常用增强学习实验环境 II (ViZDoom, Roboschool, TensorFlow Agents, ELF, Coach等)

原创

ariesjzj 于 2017-11-12 11:15:51 发布 12554 收藏 35

文章标签： 强化学习 增强学习 深度强化学习 RL DRL

本文链接：https://blog.csdn.net/jinzhuojun/article/details/78508203

原文链接：http://blog.csdn.net/jinzhuojun/article/details/78508203

前段时间Nature上发表的升级版Alpha Go - AlphaGo Zero再一次成为热点话题。作为其核心技术之一的Deep reinforcement learning（深度增强学习，或深度强化学习）也再一次引发关注。Alpha Zero最有意义的地方之一是它去除了从人类经验（棋谱）中学习的过程，而是完全通过“左右互博”式的学习击败了自己的“前辈”。这也很能体现强化学习的特点，就是在弱监督信息下通过”Trial and error”来自我学习。

这两年DRL随着深度学习的大热也是火得不行。于是各种新的强化学习研究平台如雨后春笋冒出来，而且趋势也是从简单的toy场景慢慢扩展到3D迷宫，第一人称射击游戏，即时策略类游戏和复杂机器人控制场景等。之前曾写文介绍了一些流行的强化学习实验环境(常用强化学习实验环境 I (MuJoCo, OpenAI Gym, rllab, DeepMind Lab, TORCS, PySC2))。本文是第二弹。 ps: 真羡慕现在研究强化学习的孩子，各种五花八门的实验环境，算法参考实现也可以随便挑。。。

在第一篇中介绍过的本文就不重复累述了，这里把本文涉及的项目大致分为两类：

1. 实验场景类: 像OpenAI Gym，MuJoCo这些。

名称	github链接	类型	语言	平台	官方介绍
ViZDoom	代码	FPS	C++, Lua, Java, Python	Linux,Windows,Mac OS	官网 论文 教程
Roboschool	代码	Physical simulation	Python	Linux, Mac OS	博客
Multi-Agent Particle Environment	代码	Multi-agent	Python	Linux	论文 论文

2. 研究框架类: 一般会整合多个实验场景并提供方便统一的接口，方便其它场景的接入，并提供一些高级功能（如并行化），同时可能附带一些算法参考实现。

名称	github链接	场景	语言	实现算法	相关机构	官方介绍
TensorFlow Models	代码	OpenAI Gym, MuJoCo	Python	Actor Critic，TRPO，PCL，Unified PCL，Trust-PCL，PCL + Constraint Trust Region，REINFORCE，UREX	Community	N/A

名称	github链接	场景	语言	实现算法	相关机构	官方介绍
TensorFlow Agents	代码	OpenAI Gym	Python	BatchPPO	Google	论文
Universe/universe-starter-agent	代码1 代码2	Atari, CarPole，Flashgames，Browser task, etc.	Python	A3C	OpenAI	博客
ELF	代码	MiniRTS, Atari, Go	Python	PG, AC	Facebook	论文 教程
Coach	代码	OpenAI Gym, ViZDoom, Roboschool, GymExtensions, PyBullet	Python	DQN, DDQN, Dueling Q Network, MMC, PAL, Distributional DQN, Bootstrapped DQN, NEC Distributed: N-Step Q Learning,NAF, PG, A3C, DDPG, PPO, Clipped PPO, DFP 可参见算法总览图	Intel	文档 博客
Unity Machine Learning Agents	代码	3D Balance Ball, GridWorld	Python	PPO	Unity	文档 博客

下面介绍下它们的环境搭建。基础环境Ubuntu 16.04，Cuda 8.0(cuDNN 6.0)，TensorFlow 1.2.1，Anaconda 4.4.0, Python 2.7/3.5。

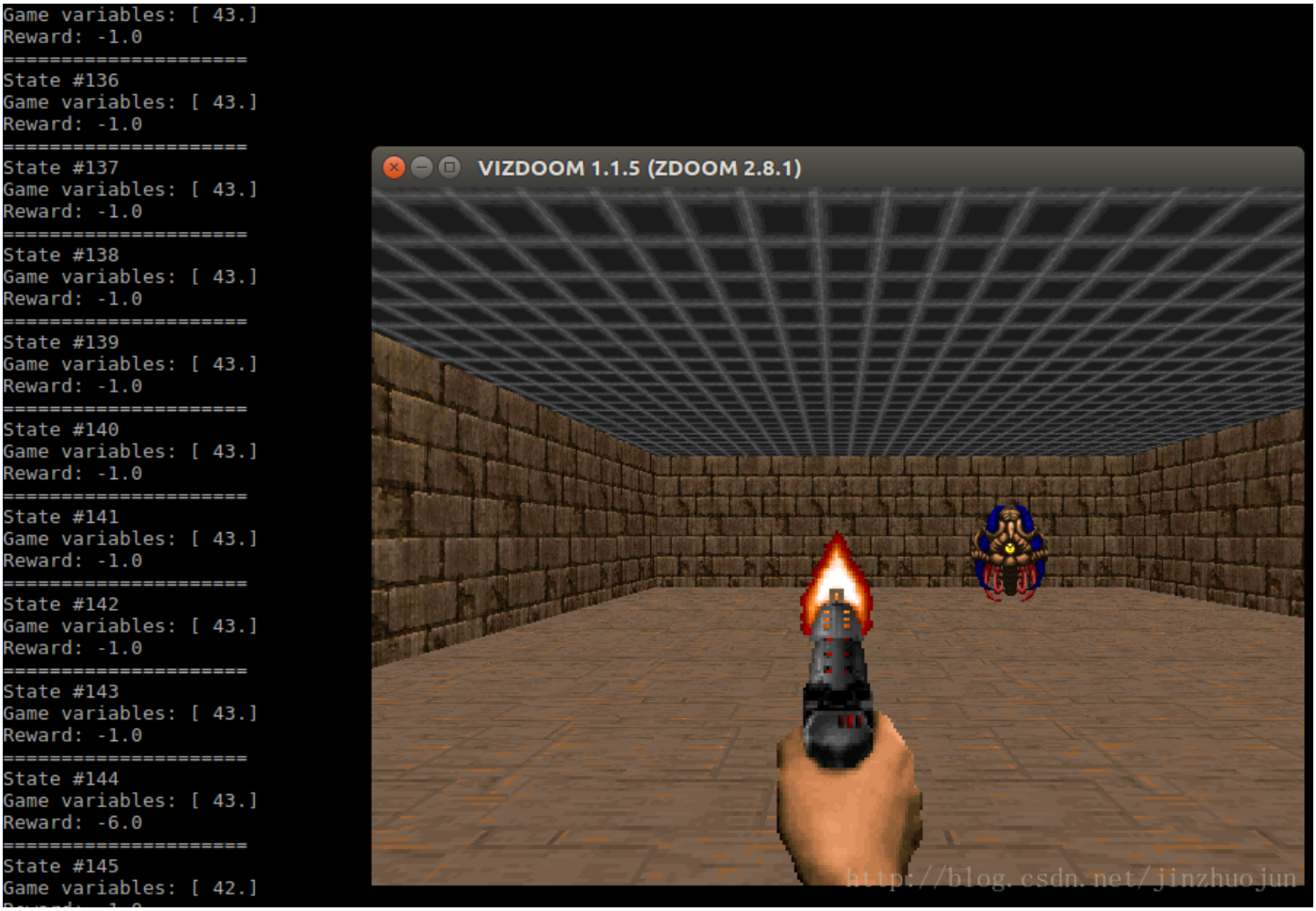
ViZDoom

提供了用AI玩毁灭战士游戏的环境。主要用于机器学习研究，尤其是DRL算法研究。环境搭建可以参见官方文档：https://github.com/mwydmuch/ViZDoom/blob/master/doc/Building.md

