Deep learning frameworks:

TensorFlow, Theano, Keras, Torch, Caffe

Vicky Kalogeiton, Stéphane Lathuilière, Pauline Luc, Thomas Lucas, Konstantin Shmelkov

Deep Learning Reading Group, 25 January 2017

Introduction

TensorFlow	Google Brain, 2015 (rewritten DistBelief)
Theano	University of Montréal, 2009
Keras	François Chollet, 2015 (now at Google)
Torch	Facebook AI Research, Twitter, Google DeepMind
Caffe	Berkeley Vision and Learning Center (BVLC), 2013

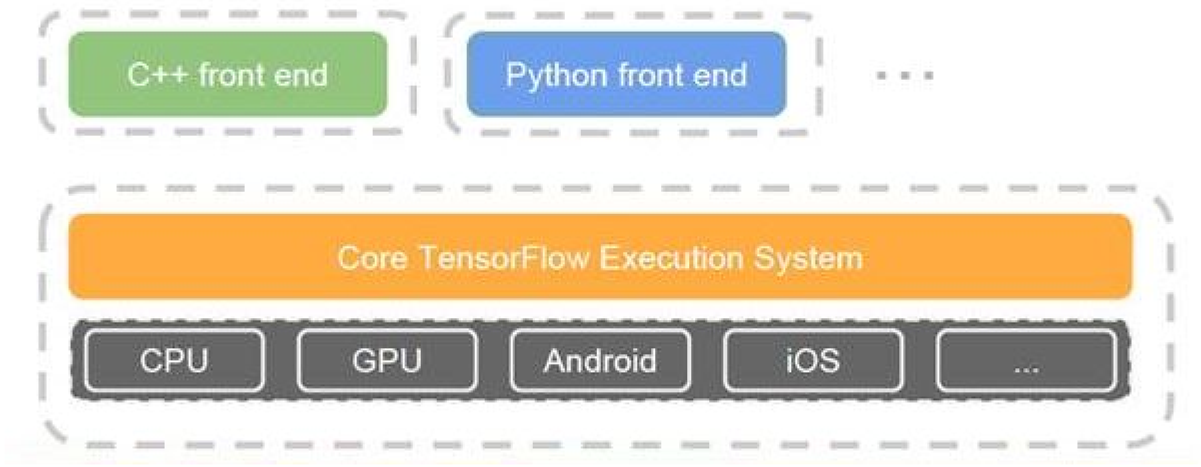
Outline

1. Introduction of each framework
    a. TensorFlow
    b. Theano
    c. Keras
    d. Torch
    e. Caffe
2. Further comparison
    a. Code + models
    b. Community and documentation
    c. Performance
    d. Model deployment
    e. Extra features
3. Which framework to choose when ..?

