Robot Learning Lab 4 - Policy gradient and Actor critic algorithms

Borella Simone (S317774)
Computer engineering - Artificial Intelligence
Polytechinc University of Turin, Italy
s317774@studenti.polito.it

1 Abstarct

This laboratory focuses on the implementation of policy gradient reinforcement learning algorithms, with a specific emphasis on the REINFORCE algorithm. The exploration extends to more advanced techniques such as Proximal Policy Optimization (PPO) and Soft Actor Critic (SAC), using well-known libraries like Stable-Baselines3. The objective is to provide a comprehensive understanding of these algorithms and their practical application in reinforcement learning tasks, taking in exam the Cartpole environment with continuous action space.

2 Cartpole environment

The well-known cartpole environment is exploited for experiments on the explained algorithms, but the environment is modified to have a continuous action space instead of discrete.

$$a_{ds} = [F_{left}, F_{right}] \rightarrow a_{cs} = [F] \tag{1}$$

While in the discrete action space version only a constant force to the left or to the right equal to $10N$ is possible, in new defined continuous action space a unique value describes a force applied to the cart, to the right if positive, to the left if negative.

3 REINFORCE

3.1 Introduction

The REINFORCE algorithm is a Monte Carlo policy gradient method used for reinforcement learning in control tasks. The algorithm is designed to optimize the policy of an agent to maximize the expected cumulative reward over time. The REINFORCE method operates by estimating the gradient of the expected cumulative reward, also called return, with respect to the policy parameters and trying to optimize the policy by updating its parameters.

A policy is a probability distribution that describe the probability to take an action a given a state s, and as it needs to be optimized, the policy is parametric, where θ represent parameters.

$$\pi(a|s, \theta) \tag{2}$$

The objective function is represented by the expected cumulative return, where τ is a trajectory computed following the policy π_θ, γ is the discount factor and R_t is the immediate reward at timestep t.

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t R_t \right] \tag{3}$$

The gradient of the objective function is then computed as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log\pi(a_t|s_t,\theta)\cdot\left(\sum_{t'=t}^{T}\gamma^{t'-t}R_{t'}\right)\right] \tag{4}$$

The parameters describing the policy are then updated along the way of the objective function gradient:

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta) \tag{5}$$

3.2 Baseline

In order to reduce the variance of the gradient estimates and potentially improve the stability and convergence of the learning process a baseline is needed. A baseline is a term used to refer to a value that is subtracted from the estimated returns in the policy gradient update.

The baseline helps the algorithm differentiate between actions that have similar immediate returns but differ in their overall advantage, identifying actions that are more likely to lead to favorable outcomes in the future, even if their immediate rewards are similar.

In this way the training variance is decreased and stability increased.

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t R_t\right] - b(s) \tag{6}$$

The choice of a baseline is case dependent and here are listed some common baselines:

- **Constant baseline:** simple choice, can be defined doing a benchmark with significant constant baselines.
- **Average reward:** it assumes a well defined average reward.
- **Action-value function:** it provides a measure of advantage at each timestep, but $Q(s,a)$ should be estimated.
- **State-value function:** it provides a measure of advantage at each timestep, but $V(s)$ should be estimated.

3.3 Experiments

The implementation of the REINFORCE algorithm exploits a neural network to approximate the policy action return given a state. In the training phase the policy return a gaussian distribution with the actual optimal action value as mean and constant variance $\sigma^2 = 5$.

Figure 1 shows the training process of 5 different agents. In this figure is clear that the variance is high and compared to baseline based solutions also performance is worse.

A constant baseline equal to $b = 20$ is exploited and as illustrated in Figure 2 the variance is decreased and also convergence and stability properties are better.

Benchmarking different kinds of constant baselines in the range $[0, 250]$, it can be noticed that the variance decrease gradually and the best performance is reached with constant baseline $b = 50$. Figure 5 and Figure 4 illustrate respectively the training of multiple agents with constant baseline $b = 50$ and constant baseline $b = 250$.