依赖中其它还好，比较麻烦的是需要Torch 7 。安装方法参考Getting started with Torch。

安装过程中会下载freedoom-0.11.3.zip，但我这网速渣下载非常慢，可以先从http://freedoom.github.io/download.html上下好放在项目的根目录 。如果你网速快请忽略。

安装完成后运行examples/python/basic.py，会起一个图形界面，其中智能体(agent)采取的动作是随机的。



它还用多种语言实现了DQN算法，位于examples目录下。

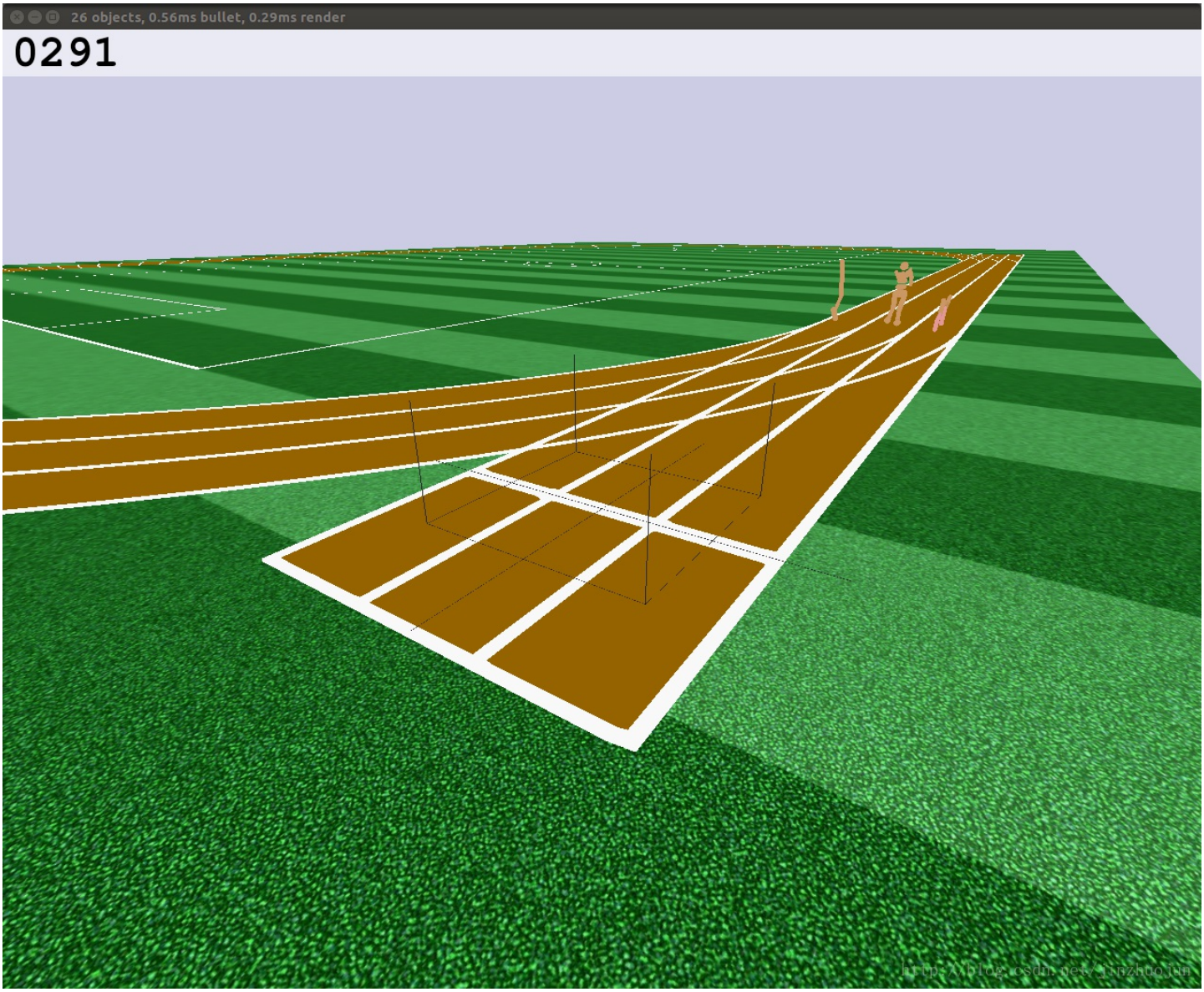
Roboschool

MuJoCo在许多的DRL论文中被用于经典的控制场景实验。但它有个缺点是要钱（除了30天免费版及学生版）。而这个缺点对于我们穷人来说可以说是致命的。作为MuJoCo实现的替代品，OpenAI开发了基于Bullet物理引擎的Roboschool。它提供了OpenAI Gym形式的接口用于模拟机器人控制。目前包含了12个环境。其中除了传统的类似MuJoCo的场景，还有交互控制，及多智能体控制场景。

安装方法比较简单，基本按github上readme即可。比如运行下面例子：

```
python $ROBOSCHOOL_PATH/agent_zoo/demo_race2.py
```

后的效果：



如果你不幸遇到下面的问题：

```
jzj@jlocal:~/source/roboschool$ python $ROBOSCHOOL_PATH/agent_zoo/RoboschoolHumanoidFlagrun_v0_2017may.
[2017-10-28 21:56:26,100] Making new env: RoboschoolHumanoidFlagrun-v1
QGLShaderProgram: could not create shader program
bool QGLShaderPrivate::create(): Could not create shader of type 2.
python3: render-simple.cpp:250: void SimpleRender::Context::initGL(): Assertion `r0' failed.
Aborted (core dumped)
```

根据https://github.com/openai/roboschool/issues/15中描述是一个已知bug，有个WR是在脚本前面加上from OpenGL import GLU。