Introduction of each framework

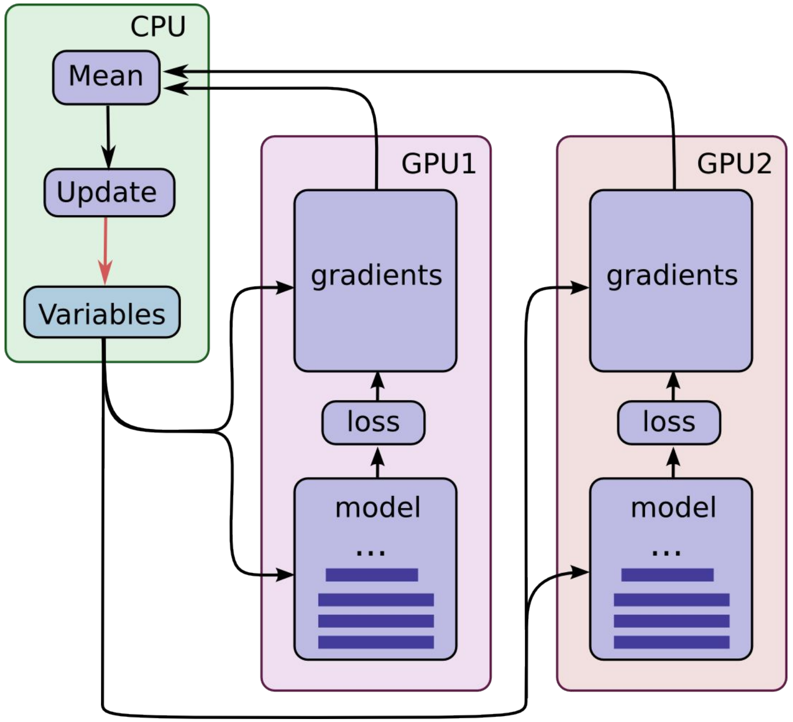
TensorFlow architecture

1) Low-level core (C++/CUDA)
2) Simple Python API to define the computational graph
3) High-level API (TF-Learn, TF-Slim, soon Keras...)

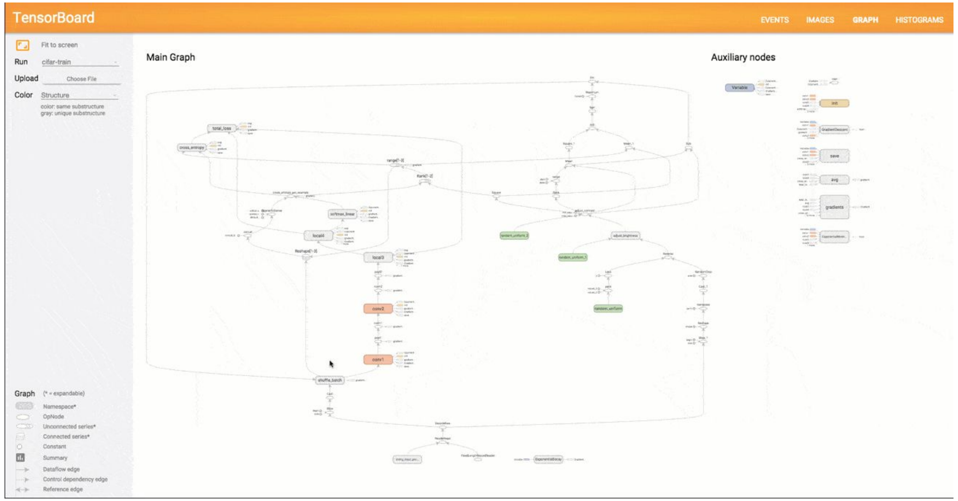


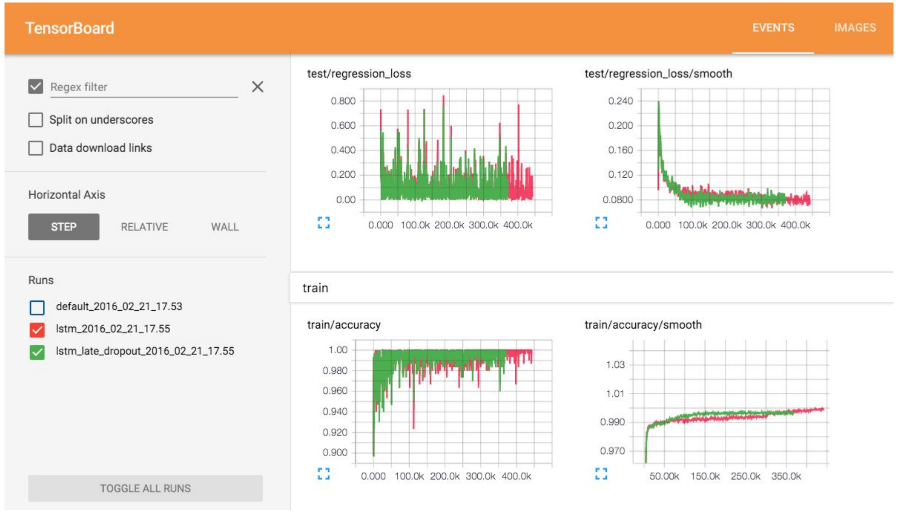
TensorFlow computational graph

- auto-differentiation!
- easy multi-GPU/multi-node
- native C++ multithreading
- device-efficient implementation for most ops
- whole pipeline in the graph: data loading, preprocessing, prefetching...