Finally another implementation a standardization is performed by normalizing the return to zero mean and unit variance by subtracting the mean of all discounted rewards and dividing by the standard deviation.

$$J_{standardized} = \frac{J - \bar{J}}{\sigma_J} \tag{7}$$

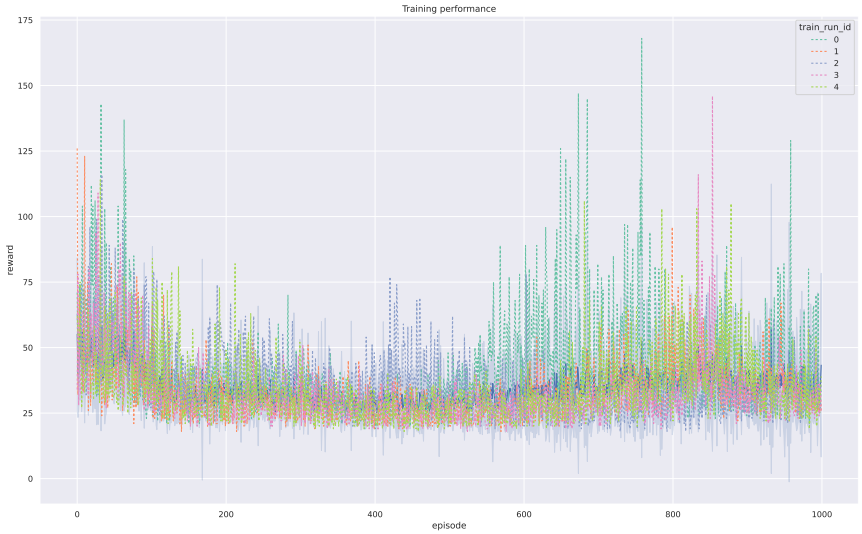

Figure 1: Multiple training processes without baseline

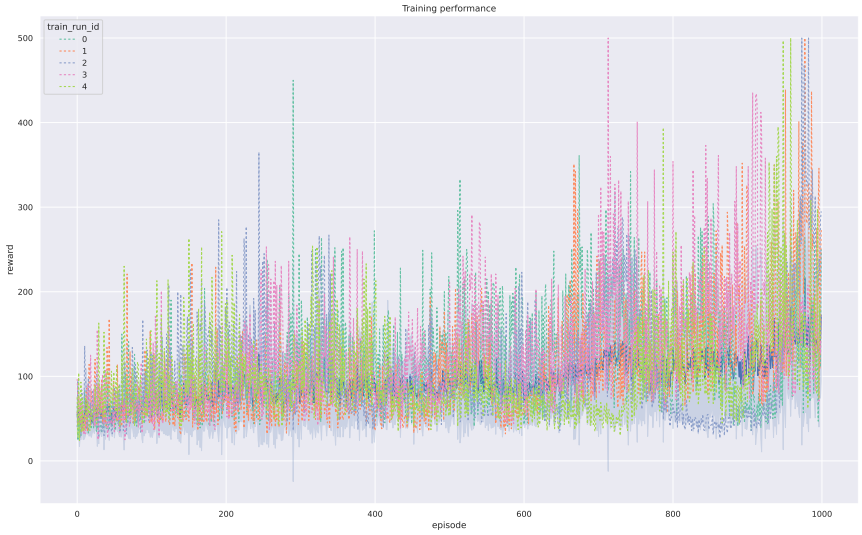

Figure 2: Multiple training processes with constant baseline 20

