Practical Deep Neural Networks

GPU computing perspective

Python Platform for Scientific Computing

Yuhuang Hu Chu Kiong Loo

Advanced Robotic Lab
Department of Artificial Intelligence
Faculty of Computer Science & IT
University of Malaya

Outline

Outline

Why Ubuntu?

- ✗ There are more than thousands kinds of *nix distribution, and Ubuntu is only one of them.
- ✗ Ubuntu contains non-free software.
- ✓ Ubuntu is widely used by academic community in computing.
- ✓ Most of Deep Learning libraries explicitly support Ubuntu.
- ✓ Stable, fast, less issues.

Setting up Ubuntu: Tips

- ☆ Stick to stable release `xx.04.x LTS`, *e.g.* we are using `Ubuntu 14.04.2 LTS`.
- ☆ Update right after your fresh installation.
- ☆ **DO NOT** install graphical driver through `Ubuntu`'s package repository.
- ☆ **DO NOT** update anything anymore in the future if you don't have to.

Setting up Ubuntu: necessary software before start

Setup basic development environment

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get build-essential binutils
$ sudo apt-get install git cmake
$ sudo apt-get install openjdk-7-jdk
```

Outline

Anaconda

We use a specific distribution of `Python` — `Anaconda`, which is designed for large-scale data processing, predictive analytics, and scientific computing. You can download from here:

`http://continuum.io/downloads`

Since most of software are not compatible with `Python 3.*`, please download `Python 2.7`. And Install with

Setup Anaconda

```
$ bash Anaconda-2.2.0-Linux-x86.sh
```

Anaconda

- ✓ Easy install, easy remove (*delete `Anaconda` folder from home folder*)
- ✓ Awesome package management, no mess in system library.
- ✓ Simplified complex package installation.
- ✓ Complete computing libraries pre-installed, immediately ready-to-go.

Theano

- ✓ Theano has been pre-installed in `Anaconda`.
- ❒ Setup Theano configuration.
- ❒ Setup GPU environment for Theano. [Next Section].

Configure Theano

Before you start, you need to create a new file `.theanorc` under your home folder:

```
$ touch .theanorc
```

Assume you have only CPU available, you can edit this file as follows:

.theanorc

```
[global]
floatX = float32
device = cpu
```

You are ready to go! ☺

Outline

FYI

This configuration is only a suggest, choose whatever you feel comfortable.

Why Eclipse? And why PyDev?

- And IDE that can do a lot.
- Excellent hightlighter, language support, source code control, auto-completion.
- Easy to configure, easy to extend.

Install Eclipse

If you installed JDK 7 like previous, then you can download Eclipse Kepler:

`http://eclipse.org/kepler/`

If you installed JDK 8, you can use Eclipse Luna

`http://eclipse.org/luna/`

Download *Eclipse Standard 4.3.2* for Eclipse Kepler or *Eclipse IDE for Eclipse Committers 4.4.2* for Eclips Luna.
Unzip the entire `eclipse` folder to you home folder, the installation is then ready.

Setup PyDev

Eclipse is designed as a Java IDE, however, you can install new software for other languages like Python.

- Select help→Install New Software, type

  `http://pydev.org/updates`

  in *Work with:*, hit Enter, PyDev is retrieved.
- Select PyDev and leave PyDev Mylyn Integration as blank.
- Click Next and then follow the instruction. If there is an warning for unauthorized software, make sure you select all of them, they are not malware, just some routines for PyDev.
- Restart Eclipse, you should be able to create PyDev Project.

Hook up Anaconda

Assumed you installed PyDev successfully.

- ☞ Select Window→Preferences→PyDev→Interpreters→Python Interpreter
- ☞ Click New, type "Anaconda" in Interpreter Name and "`/path/to/anaconda/bin/python`" in Interpreter Executable (or use Browse to select your Anaconda's python executable). Click OK, then Eclipse will associate all installed packages in Anaconda.
- ☞ Click Apply and then Click OK to exit. You now can use Anaconda in Eclipse.

Create your first PyDev Project

Assume you installed PyDev and configured Anaconda

- Select File→PyDev Project→Next
- Type in project name and select Interpreter as "Anaconda".
- Click Finish, you project is created.
- You can now create your script or develop your package.

Outline

Heads Up

DO NOT use `Ubuntu`'s package repository to install NVIDIA GPU Driver and CUDA Toolkit, it's not state-of-the-art, and stable.
Procedures discussed here are experimental, no success guarantee.

Why NVIDIA?

- ✓ Widely used in industry and academic world.
- ✓ **All** popular Deep Learning libraries support CUDA, which is designed for NVIDIA graphics card only.
- ✓ Excellent performance and easy-to-use library interface.
- ✓ Industry standard.

Prerequisites

- A computer with a NVIDIA graphics card that supports CUDA technology.
- Enough disk space and at least 4 GB RAM. (Assume 64bit machine)
- Compute Capability of your card has to be at least 3.0 if you are intending to use `cuDNN` routines.
- A heart that is strong enough to take failure.

Install GPU driver

- Check out your GPU's info:

