We introduce a new algorithm of reinforcement learning in continuous state and action spaces. In order to construct a control policy, the algorithm utilizes the entire history of agent-environment interaction. The control policy is thus estimated from the entire available information, and is not a result of stochastic convergence as in classical reinforcement learning approaches. The policy is derived from the history directly, not through a model of the environment.

We test our algorithm in the Cart-Pole Swing-Up simulated environment. The algorithm learns to control this plant in about 100 trials, which corresponds to 15 minutes of plant’s real time. This time is several times shorter than the one required by other algorithms.

The Goal
We look for a reinforcement learning algorithm that will work in the following setup.
1. The controller of some machine has unknown parameters.
2. The initial knowledge on the plant’s model is poor.
3. An iterative way is to employ
4. The controller of some machine has unknown parameters.
   The initial knowledge on the plant’s model is poor.
   Perform the control
5. Cart-Pole Swing-Up

Algorithm assumptions
The assumed setup has several implications. On one hand, we do not expect that the training procedure will be instantaneous. On the other hand, for economic reasons, the duration of the training must be short, i.e. days are better than weeks and hours are better than days. We try to design an efficient algorithm based on the following assumptions:
1. The initial knowledge on the plant’s model is poor.
2. The entire training history can be stored and utilized in some computational process.
   Contemporary computers are powerful enough to store and process large amounts of data. There is no requirement that consecutive events from the training session have to be utilized immediately and ‘forgotten’. Instead, they can be ‘remembered’ and processed repeatedly during the training session.

An actor-critic framework
The algorithm uses two data structures called actor and critic.

Actor
The actor consists of a neural network and a family of distributions parameterized by the output of the network.
For the given network \( \theta \) and the family of distributions \( \psi \), the following items are dependent just on the network’s weights \( \psi \).
- Control policy \( \pi(\theta) \).
- Stationary state distribution \( \psi(s, \theta) \).
- Performance measured by a pre-action-value function defined similarly to the action-value function

\[
V^\pi(s) = \mathbb{E}_t \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \mid s_0 = s, \pi \sim \pi(\theta) \right] = \mathbb{E}_t \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]

Intensive Randomized Policy Optimizer
The IRPO maximizes \( V^\pi(s, \theta) \), a batch estimate of \( V^\pi(s, \theta) \) based on the entire history of agent-environment interactions.

Simulations, Cart-Pole Swing-Up
The Cart-Pole Swing-Up is a modification of the inverted pendulum frequently used as a benchmark for the reinforcement learning algorithms. The goal of control is to avoid hitting the track bounds, swing the pole turn it up and stabilize upwards. The reward in this problem is typically equal to the elevation of the pole tip. Initially, the swinging pole leaves, and the rewards are close to 0. When finally the pole is stabilized upwards, the reward is close to 1. Controller’s interventions take place every 0.1s.

Note the relation between \( V^\pi(s, \theta) \) and the value function \( V^\pi(s, \theta) \) defined in a standard way: \( V^\pi(s, \theta) = \mathbb{E}_t \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) \mid s_0 = s, \pi \sim \pi(\theta) \right] \).
The actor’s objective is to determine a policy of maximal performance averaged by state distribution. It should minimize
\[
\mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]
with respect to \( \pi \).

The critic consists of a neural network and a family of distributions parameterized by the output of the network.
The critic’s objective is to become a mean-square approximation of the stationary state distribution. It should maximize
\[
\mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]
with respect to \( \theta \).

Calculated the values in real time: 1. Define the control action \( a_t \):
\[
a_t \sim \pi^\phi(\theta(s_t))
\]
2. Perform the control \( a_t \) and observe the next state \( s_{t+1} \) and the reinforcement \( r_{t+1} \).
3. Add the quintet \( (s_t, a_t, r_{t+1}, s_{t+1}, \theta(s_t)) \) to the dataset. \( \theta(s) \) is the same value as the one that served in action selection.
4. Set \( t = t + 1 \) and repeat from Step 1.

In the meantime, run the approximate policy iteration:
1. Policy evaluation. Adjust \( \theta(s) \) for \( \hat{V}(s) \) to minimize an estimator of \( \hat{V}(s) \):
\[
\mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]
for all \( i \in \{1, \ldots, T\} \) to minimize
\[
\hat{V}(s) = \mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]
2. Policy improvement. Adjust \( \theta(s) \) for \( \hat{V}(s) \) to maximize an estimator of \( \hat{V}(s) \):
\[
\mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]
for all \( i \in \{1, \ldots, T\} \) and a fixed \( \pi \) to maximize
\[
\hat{V}(s) = \mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]

The Intensive Randomized Policy Optimizer algorithm.

Off-line evaluation and optimization of randomized policy
We are given the history of the agent-environment interaction up to the moment \( t \). The states visited \( \{s_i, \pi_i \mid i = 1, \ldots, t\} \), the rewards associated with each state \( \{r_i \mid i = 1, \ldots, t\} \), and the control actions that have been performed, namely \( \{\pi_i \mid i = 1, \ldots, t\} \). The actions have been chosen according to \( \pi \), where \( \{i, \pi_i \mid i = 1, \ldots, T\} \) are given.

There are two problems: first, to estimate \( \hat{V} \), the value function of the current policy and second, to optimize the policy. We reduce both to the estimation problems and solve them with use of importance sampling.

Let denote:
\[
\hat{q}_t = \hat{r}_t + \gamma \hat{V}(s_{t+1}, \pi_{t+1})
\]

To evaluate the policy (i.e. to train the critic) we need to minimize
\[
\hat{c}_t = \mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]
with respect to \( \pi \). We minimize the estimates:
\[
\hat{c}_t = \mathbb{E}_s \left[ \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi_i(s_i)) \mid s_0 = s, \pi \sim \pi(\theta) \right]
\]

To optimize the policy (i.e. to train the actor) we need to minimize
\[
\mathbb{E}_s \left[ r(s_t, a_t) \mid a_t \sim \pi(\theta(s_t)) \right]
\]
with respect to \( \pi \). We minimize the estimates:
\[
\mathbb{E}_s \left[ r(s_t, a_t) \mid a_t \sim \pi(\theta(s_t)) \right]
\]