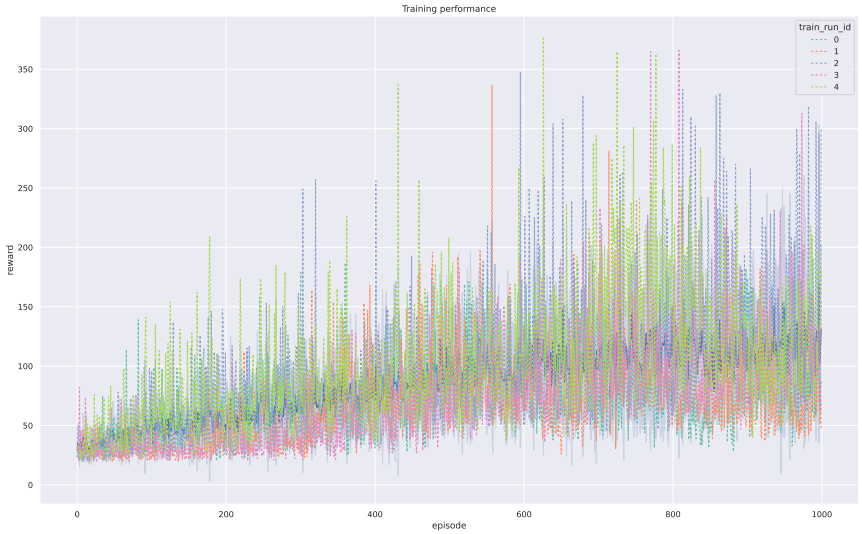

Figure 3: Multiple training processes with constant baseline 50

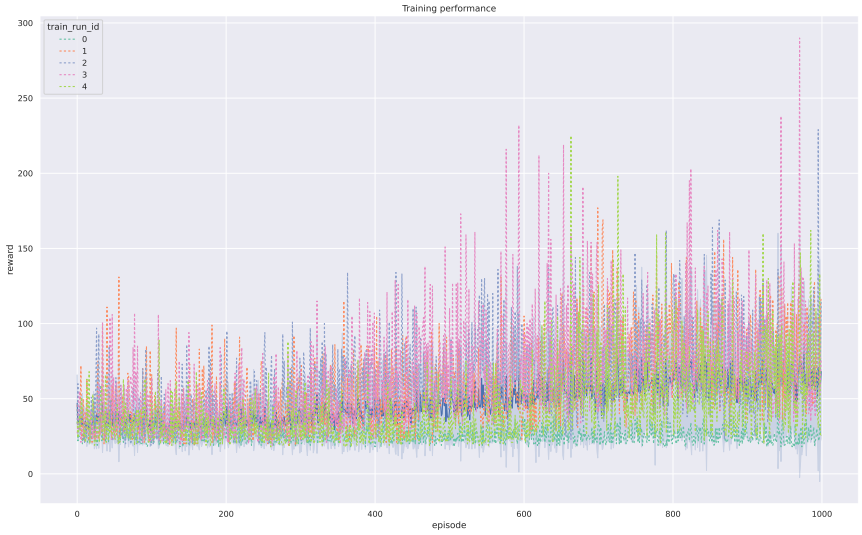

Figure 4: Multiple training processes with constant baseline 250

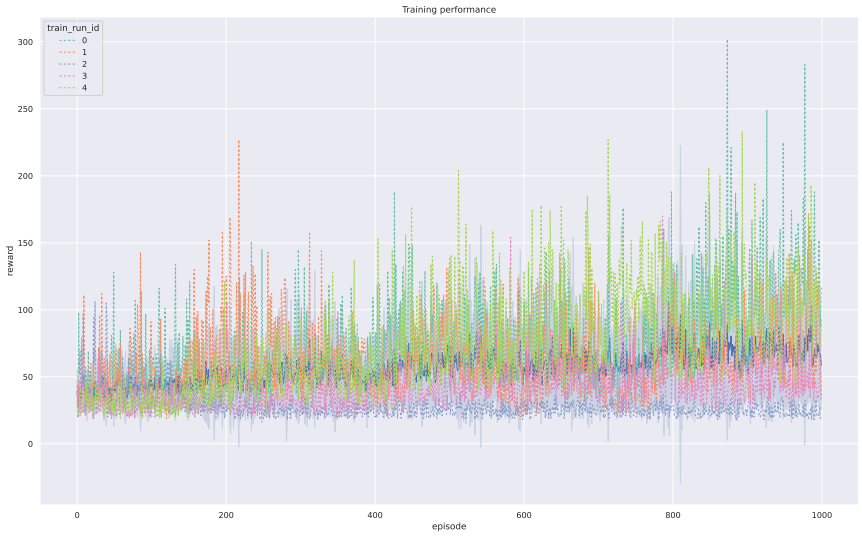

Figure 5: Multiple training processes with standardization and no baseline

Here are listed some results by training different models with the techniques explained above over 5000 episodes.