TensorBoard





- [✓] Regex filter
- [ ] Split on underscores
- [ ] Data download links

Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs
- [ ] default_2016_02_21_17.53
- [✓] lstm_2016_02_21_17.55
- [✓] lstm_late_dropout_2016_02_21_17.55

TensorFlow development

- + bleeding edge (GitHub yay!)
- + division in core and contrib => very quick merging of new hotness
- + a lot of new related API: CRF, BayesFlow, SparseTensor, audio IO, CTC, seq2seq
- + so it can easily handle images, videos, audio, text...
- + if you really need a new native op, you can load a dynamic lib
- - sometimes contrib stuff disappears or moves
- - recently introduced bells and whistles are barely documented

Presentation of Theano:

- Maintained by Montréal University group.
- Pioneered the use of a computational graph.
- General machine learning tool -> Use of Lasagne and Keras.
- Very popular in the research community, but not elsewhere. Falling behind.

What is it like to start using Theano?

- Read tutorials until you no longer can, then keep going.
- Once you are convinced that coding in pure Theano is cumbersome, pick up a Deep-learning library to go on top. (Lasagne/Keras).
- Make the Theano/Lasagne documentation your home page.

Theano’s flexibility.

- Automatic differentiation.
- Lasagne is very well conceived, saves a lot of code when trying new things without hurting flexibility.
- Most new ideas can be implemented quickly with simple modifications of existing “layers”.

Debugging in Theano: farewell to print debugging.

Main issues:

- Compile time of big models can be a huge pain.
- Error messages can be cryptic and pop up in the middle of nowhere.

Solutions: Be smart.

- Use reduced models (batch size of 1, fewer units per layer, fewer layers).
- Write modular code with defensive checks and unit test everything.
- Some debugging tools are provided.
- No prints.

Keras: strengths

- Easy-to-use Python library
- It wraps Theano and TensorFlow (it benefits from the advantages of both)
- Guiding principles: modularity, minimalism, extensibility, and Python-nativeness
- Why python? Easy to learn, powerful libraries (scikit-learn, matplotlib...)
- Many easy-to-use tools: real-time data augmentation, callbacks (Tensorboard visualization)
- Keras is gaining official Google support

Keras : simplicity

TF example:

```
kernel = tf.Variable(tf.truncated_normal([3, 3, 64, 64], type=tf.float32,stddev=1e-1), name='weights')
conv = tf.nn.conv2d(self.conv1_1, kernel, [1, 1, 1, 1], padding='SAME')
biases = tf.Variable(tf.constant(0.0, shape=[64], dtype=tf.float32), trainable=True, name='biases')
out = tf.nn.bias_add(conv, biases)
self.conv1_2 = tf.nn.relu(out, name='block1_conv2')
```

Keras:

```
x = Convolution2D(64, 3, 3, activation='relu', border_mode='same', name='block1_conv2')(x)
```

Keras: Weakness

- Less flexible
- No RBM for example
- Less projects available online than caffe
- Multi-GPU not 100% working

