

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

2015-12-29

#neon

Author Bio Image
Tambet Matiisen

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.

How To Install Simple-DQN?

There is not much to talk about here, just follow the instructions in the [README](#). Basically all you need are [Neon](#), [Arcade Learning Environment](#) and [simple_dqn](#) itself. For trying out pre-trained models you don’t even need a GPU, because they also run on CPU, albeit slowly.

What You Can Do With Simple-DQN?


Running a Pre-trained Model

Once you have everything installed, the first thing to try out is running pre-trained models. For example to run a pre-trained model for Breakout, type:

```
./play.sh snapshots/breakout_77.pkl
```

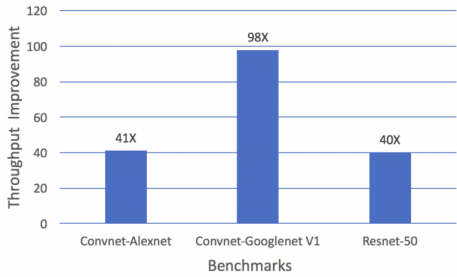
Or if you don’t have a (Maxwell) GPU available, then you can switch to CPU:

```
./play.sh snapshots/breakout_77.pkl --backend cpu
```

You should be seeing something like this, possibly even accompanied by annoying sounds 



Related Blog Posts

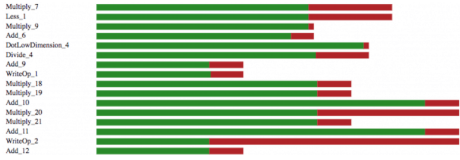


neon™ 2.0: Optimized for Intel® Architectures

neon™ is a deep learning framework created by Nervana Systems with industry leading performance on GPUs thanks to its custom assembly kernels and optimized algorithms. After Nervana joined Intel, we have been working together to bring superior performance to CPU platforms as well. Today, after the result of a great collaboration between the teams, we...

[Read more >](#)

#neon

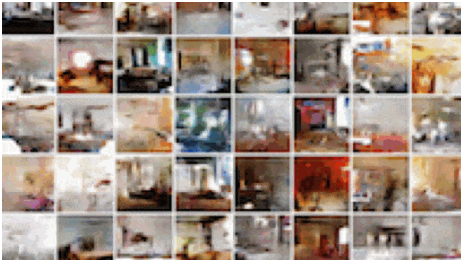


Intel® Nervana™ Graph Beta

We are building the Intel Nervana Graph project to be the LLVM for deep learning, and today we are excited to announce a beta release of our work we previously announced in a technical preview. We see the Intel Nervana Graph project as the beginning of an ecosystem of optimization passes, hardware backends and frontend...

[Read more >](#)

#Intel Nervana Graph #neon



Training Generative Adversarial Networks in Flexpoint

Training Generative Adversarial Networks in Flexpoint With the recent flood of breakthrough products using deep learning for image classification, speech recognition and text understanding, it's easy to think deep learning is just about supervised learning. But supervised learning requires labels, which most of the world's data does not have. Instead, unsupervised learning, extracting insights from...

[Read more >](#)

#neon

[Load More Posts](#)

Inspired by this Blog?

Keep tabs on all the latest news with our monthly newsletter.

SUBSCRIBE



[Terms of Use](#) | [Trademarks](#) | [Privacy](#) | [Cookies](#)