Multi-Agent Particle Environment

多智能体粒子世界。主要用于多智能体场景下的DRL相关研究。项目不大，依赖也比较简单，基本只要OpenAI gym。安装方法：

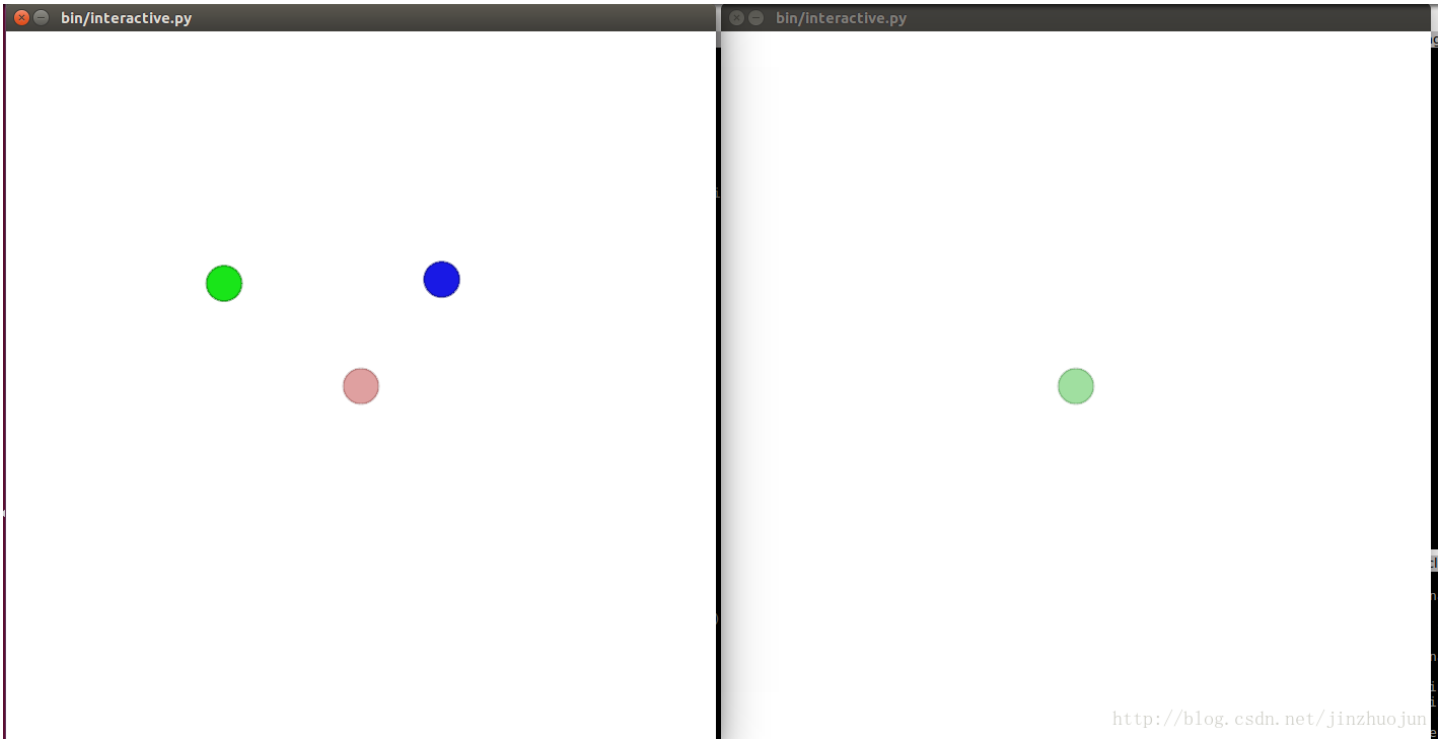
```
pip3 install -e . --user
```

然后就可以运行了:

```
bin/interactive.py --scenario <env>
```

其中<env>在其readme中有列。比如:

```
bin/interactive.py --scenario simple_push.py
```



TensorFlow Models

这是TensorFlow models中提供的强化学习算法集。环境搭建比较简单，如果已经装了OpenAI Gym和MuJoCo，基本装完了TensorFlow就可以跑。建议拿python 2.7跑，拿3.x要改些东西，比较麻烦。

装完后跑个readme中的例子试下：

```
python trainer.py --logtostderr --batch_size=400 --env=DuplicatedInput-v0 \
  --validation_frequency=25 --tau=0.1 --clip_norm=50 \
  --num_samples=10 --objective=urex
```

看到下面这样的输出就是在训练了：

```
INFO:tensorflow:at training step 0, model step 1: avg loss 0.945395, avg reward -0.240000, episode rewards: -0.280000
[2017-11-12 09:30:22,824] at training step 0, model step 1: avg loss 0.945395, avg reward -0.240000, episode rewards: -0.280000
INFO:tensorflow:at training step 25, model step 26: avg loss 2.251726, avg reward -0.236400, episode rewards: -0.243400
[2017-11-12 09:30:31,415] at training step 25, model step 26: avg loss 2.251726, avg reward -0.236400, episode rewards: -0.243400
INFO:tensorflow:at training step 50, model step 51: avg loss 1.927539, avg reward 0.354400, episode rewards: 0.404800
[2017-11-12 09:30:40,508] at training step 50, model step 51: avg loss 1.927539, avg reward 0.354400, episode rewards: 0.404800
INFO:tensorflow:at training step 75, model step 76: avg loss 0.936020, avg reward 10.514900, episode rewards: 10.589200
[2017-11-12 09:30:54,340] at training step 75, model step 76: avg loss 0.936020, avg reward 10.514900, episode rewards: 10.589200
INFO:tensorflow:at training step 100, model step 101: avg loss 0.075042, avg reward 15.326050, episode rewards: 15.342600
[2017-11-12 09:31:10,897] at training step 100, model step 101: avg loss 0.075042, avg reward 15.326050, episode rewards: 15.342600
INFO:tensorflow:at training step 125, model step 126: avg loss 0.014977, avg reward 15.311350, episode rewards: 15.328000
[2017-11-12 09:31:27,051] at training step 125, model step 126: avg loss 0.014977, avg reward 15.311350, episode rewards: 15.328000
INFO:tensorflow:at training step 150, model step 151: avg loss 0.030775, avg reward 15.319300, episode rewards: 15.308800
[2017-11-12 09:31:43,405] at training step 150, model step 151: avg loss 0.030775, avg reward 15.319300, episode rewards: 15.308800
INFO:tensorflow:at training step 175, model step 176: avg loss 0.028864, avg reward 15.334300, episode rewards: 15.300000
[2017-11-12 09:31:59,589] at training step 175, model step 176: avg loss 0.028864, avg reward 15.334300, episode rewards: 15.300000
INFO:tensorflow:at training step 200, model step 201: avg loss 0.020835, avg reward 15.311650, episode rewards: 15.324000
[2017-11-12 09:32:15,630] at training step 200, model step 201: avg loss 0.020835, avg reward 15.311650, episode rewards: 15.324000
INFO:tensorflow:at training step 225, model step 226: avg loss 0.007511, avg reward 15.328800, episode rewards: 15.315400
[2017-11-12 09:32:31,812] at training step 225, model step 226: avg loss 0.007511, avg reward 15.328800, episode rewards: 15.315400
INFO:tensorflow:at training step 250, model step 251: avg loss 0.012900, avg reward 15.304350, episode rewards: 15.308600
[2017-11-12 09:32:48,481] at training step 250, model step 251: avg loss 0.012900, avg reward 15.304350, episode rewards: 15.308600
INFO:tensorflow:at training step 275, model step 276: avg loss 0.005692, avg reward 15.349000, episode rewards: 15.368000
[2017-11-12 09:33:04,655] at training step 275, model step 276: avg loss 0.005692, avg reward 15.349000, episode rewards: 15.368000
INFO:tensorflow:at training step 300, model step 301: avg loss 0.000627, avg reward 15.358450, episode rewards: 15.384000
[2017-11-12 09:33:20,879] at training step 300, model step 301: avg loss 0.000627, avg reward 15.358450, episode rewards: 15.384000
INFO:tensorflow:at training step 325, model step 326: avg loss 0.003151, avg reward 15.322000, episode rewards: 15.372000
[2017-11-12 09:33:36,812] at training step 325, model step 326: avg loss 0.003151, avg reward 15.322000, episode rewards: 15.372000
INFO:tensorflow:at training step 350, model step 351: avg loss 0.001059, avg reward 15.339000, episode rewards: 15.404000
[2017-11-12 09:33:52,741] at training step 350, model step 351: avg loss 0.001059, avg reward 15.339000, episode rewards: 15.404000
INFO:tensorflow:at training step 375, model step 376: avg loss 0.003549, avg reward 15.331000, episode rewards: 15.316000
[2017-11-12 09:34:09,033] at training step 375, model step 376: avg loss 0.003549, avg reward 15.331000, episode rewards: 15.316000
INFO:tensorflow:at training step 400, model step 401: avg loss 0.012117, avg reward 15.336000, episode rewards: 15.308000
[2017-11-12 09:34:25,551] at training step 400, model step 401: avg loss 0.012117, avg reward 15.336000, episode rewards: 15.308000
INFO:tensorflow:at training step 425, model step 426: avg loss 0.087670, avg reward 15.315600, episode rewards: 15.323400
[2017-11-12 09:34:41,693] at training step 425, model step 426: avg loss 0.087670, avg reward 15.315600, episode rewards: 15.323400
```