Trained model: *model_no_baseline.mdl*
Test results: Average test reward: 68.64 episode length: 68.64

Trained model: *model_baseline_20.mdl*
Test results: Average test reward: 284.85 episode length: 284.85

Trained model: *model_baseline_50.mdl*
Test results: Average test reward: 500.0 episode length: 500.0

Trained model: *model_baseline_250.mdl*
Test results: Average test reward: 77.0 episode length: 77.0

Trained model: *model_standardization.mdl*
Test results: Average test reward: 242.59 episode length: 242.59

3.4 Discrete action space

These experiments are made on a continuous action space and so the policy is a function that returns a continue probability distribution over actions given a certain state.

Of course the REINFORCE algorithm could also be applied to the original gym cartpole environment, with discrete action space. In particular the policy would be a function that given a state returns a discrete probability distribution over all possible actions.

Talking about the implementation the neural network that approximates the policy changes slightly its structure, having as many output as the number of discrete actions are. Then real values returned by the neural network are then given to softmax function, which aim is to convert a vector of real numbers into a probability distribution.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{8}$$

3.5 Real-world control problems

There are some relevant issues when trying to make a control with reinforcement learning techniques:

- **Continuous observation and action space:** dealing with the real world means dealing with continuous dimensions and so training a model with continuous observation space and action space could be more challenging.
- **Sensors noise and uncertainty:** real sensors are noisy and the observation taken by the training process is affected by uncertainty and so is different from the ground truth.
- **Non stationary system:** dynamics of real system is never stationary.
- **Latency:** observation with sensor measurements and training steps require time and so both training and deploying of reinforcement learning control system could lead in some errors due to this fact if not controlled.
- **Safety and simulation training:** sometimes training a control with reinforcement learning is not feasible for safety reasons. A simulation could be use to train a model to deploy then in a real control filling the gap between the simulation environment and real world, but is not always possible due to the difficulty of this task.

Apart from these general problems dealing with reinforcement learning there could be other algorithm-specific problems about REINFORCE with the current implementation.

Actually the reward is +1 if the cartpole survive and 0 if not, which means that the actual objective of the control is to make the system stay alive.

A problem could be the way it stays alive. Without specifying other contributions to the reward function that increase the reward if the actual state is optimal, the way the policy behave is unexpected (e.g. it could oscillate around the middle point $x = 0$ and with a velocity v higher than the expected).

Once fixed the reward function to get a more stable behaviour we could again lead into a suboptimal policy.

Survival-focused rewards might lead to policies that prioritize conservative and risk-averse actions, even in situations where more dynamic or efficient behaviors are possible. This can result in suboptimal performance.

4 Actor-Critic Methods

4.1 Introduction

Actor-Critic methods combine the advantages of value-based and policy-based reinforcement learning approaches. This hybrid approach aims to leverage the strengths of each paradigm, leading to more stable and efficient learning in complex environments.

The actor's primary responsibility is to learn a policy that maps states to actions. The training objective of the actor is to improve the policy so that it selects actions that lead to higher cumulative rewards. This is often done by adjusting the parameters of the policy through gradient ascent, guided by the expected advantage or the TD (temporal difference) error.

The critic's main role is to evaluate the actions taken by the actor by estimating the state-value function or the action-value function.

The actor selects actions based on the policy, and the critic evaluates these actions by estimating their associated values. The feedback from the critic, in the form of advantage or TD error, is used to update the actor's policy to encourage actions that lead to higher rewards. The critic's value function is updated to better approximate the true values of states or state-action pairs based on the observed returns from the environment.

4.2 PPO

Proximal Policy Optimization (PPO) belongs to the family of policy optimization algorithms and is specifically designed to address some of the issues associated with earlier algorithms like Trust Region Policy Optimization (TRPO).

PPO focuses on optimizing the policy, optimizing the policy parameters to increase the likelihood of good actions that lead to higher rewards.

PPO introduces a constraint on the size of policy updates to prevent large policy changes that may destabilize the learning process. This constraint helps maintain a "trust region" within which the policy is updated.

To keep this "trust region" a clipped surrogate objective function, that approximates the policy update, encourages the new policy to be similar to the old policy.

The surrogate objective function is weighted by advantages (indicating the quality of actions) estimated thanks to the estimated state-value function.

PPO typically uses minibatches of experiences for policy updates, which can enhance the stability of the learning process.

PPO is considered to be sample efficient.

