning methods with continuous state-spaces. We adapted two popular algorithms, REINFORCE and Deep Q-Network, to the framework of central and local differential privacy. Through experimentation on the Cartpole environment, we found that although privacy does have an impact on sample efficiency, the agent is still able to learn an optimal policy with reasonable levels of privacy. Our findings also suggest that the incremental nature of DQN makes it more sensitive to noise compared to REINFORCE.

An extension of our work could include theoretical analysis to derive regret bounds for our private algorithms. On the experimental side, it would be informative to evaluate our methods on more complex environments in order to investigate their scalability.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2016.

[2] Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. Local Differential Privacy for Deep Learning. *IEEE Internet of Things Journal*, 7(7):5827–5842, jul 2020.

[3] 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.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

[5] Xiaoyu Chen, Kai Zheng, Zixin Zhou, Yunchang Yang, Wei Chen, and Liwei Wang. (locally) differentially private combinatorial semi-bandits. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[6] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local Privacy, Data Processing Inequalities, and Statistical Minimax Rates, 2013.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] Amir-massoud Farahmand. Action-Gap Phenomenon in Reinforcement Learning. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 172–180, Red Hook, NY, USA, 2011. Curran Associates Inc.

[9] Pratik Gajane, Tanguy Urvoy, and Emilie Kaufmann. Corrupt bandits for preserving local privacy. In Firdaus Janoos, Mehryar Mohri, and Karthik Sridharan, editors, *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, pages 387–412. PMLR, 07–09 Apr 2018.

[10] Evrard Garcelon, Vianney Perchet, Ciara Pike-Burke, and Matteo Pirotta. Local Differential Privacy for Regret Minimization in Reinforcement Learning, 2020.

[11] Chonghua Liao, Jiafan He, and Quanquan Gu. Locally differentially private reinforcement learning for linear mixture markov decision processes. *CoRR*, abs/2110.10133, 2021.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013.

[13] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization, 2017.

[14] Xinlei Pan, Weiyao Wang, Xiaoshuai Zhang, Bo Li, Jinfeng Yi, and Dawn Song. How You Act Tells a Lot: Privacy-Leakage Attack on Deep Reinforcement Learning, 2019.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

[16] Kritika Prakash, Fiza Husain, Praveen Paruchuri, and Sujit P. Gujar. How private is your rl policy? an inverse rl based analysis framework, 2021.

[17] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[19] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[20] Giuseppe Vietri, Borja Balle, Akshay Krishnamurthy, and Zhiwei Steven Wu. Private Reinforcement Learning with PAC and Regret Guarantees, 2020.

[21] Christopher Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, 8:279–292, 05 1992.

A Additional Figures

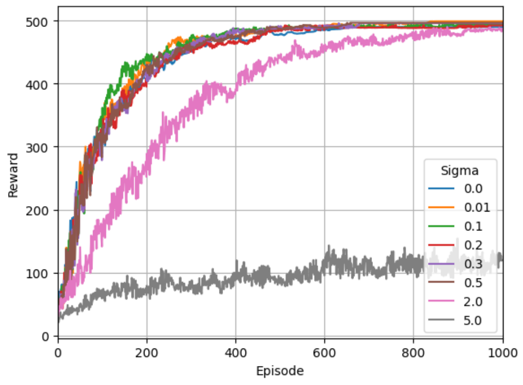
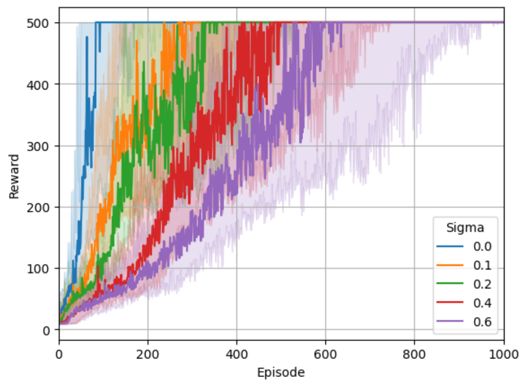
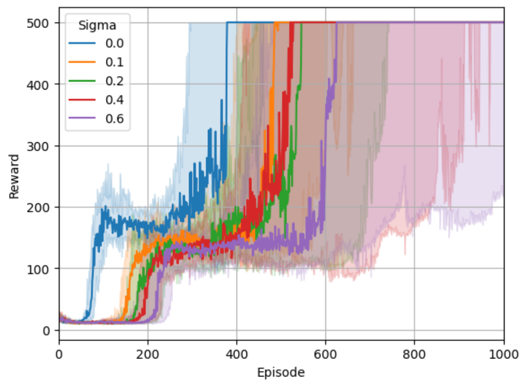


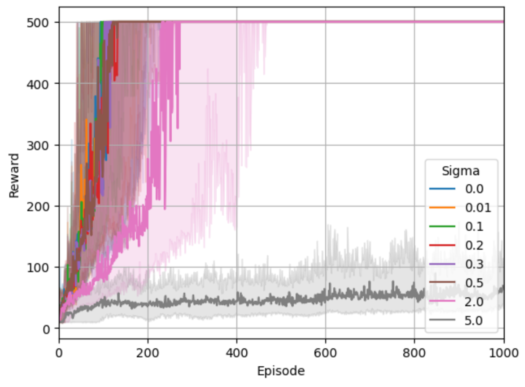
Figure 7: Impact of noise level on the evolution of the average reward for REINFORCE with LDP



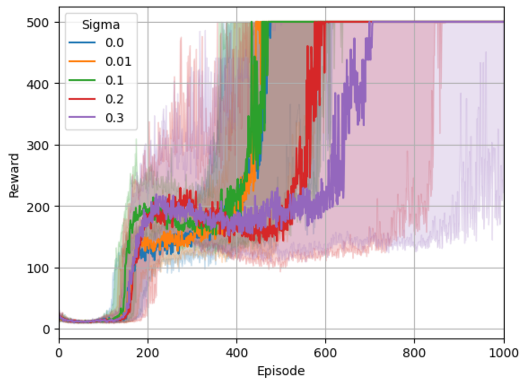
(a) DP-SGD REINFORCE



(b) DP-SGD DQN



(c) REINFORCE with LDP



(d) DQN with LDP

Figure 8: Impact of noise level on the evolution of the median agent's reward