TensorFlow Agents

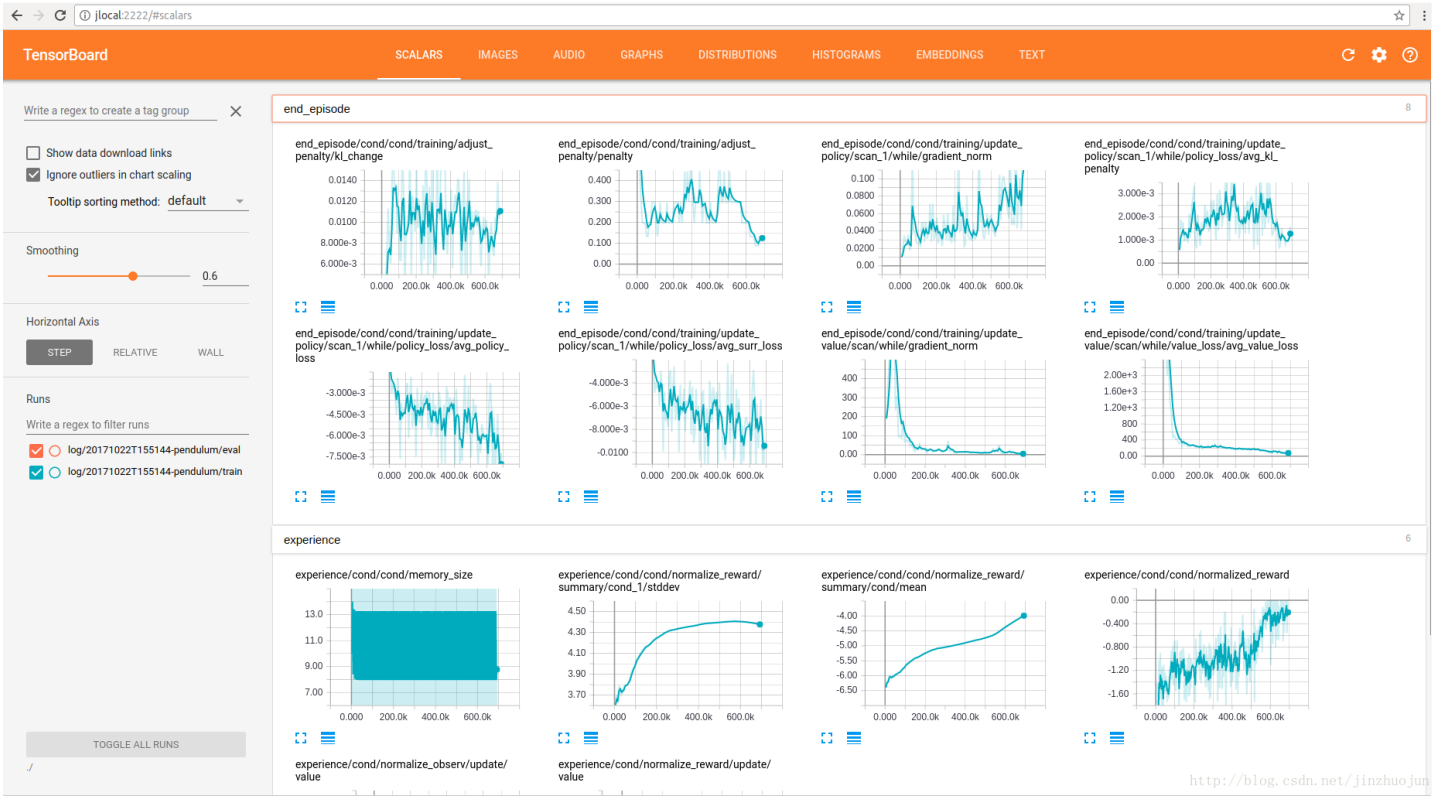
由Google两位研究员研发，用于在TensorFlow中构建并行强化学习算法。比较大的特点是容易开发并行强化学习算法。除了TensorFlow和OpenAI Gym，还需要安装ruamel.yaml：

```
pip install ruamel.yaml
```

按readme可以运行下例子：

```
$ python3 -m agents.scripts.train --logdir=./log --config=pendulum
$ tensorboard --logdir=./ --port=2222
```