Torch - Framework Architecture

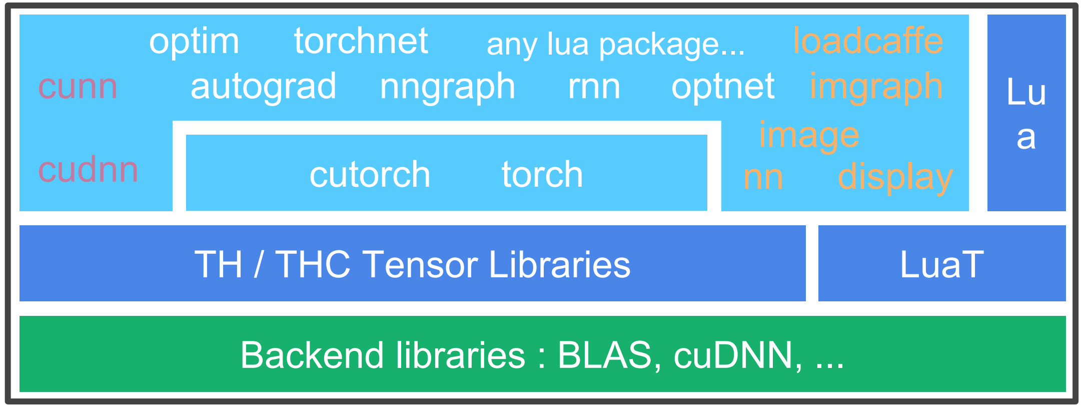
Mixed language :

- ➢ C / CUDA backend built on common backend libraries
- ➢ Lua frontend, running on LuaJIT

Why LUA ???

- Fast & embeddable
- Readable
- Very good interface to C

Architecture of the Torch framework. Figure inspired from torch presentation at OMLW 2014



Package installation uses luarocks.

Torch - Getting Started

Learning Lua

...in 15 minutes + gotchas

...if I'm a "book person"

...as I'm coding

Torch

Official presentation at OMLW 14

Official documentation

Pointers for Torch :

Torch Cheatsheet

Tutorials, official and unofficial packages, demos and code

Torch for Matlab or Numpy users

Model Zoo

Awesome-torch

Training on multi-GPUs over ImageNet

Distributed training with Torch

Torch - Main Strengths (1)

- Flexibility
  - Easy extensibility - at any level, thanks to easy integration with C
    - Result :
      - whatever the problem, there is a package.
      - new generic bricks are often very rapidly implemented by the community and are easy to pull
  - Imperative (vs declarative)
  - Typical use case : write a new layer, with GPU implementation :
    a. Implement for CPU nn
    b. Implement for GPU cunn
    c. Test (jacobian and unit testing framework)

