Guest Post (Part II): Deep Reinforcement Learning with Neon

This is part 2 of a blog series on deep reinforcement learning. See part 1 "Demystifying Deep Reinforcement Learning" for an introduction to the topic.

The first time we read DeepMind's paper "Playing Atari with Deep Reinforcement Learning" in our research group, we immediately knew that we wanted to replicate this incredible result. It was the beginning of 2014, cuda-convnet2 was the top-performing convolutional network implementation and RMSProp was just one slide in Hinton's Coursera course. We struggled with debugging, learned a lot, but when DeepMind also published their code alongside their Nature paper "Human-level control through deep reinforcement learning", we started working on their code instead.

The deep learning ecosystem has evolved a lot since then. Supposedly a new deep learning toolkit was released once every 22 days in 2015. Amongst the popular ones are both the old-timers like Theano, Torch7 and Caffe, as well as the newcomers like Neon, Keras and TensorFlow. New algorithms are getting implemented within days of publishing.

At some point I realized that all the complicated parts that caused us headaches a year earlier are now readily implemented in most deep learning toolkits. And when Arcade Learning Environment – the system used for emulating Atari 2600 games – released a native Python API, the time was right for a new deep reinforcement learning implementation. Writing the main code took just a weekend, followed by weeks of debugging. But finally it worked! You can see the result here: https://github.com/tambetm/simple_dqn

I chose to base it on Neon, because it has

- the fastest convolutional kernels,
- all the required algorithms implemented (e.g. RMSProp),
- a reasonable Python API.

I tried to keep my code simple and easy to extend, while also keeping performance in mind. Currently the most notable restriction in Neon is that it only runs on the latest nVidia Maxwell GPUs, but that's about to change soon.

In the following I will describe:

1. how to install Simple-DQN
2. what you can do with Simple-DQN
3. how does Simple-DQN compare to others
4. and finally how to modify Simple-DQN.