利用TensorFlow的tensorboard可以看到训练过程数据的图形化显示：

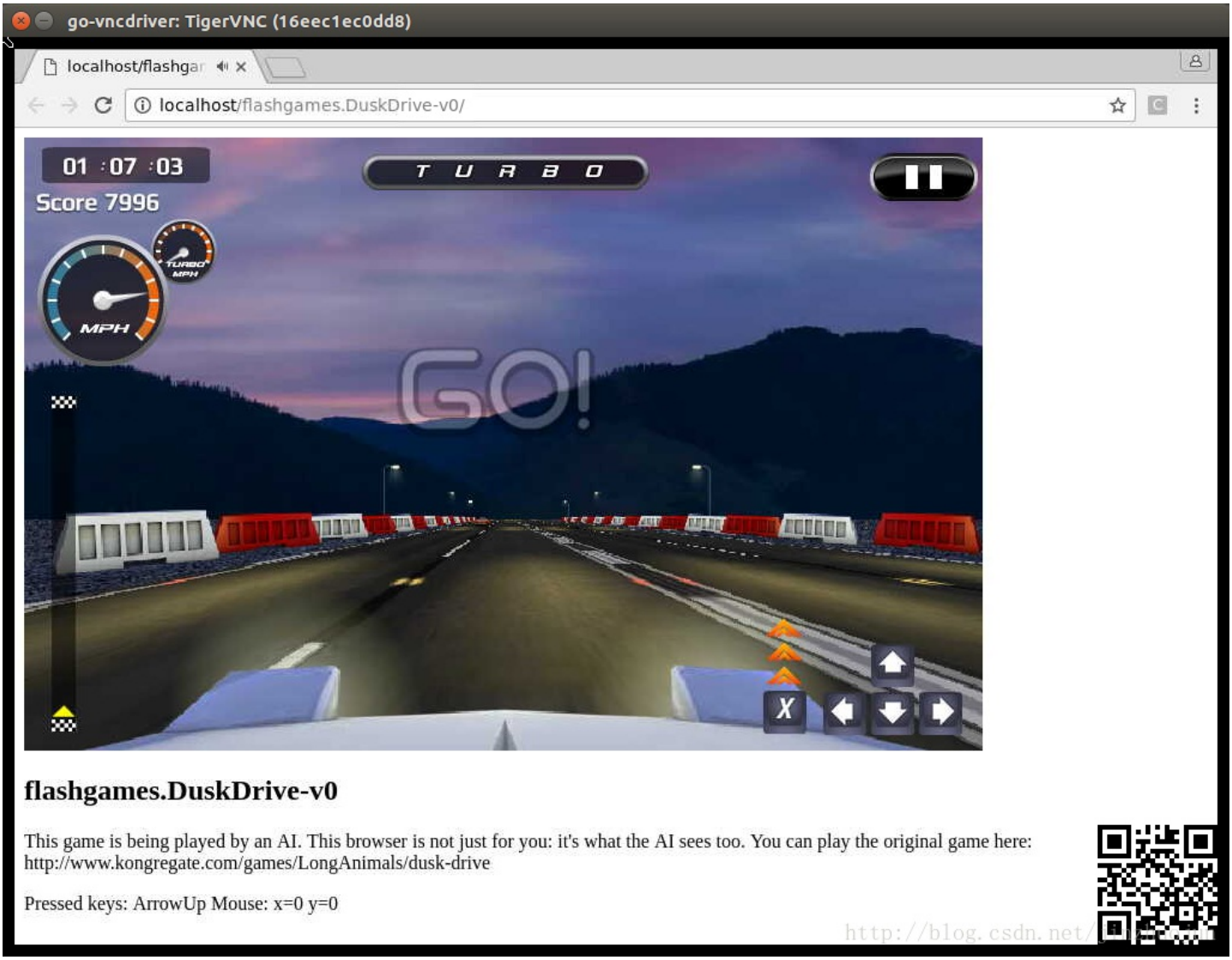


Universe

OpenAI出品，用于衡量和训练游戏中的AI的软件平台。特点之一是可以让一个现有的程序变为OpenAI Gym环境，而且不用改动程序的内部代码和API。它将程序封装成docker容器，然后以人类的使用接口（键盘鼠标，屏幕输出）提供给AI学习模块。目前已经包含了1000+环境 。

该框架可以运行在Linux和OSX上，支持Python 2.7和3.5。官方readme里说内部用的3.5，所以python 3.5的支持应该会好些。注意它会用到docker（比如那些游戏会run在container中，然后通过VNC显示），所以需要先安装docker。docker安装可参见：https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/

跑起readme中flashgames.DuskDrive-v0的例子后可以看到如下输出：



另外universe-starter-agent项目实现了一个用于universe环境中的agent，包含了A3C算法的实现。

ELF

其特点如其名字，可扩展(Extensive)，轻量(Lightweight)和灵活(Flexible)。它特点之一是可以让多个游戏实例并行执行。另外任何C++接口的游戏都可以通过wrapper接入到该框架中。目前支持MiniRTS（一个简化版即时策略游戏）及其扩展、Atari和围棋。引擎部分用C++部分实现，接口为python。

环境搭建只要运行readme中的install脚本即可。此外注意还需要用到tbb(Intel的并行编程库)：

```
sudo apt-get install libtbb-dev
```

比较特殊的是需要依赖PyTorch。可能由于我的网络非常之渣，用conda命令安装非常之慢：

```
conda install pytorch cuda80 -c soumith
```

建议你也可以选择在http://pytorch.org/上选择你的环境和想要的安装方式，比如：

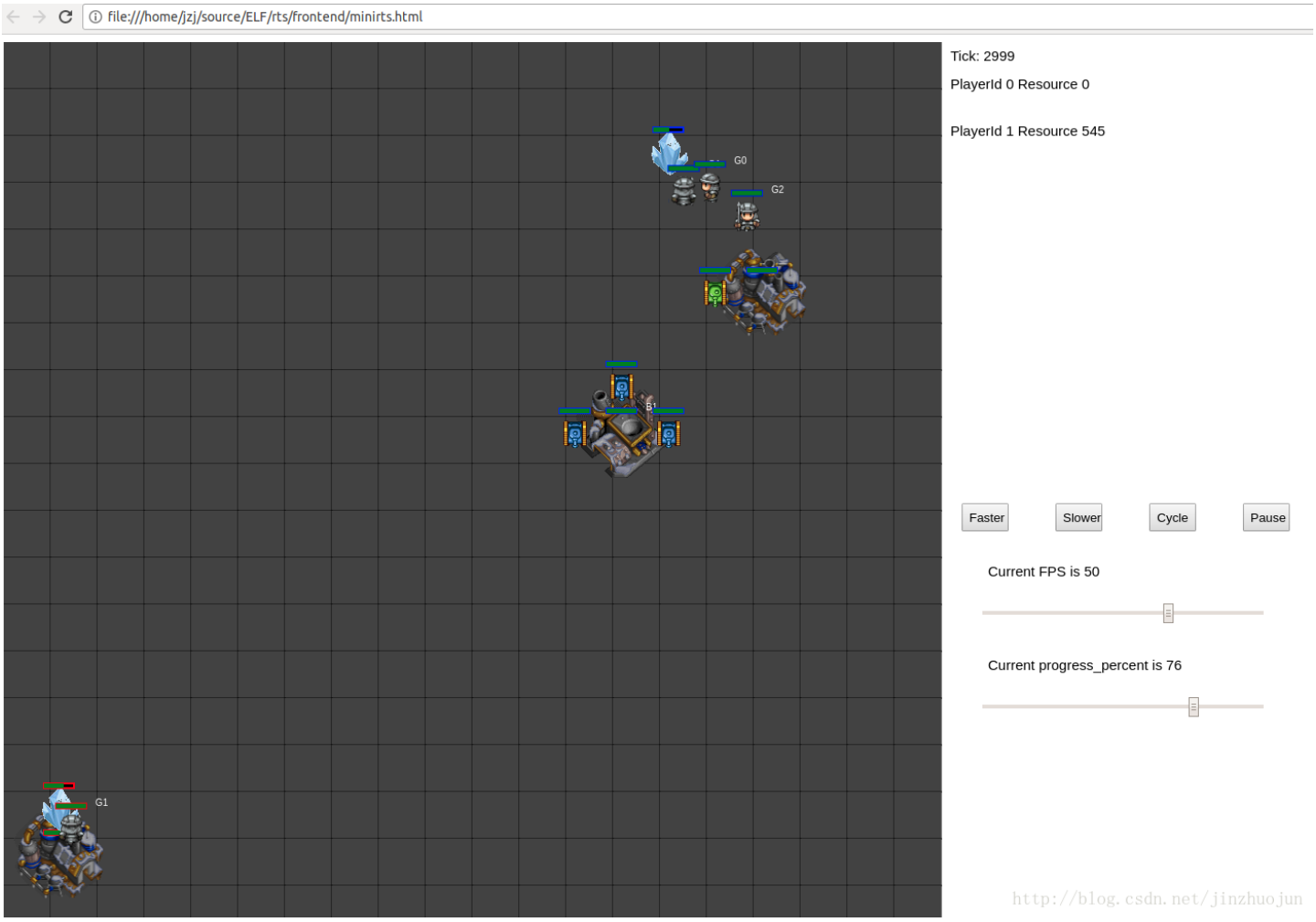
```
pip3 install http://download.pytorch.org/whl/cu80/torch-0.2.0.post3-cp35-cp35m-manylinux1_x86_64.whl
pip3 install torchvision
```

