NIPS 2017: Learning to Run with PPO and Reward Shaping

Adam Stelmaszczyk[1], Piotr Jarosik[2]

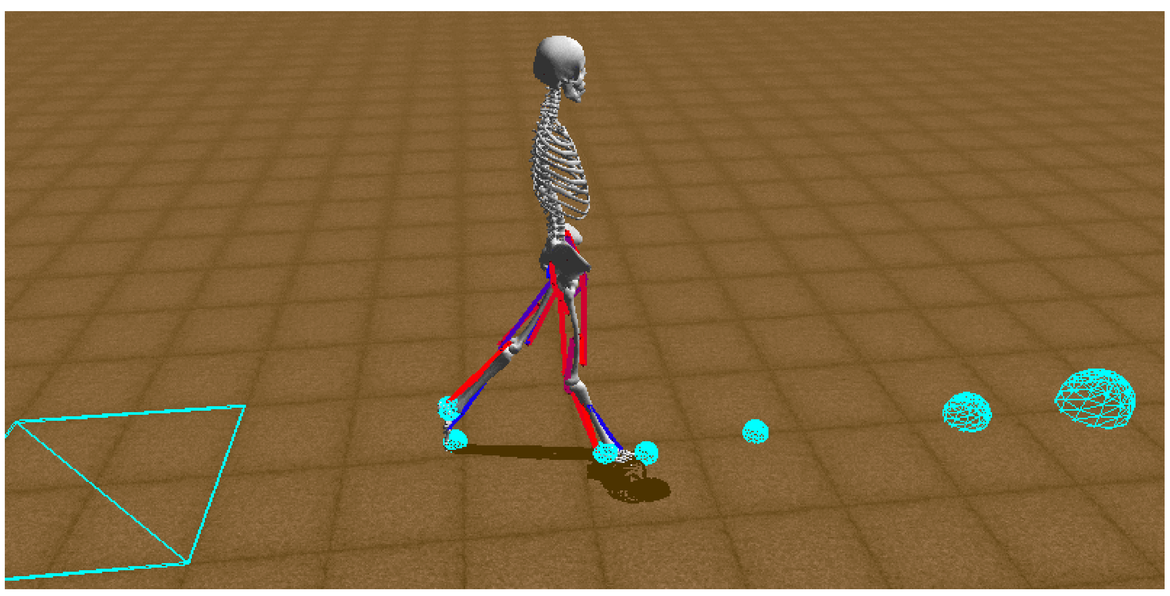
[1]Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw
[2]Institute of Fundamental Technological Research, Polish Academy of Sciences

Abstract

In the NIPS 2017 Learning to Run competition the goal was to build a controller for a human model to make it run as fast as possible through an obstacle course. Our team placed 22nd with a model trained with Proximal Policy Optimization (PPO) in 5 days on 80 CPU cores. We used reward shaping, extended observation vector and recompiled OpenSim simulator with lower accuracy to have 3x faster simulations.

Introduction

Over 450 teams tried to build a controller for a human model, **optimizing muscle activity such that the model travels as far as possible within 10 seconds**. It was a control problem with a continuous space of 41 inputs and 18 outputs. High computational cost of the simulations was part of the problem.



Our final model in action

Methods

We tried the following algorithms:

- Proximal Policy Optimization (**PPO**) [1],
- Deep Deterministic Policy Gradient (**DDPG**) [2],
- Evolution Strategies (**ES**) [3].

We also did:

Reward shaping. We guided learning to promising areas by shaping the reward function. We:

- employed a penalty for straight legs,
- employed a penalty for head behind pelvis,
- added 0.01 reward for every time step,
- used velocity instead of the passed distance.

Feature engineering. We changed all the positions from absolute to relative to the pelvis. We also extended the input to 82 values by adding:

- the remaining velocities and accelerations of the body parts,
- ground touch indicator for toes and tali,
- a queue of two obstacles: the next one (preserved from the original observation) and the previous one.