4.3 SAC

Soft Actor-Critic (SAC) is an off-policy algorithm designed for continuous action spaces. It aims to maximize the expected sum of cumulative rewards while taking into account both the primary task and the entropy of the policy.

The actor and the critic are responsible for policy estimation and state-value function estimation. Soft Actor-Critic introduces entropy regularization to encourage exploration. The entropy term ensures that the policy is not too deterministic, promoting a balance between exploration and exploitation. The entropy term is weighted by a temperature parameter, which allows for adjusting the importance of entropy in the overall objective. A higher temperature puts more emphasis on exploration, while a lower temperature focuses more on maximizing the expected cumulative reward.

SAC is known for its sample efficiency, meaning it can achieve good performance with fewer samples compared to some other reinforcement learning algorithms.

4.4 Experiments

Stable-Baselines 3 stands out as a cutting-edge reinforcement learning (RL) library that provides effective and reliable implementations of a range of algorithms. Notably, it provide robust implementations for Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC).

Figure 6 shows the learning curve reached with PPO. The smoothness of the learning curve confirm the convergence property due to the optimization of the policy which stay close to a "trust region", which on the other side limits slightly the sample efficiency and convergence speed.

In Figure 7 instead we can see a different sample efficiency which characterize this algorithm, proposing a less stable solution, trying to encourage exploration.

Compared to the REINFORCE training process with constant baseline $b = 50$ illustrated in Figure 8, both PPO and SAC results to be more stable and with better convergence properties.

Actor-critic solutions performances are better than the performances of REINFORCE algorithm thanks to different factors.

REINFORCE vs PPO:

- **Sample efficiency:** PPO reuses collected experiences more effectively through multiple updates, and so is more sample efficient.
- **Convergence:** PPO often uses an adaptive learning rate mechanism, adjusting the step size based on training progress, increasing convergence.
- **Stability:** PPO incorporates a clipped surrogate objective, enhancing stability by constraining policy updates (trust region). PPO's policy update is based on the ratio of new to old policy probabilities, with a modification to prevent large updates.

REINFORCE vs SAC:

- **Sample efficiency:** SAC utilizes entropy regularization, encouraging exploration and allowing for more efficient use of collected samples.
- **Convergence:** SAC incorporates a critic (Q-function) in addition to the actor, providing a more sophisticated estimate of the value of different actions enhancing learning compared to the purely policy-based approach of REINFORCE.
- **Stability:** SAC incorporates entropy regularization, a target value for the action-value function, and a temperature parameter that can be adjusted to control exploration.

Here are testing results for PPO and SAC models trained over 150000 timesteps and tested over 100 timesteps:

Trained model: *model_PPO.mdl*
Test results: Test reward (avg +/- std): (500.0 +/- 0.0) - Num episodes: 100

Trained model: *model_SAC.mdl*
Test results: Test reward (avg +/- std): (498.98 +/- 7.679817706169853) - Num episodes: 100

Here is reported again the test results of the REINFORCE implementation with constant baseline $b = 50$:

Trained model: *model_baseline_50.mdl*
Test results: Average test reward: 500.0 episode length: 500.0