安装完后，根据readme，可以运行下几个例子试下：

```
sh ./train_minirts.sh --gpu 0   #  训练MiniRTS
```

训练过程很吃资源，其它事就别干了，换台电脑或者去喝杯咖啡吧。训练过程产生的模型文件会存为save-xxx.bin的形式。假设为save-2245.bin。然后可以用下面的脚本做模型的evaluation：

```
sh eval_minirts.sh ./save-2245.bin 20
```



如果要进行self-play（机器人自己和自己玩）可以执行下面的脚本：

```
sh ./selfplay_minirts.sh ./save-2245.bin
```

如果在运行eval或者self-play脚本时加上参数–save_replay_prefix replay，会生成一坨replayXXX-X.rep的重放文件。它们类似于以前玩的星际或者帝国中存档文件，可以用于重放。

下面是运行中可能碰到的问题及可以一试的解决方法：
Q: RuntimeError: cuda runtime error (10) : invalid device ordinal at torch/csrc/cuda/Module.cpp:87
A:这是因为selfplay_minirts.sh中默认整了多块GPU（–gpu 2）。我只有一块，改为–gpu 0即可。

Q: 在执行eval_minirts.sh时出错错误：
ValueError: Batch[actor1-5].copy_from: Reply[V] is not assigned
A: 在eval_minirts.sh中加参数–keys_in_reply V

Coach

由Intel收购的Nervana公司推出。该公司主要产品 是深度学习框架neon（不是arm上并行指令那个。。。）。Coach是一个基于Python语言的强化学习研究框架，并且包含了许多先进算法实现。该框架基于OpenAI Gym。其特色之一是可以方便地实现并行算法，充分利用CPU多核。另外有图形化工具可以方便地看训练过程中各项指标的曲线。它支持双后端，一个是当前大热的TensorFlow，一个是Intel自家的neon。该项目结合了Intel优化版本的TensorFlow和自家的神经网络加速库mkl-dnn，相信高性能是它的一个目标。

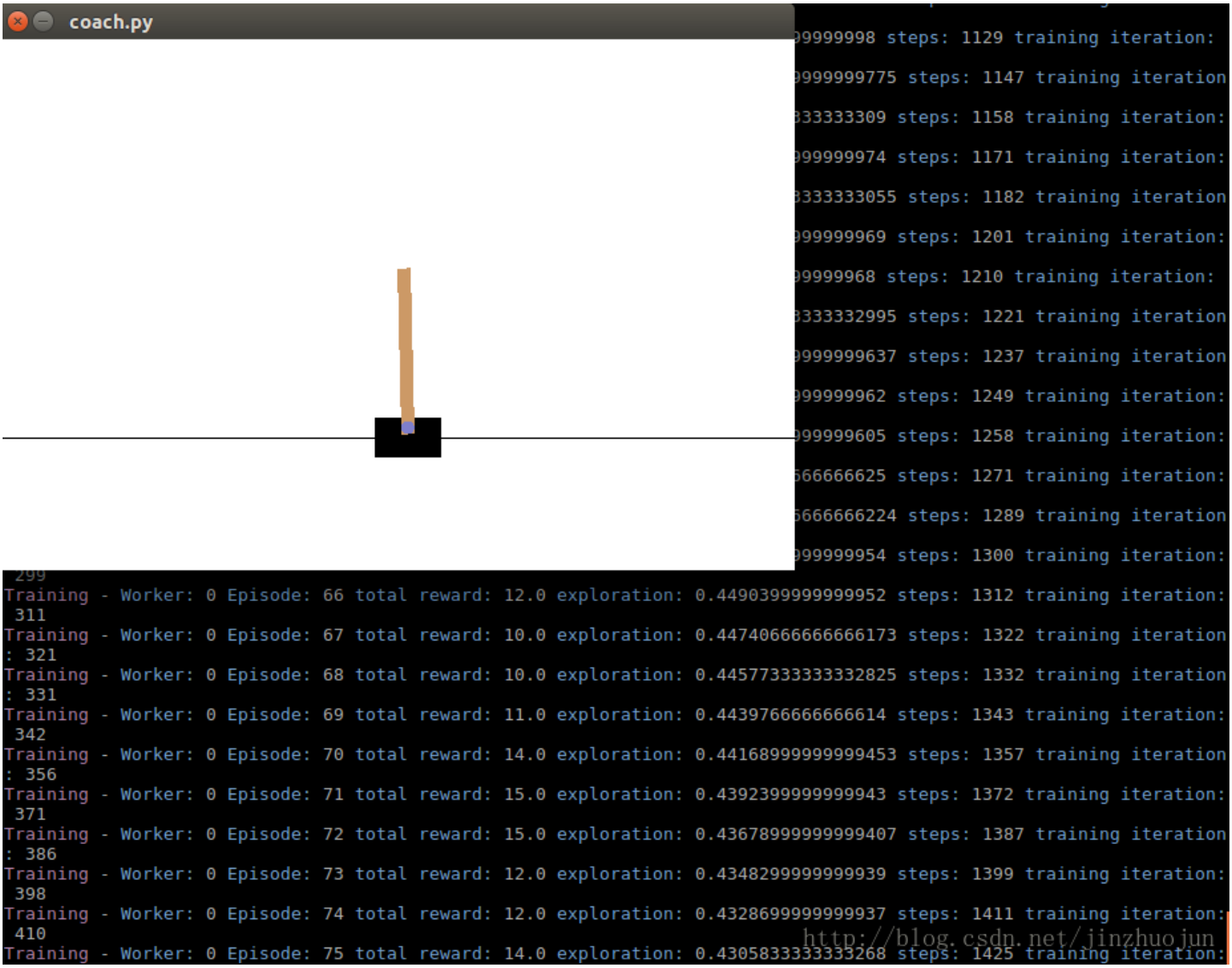
该框架要求Python 3.5。首先运行自带安装脚本：

```
./install.sh
```