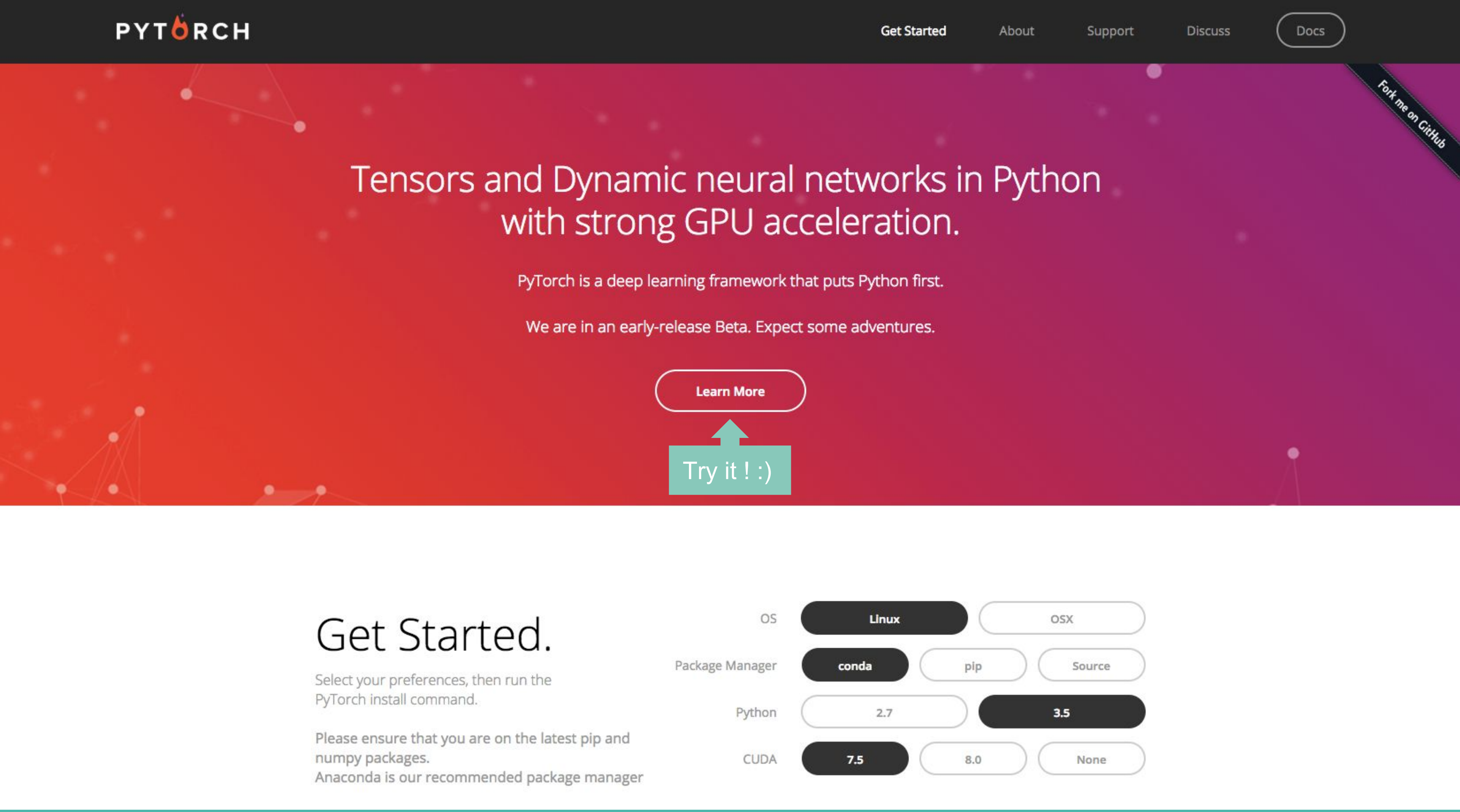
Torch - Main Strengths (2)

- Flexibility
- Readability
  - mid-level code - as well as high level - is in Lua
- Modularity
  - Easy to pull someone’s code
  - Use luarocks to install required packages
- Speed

➡ Very convenient for research.

Torch - Weaknesses

- Decent proportion of projects in Torch, but less than Caffe
- LuaJIT is not mainstream and does cause integration issues
- People hate Lua. But :
    - Easy to learn
    - If really, you cannot bring yourself to coding in Lua...



Tensors and Dynamic neural networks in Python with strong GPU acceleration.

PyTorch is a deep learning framework that puts Python first.

We are in an early-release Beta. Expect some adventures.

Try it ! :)

Get Started.

Select your preferences, then run the PyTorch install command.