  ```
  $ lspci | grep "VGA"
  ```

  You should get similar messages like:

  ```
  00:02.0 VGA compatible controller: Cirrus
  Logic GD 5446
  00:03.0 VGA compatible controller: NVIDIA
  Corporation GK104GL [GRID K520] (rev a1)
  ```

- Go to NVIDIA website to download corresponding driver:
  `http://www.nvidia.com/Download/index.aspx?lang=en-us`

Install GPU driver

- Close `nouveau` [further experiment]
- Install GPU driver

  ```
  $ bash NVIDIA-Linux-xx.run
  ```

  Reboot after you installed the the driver, your GPU is now taking in charge
- You can check your installed driver by:

  ```
  $ nvidia-smi
  ```

  You should see a table that states the status of your GPU and driver version.

DO NOT update your system anymore since this moment, new kernel update may not be compatiable with your driver and then you will lost your GUI for sure. If this happens, you have to force boot older kernel version.

Install CUDA Toolkit

At this moment, CUDA Toolkit 7 is released, this guide is based on CUDA Toolkit 7.

- Download CUDA 7 from NVIDIA's developer website:
  `https://developer.nvidia.com/cuda-downloads`
- Install CUDA by:

  ```
  $ sudo bash cuda_xx_linux64.run
  ```

  DO NOT INSTALL GPU DRIVER INSIDE THE CUDA TOOLKIT. Remember to create symbolic link if the installer asked.

Install CUDA Toolkit

Add following contents in your bash configuration file `/.bashrc`.

.bashrc

```
# This configuration uses CUDA's symbolic link
export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
export CUDA_ROOT=/usr/local/cuda
```

Install CUDA Toolkit

And configure your `.theanorc` as

.theanorc

```
[global]
floatX = float32
device = gpu0

[nvcc]
fastmath = True

[cuda]
root = /usr/local/cuda
```

in order to enable GPU usage in your python scripts.

Before the end: cuDNN

The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. Popular libraries, such as Theano, Caffe and Torch 7 have added the support of cuDNN, the performance is improved significantly during implementing Convolutional Neural Networks (ConvNets).

- Request a copy of your cuDNN from NVIDIA:
  `https://developer.nvidia.com/cuDNN`
- Extract cuDNN:

  ```
  $ tar zxvf cudnn-xx.tgz
  ```

- Copy extracted files to CUDA folder:

  ```
  $ sudo cp cudnn.h /usr/local/cuda-x/include
  $ sudo cp libcudnn* /usr/local/cuda-x/lib64
  ```

  Now your cuDNN is installed correctly.

Before the end: how to select GPU for research

Select the right GPU can save you lots of money and boost your performance, you should read through this article before your purchase:
`https://timdettmers.wordpress.com/2014/08/14/which-gpu-for-deep-learning/`
From conclusion of the article:

- best GPU overall: GTX Titan X
- cost efficient but still expensive: GTX Titan X or GTX 980
- cheapest card with no troubles: GTX 960 4GB or GTX 680
- I work with data sets $>$ 250GB: GTX Titan
- I have no money: GTX 680 3GB
- I do Kaggle: GTX 980 or GTX 960 4GB
- I am a researcher: 1-4x GTX 980
- I am a researcher with data sets $>$ 250GB: 1-4x GTX Titan

Before the end: Amazon Web Services EC2 GPU Instance

- Two kinds of GPU instances are available
  - g2.2xlarge: GPU: 1, CPU: 8 cores, RAM: 15 GB, SSD Storage
  - g2.8xlarge: GPU: 4, CPU: 32 cores, RAM: 60 GB, SSD Storage
- We configured public images for deep learning usage:
  - DGYDLGPUv3 (ami-c5c2ee97) for g2.2xlarge
  - Image for g2.8xlarge is on the way
- Use Spot Request Instance to enjoy both speed and low cost.