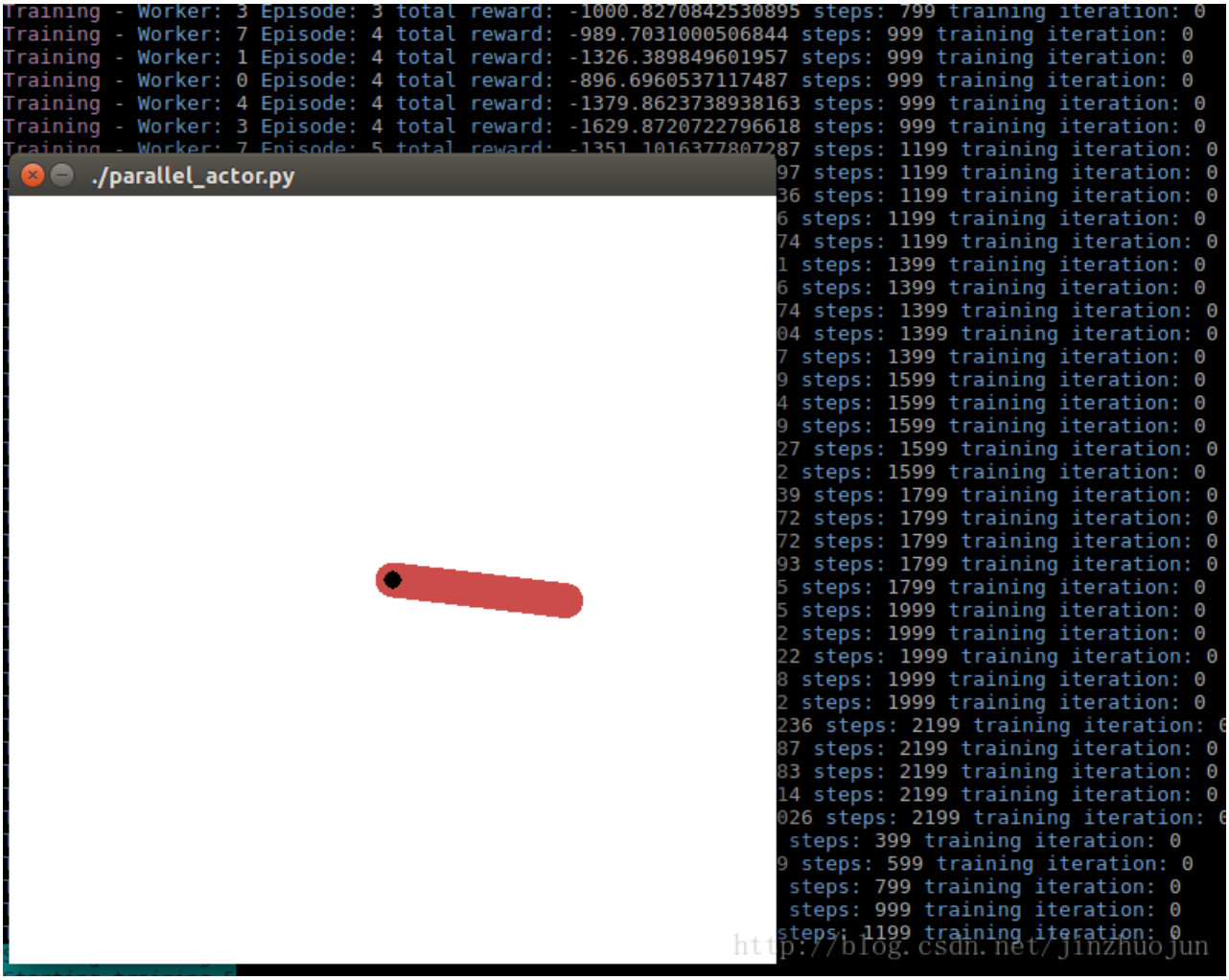
这个脚本会让输入一系列选项，其中如果选了要安装neon的话会去下mkl-dnn的库，我这下载巨慢。因为这是其中backend之一，不是必选的，因此如果网络和我一样渣的可以选择不装这个。

装完后可以运行几个例子：

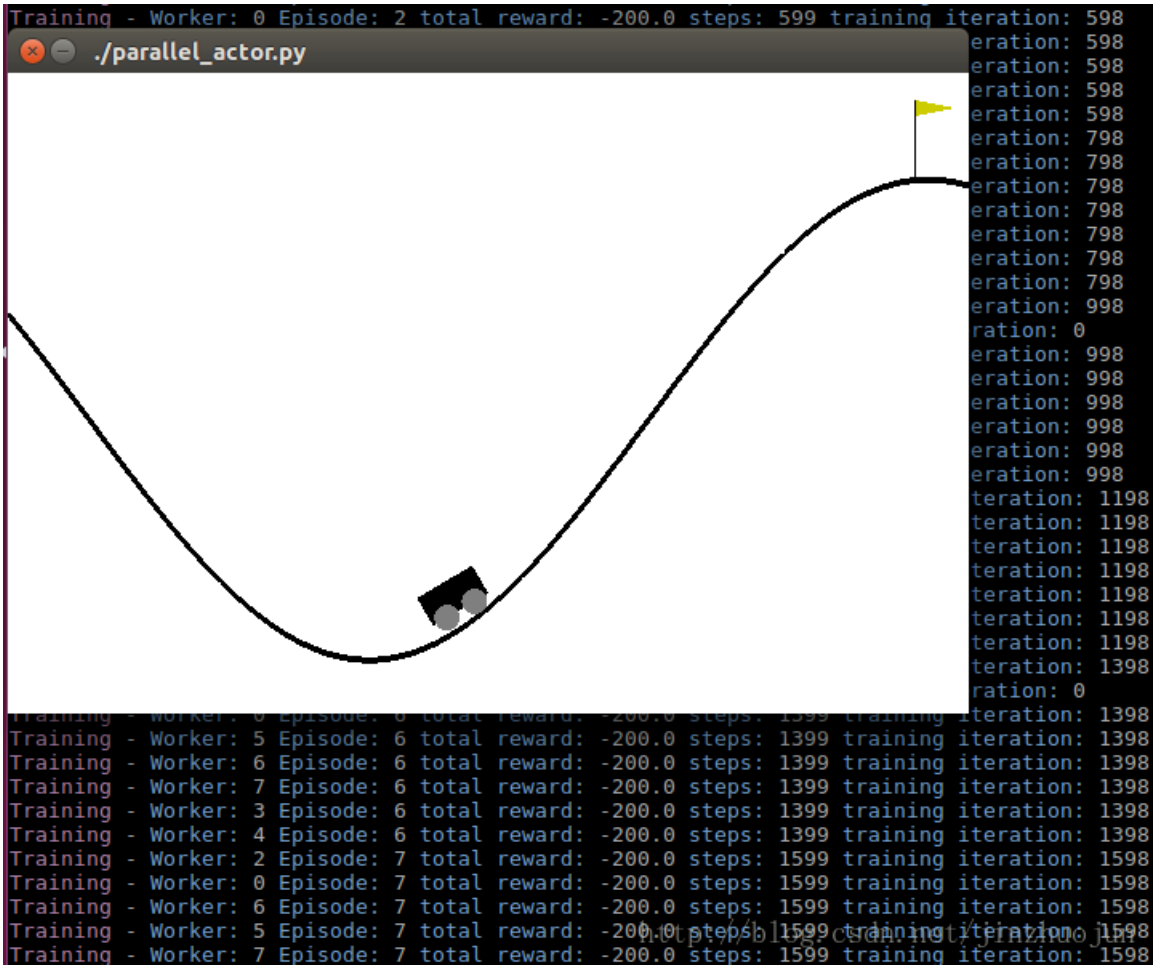
```
python3 coach.py -p CartPole_DQN -r
```




```
python3 coach.py -r -p Pendulum_ClippedPPO -n 8
```