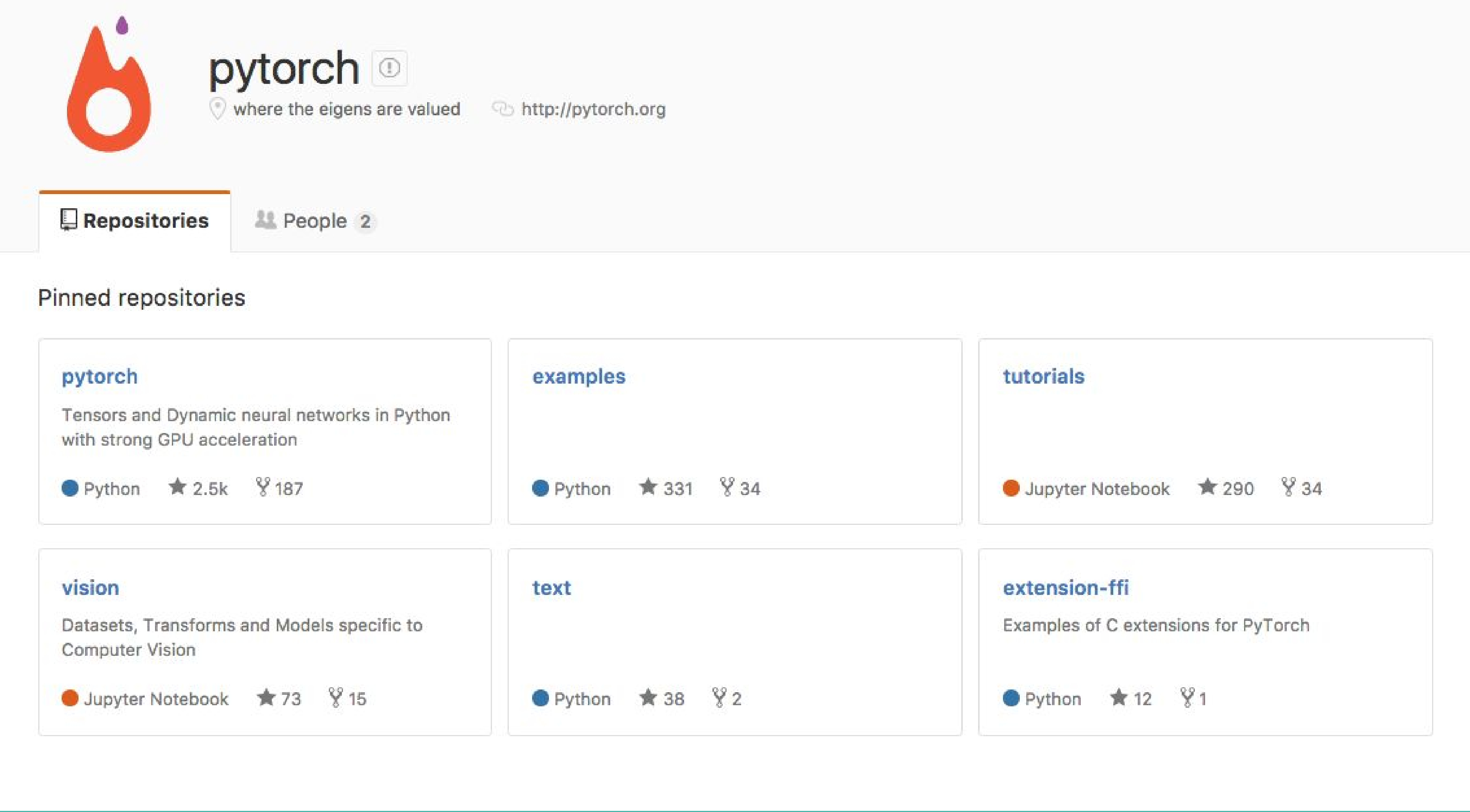
Please ensure that you are on the latest pip and numpy packages.
Anaconda is our recommended package manager

OS: [✓ Linux] OSX

Package Manager: [✓ conda] pip Source

Python: 2.7 [✓ 3.5]

CUDA: [✓ 7.5] 8.0 None



pytorch

where the eigens are valued http://pytorch.org

Repositories People 2

Pinned repositories

pytorch Tensors and Dynamic neural networks in Python with strong GPU acceleration ● Python ★ 2.5k 187	**examples** ● Python ★ 331 34	**tutorials** ● Jupyter Notebook ★ 290 34
vision Datasets, Transforms and Models specific to Computer Vision ● Jupyter Notebook ★ 73 15	**text** ● Python ★ 38 2	**extension-ffi** Examples of C extensions for PyTorch ● Python ★ 12 1

Caffe

- + Applications in machine learning, vision, speech and multimedia
- + Good for feed-forward networks and image processing
- + Excellent ConvNet implementation

- - Not intended for applications such as text, sound or time series data.