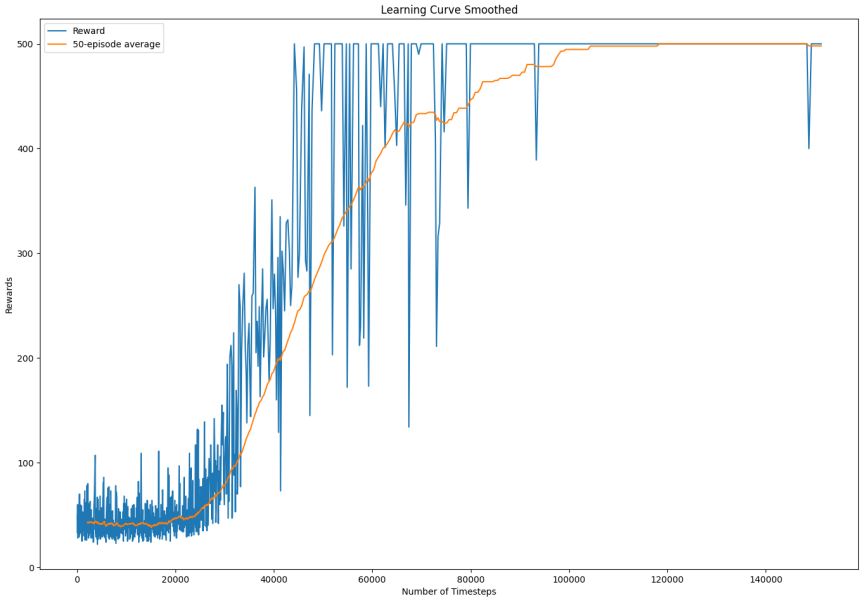

Figure 6: Training process with PPO

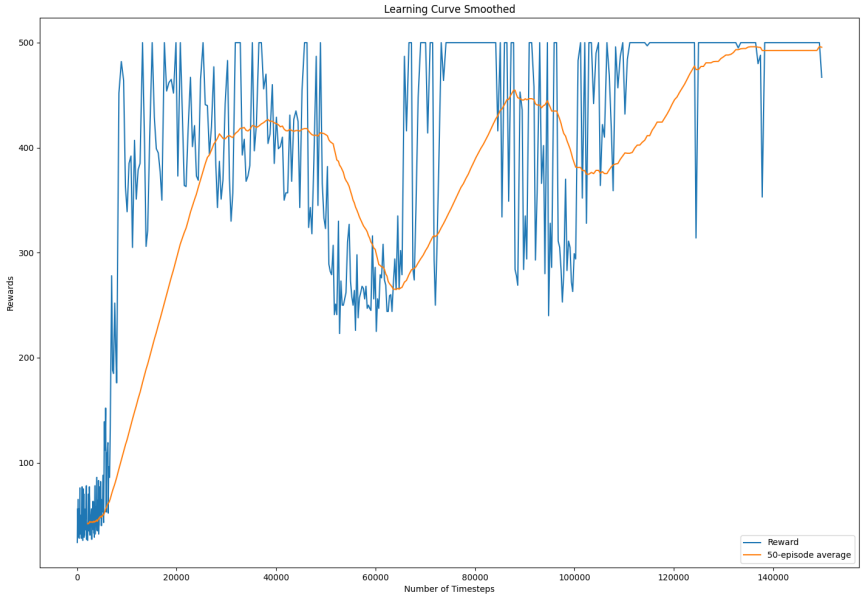

Figure 7: Training process with SAC

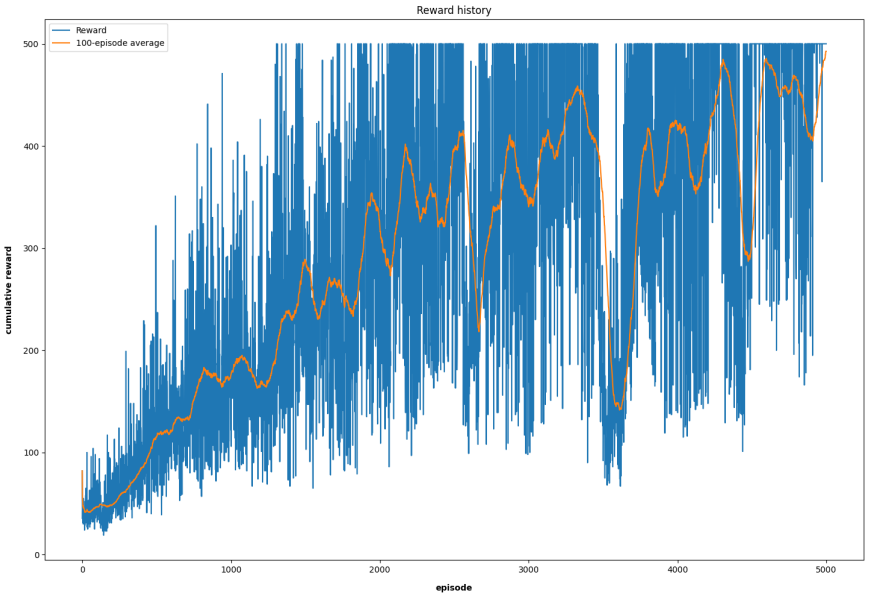

Figure 8: Training process with REINFORCE