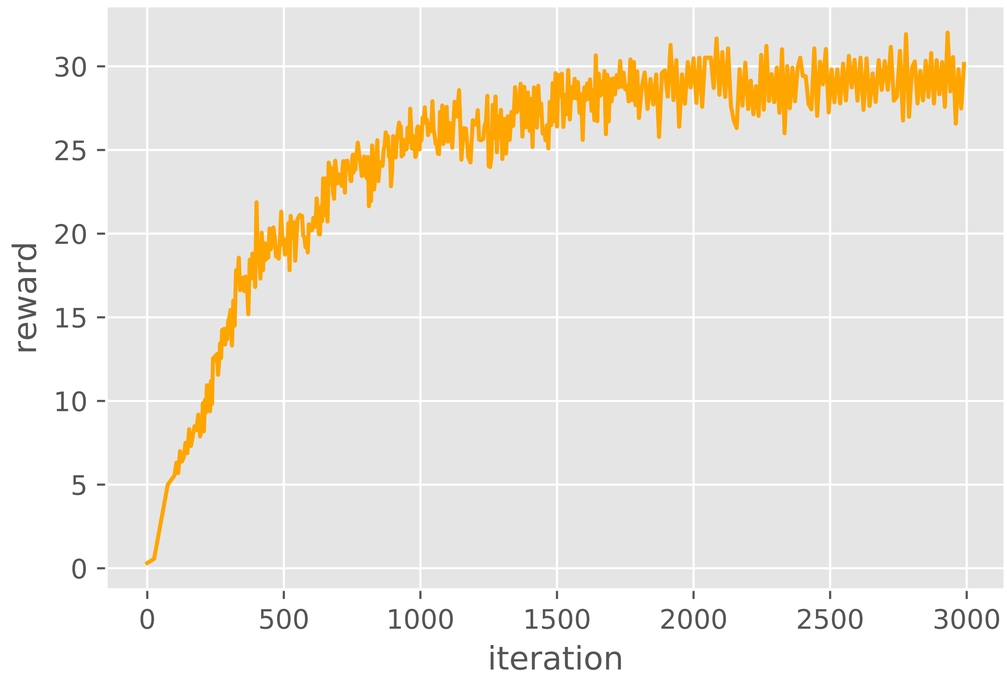
Experiments

We conducted our experiments on:

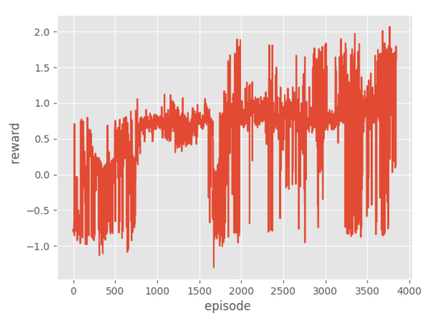
- an instance with 80 CPU cores,
- an instance with 16 CPU and 4 CUDA GPU cores,
- c5.9xlarge and c4.8xlarge Amazon's instances.

Results

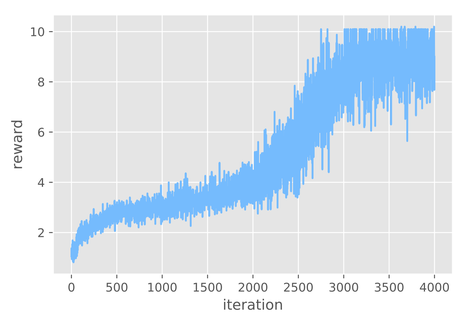
We obtained the best result using PPO, **our score in the final stage was 18.63 meters**.



PPO



DDPG



ES

The mean rewards achieved by PPO, DDPG and ES in one training run

Conclusions

We obtained the best result using PPO – the average score wasn't fluctuating as much and didn't suddenly drop. DDPG didn't perform well due to bad data normalization and not enough episodes. DDPG and ES usage in the competition environment should be more thoroughly examined. There are several things we would do differently now:

- try DDPG OpenAI baselines implementation,
- use a simpler environment in the beginning and reproduce the known results,
- make sure the normalization is done correctly, try Batch and Layer Normalization,
- tune hyperparameters in the end and in a rigorous way,
- repeat an action n times (so-called *frame skip*),
- learn also on the mirror flips of the observations,
- use TensorBoard instead of matplotlib.

[1] J. Schulman et al., *Proximal Policy Optimization Algorithms*, 2017
[2] T. Lillicrap et al., *Continuous control with deep reinforcement learning*, 2015
[3] T. Salimans et al., *Evolution Strategies as a Scalable Alternative to Reinforcement Learning*, 2017