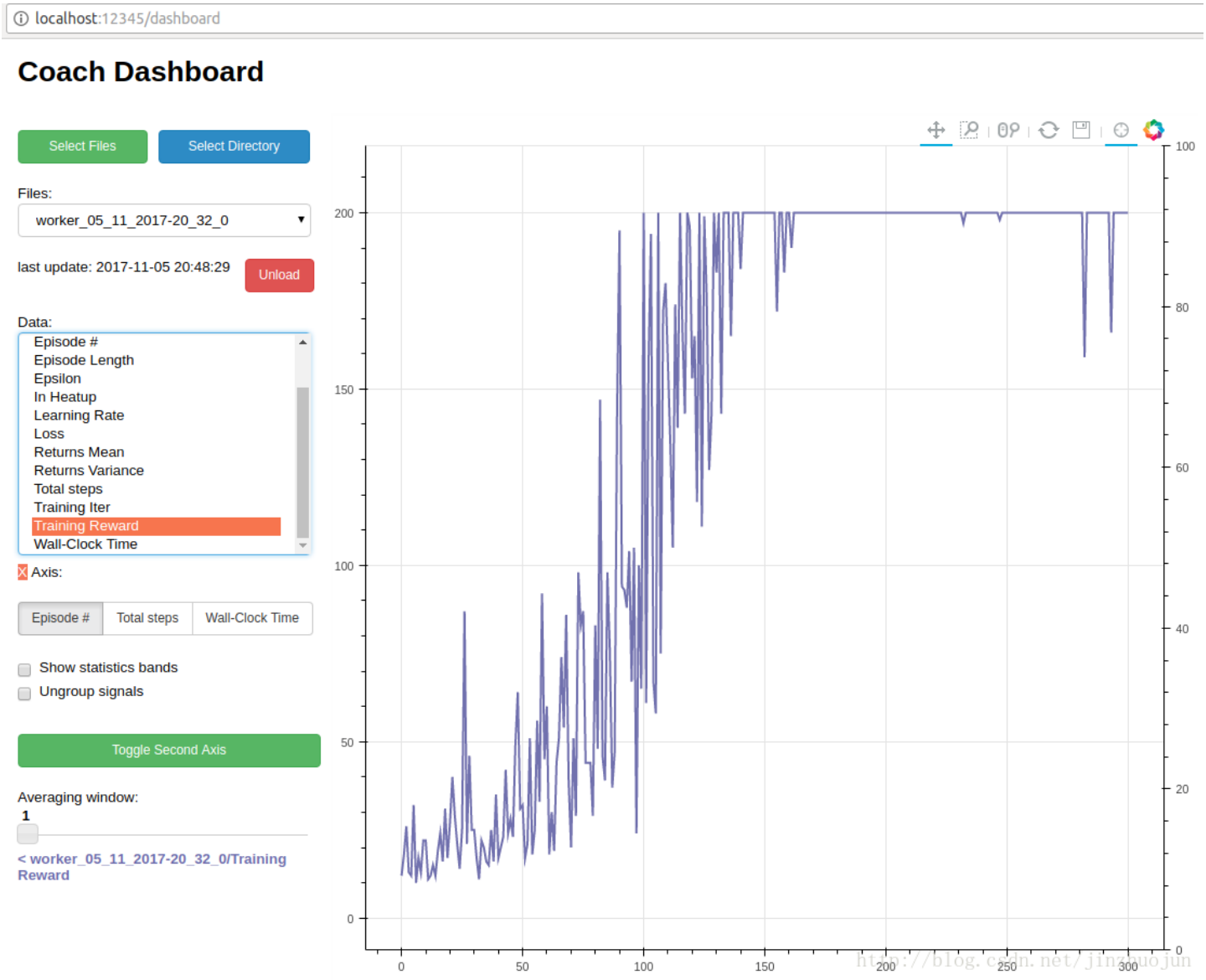



```
python3 coach.py -r -p MountainCar_A3C -n 8
```



然后可以通过下面的脚本将训练过程图形化：

```
python3 dashboard.py
```



如果过程中发现少annoy模块，可以用下面命令安装下。

Q:ModuleNotFoundError: No module named 'annoy'

A:pip3 install annoy

Unity Machine Learning Agents

大名鼎鼎的Unity公司出的。手机上的游戏很多就是基于Unity 3D引擎的。这次推出强化学习框架，它主打的是实验环境。亮点是可以结合Unity Editor来创建自定义的强化学习实验场景（详见https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Making-a-new-Unity-Environment.md）。可能也是看准了现在游戏中越来越多用到AI的趋势。它主要特点是可以支持多个观察输入，多智能体支持，并行环境计算等 。python 2和3都支持。目前支持的场景有3D Balance Ball，GridWorld和Tennis(https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Example-Environments.md)。算法部分就实现了PPO。因为主打是实验场景框架，算法意思一下就行。

因为我的工作环境基本都是Linux下的，而这个框架依赖的Unity SDK只支持Windows和Mac OS X。木有钱买水果，也实在打不起精神在Windows下搭环境，所以这个平台我没试过。大家有兴趣可以搭起来玩下。各种DRL的论文里都是Gym, MuJoCo，Torcs，DeepMind Lab这些，要是整些这个项目里的几个场景上去，或者自定义个场景，应该也挺让人新鲜的。