Caffe- Flexibility

+ very acceptable from the research community
+ easy to code with Caffe
+ easy to include different libraries
+ good Python and MATLAB interfaces
+ compatible to layers written in Python

- no auto-differentiation
- need of examples, implementations and source code to template your own code

Caffe- Interface

+ mainly: command line interface
+ supports also `pycaffe` interface

- model: defined in protobuf - using a text editor
- even if you use `pycaffe`

Caffe- Model examples

```
layer {
  name: "pool1"
  type: "Pooling"
  pooling_param {
    kernel_size: 2
    stride: 2
    pool: MAX
  }
  bottom: "conv1"
  top: "pool1"
}
```

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 0
    decay_mult: 0
  }
  convolution_param {
    num_output: 64
    kernel_size: 3
    pad: 1
   }
}
```

```
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "fc8"
  bottom: "label"
  top: "loss"
}
```

Caffe- Solver

- ➔ creation of the training network for learning and test network(s) for evaluation
- ➔ iterative optimization by calling forward / backward and parameter updating
- ➔ (periodical) evaluation of the test networks
- ➔ snapshotting of the model and solver state throughout the optimization

Caffe- Solver example

```
base_lr: 0.01          # begin training at a learning rate of 0.01 = 1e-2

lr_policy: "step"      # learning rate policy: drop the learning rate in "steps"
                       # by a factor of gamma every stepsize iterations

gamma: 0.1             # drop the learning rate by a factor of 10
                       # (i.e., multiply it by a factor of gamma = 0.1)

stepsize: 100000       # drop the learning rate every 100K iterations

max_iter: 350000       # train for 350K iterations total
```

Caffe- Architecture

When born: excellent, nowadays: average

+ layer as building block
+ Lots of layers already implemented

- Need to write C++/ CUDA code for new GPU layers
- Need to define the full forward, backward and gradient update for new layers
- Need to implement extra functions for both CPU/ GPU support (eg. Forward_gpu, Forward_cpu)

Caffe- Extra

- Not good for RNNs, mainly CNNs
- Cumbersome for big networks (eg., GoogLeNet, ResNet)

Caffe- Extra

+ first successful deep learning library
+ stable, efficient, ready for deployment
+ fastest library on CPU
+ easy to compile and install
+ easy to use pre-trained models
+ easy to fine-tune
+ GPU out-of-the-box training - even in Python
+ Out-of-the-box usage of common layer functions (eg. Conv, fc etc.)

Further comparison

Code + models

TensorFlow	Theano	Keras	Torch	Caffe
More code and models available (hype), including a lot provided by Google, some of open sourced code is already unusable due to very fast development, caffe models loaded in semi-manual way	- Lots of code to get started (tutorials and older models). - Loss of momentum and so recent models appear more slowly. - State of the art models can always be found, pretrained models can be slow to appear. - Community based: new code can be unstable.	Models from TensorFlow and Theano + converted model from caffe	State of the art classification models available in the ModelZoo. Significant proportion of research projects in torch. Loadcaffe convert caffe models to torch (often involves 0.5 to 2 days of work when non-sequential architecture or “non-standard layers”)	- Extensive code available online (even from the 1st year: >1K forked + significant changes since then) - heaven: state of the art pre-trained models available for a variety of domains - great for fine-tuning networks even without writing code

Community and Documentation

Community : (Github, groups, discussions...)

- Caffe has the largest community
- TensorFlow's is already large and growing.
- Keras' community is growing, while Theano's and Lasagne's are declining

Documentation

- Great documentation for Theano, Lasagne, Keras and Torch
- Most recent API is not documented for TensorFlow. Tutorials are often outdated.

Performance

TensorFlow	Theano	Keras	Torch	Caffe
Optimized for big models, can be memory hungry, but as fast as others (cuDNN).	- Run-time and memory competitive (efficient RNNs), - Compile time can be a huge pain (Strong -). - Multi-GPU support, not multi-machines.	Cf Theano and TensorFlow	All rely on cuDNN. Advantage : no compilation of models, which saves a lot of time during debugging. Memory : some layers are not very efficient because of inner buffers (fixed with OptNet or Pytorch.)	- Quite fast, eg. NVIDIA TitanX GPU: • Training: ~20 secs/20 iterations (5K images) • Testing: ~70 secs /validation set (50K images) - quick compilation - multi-GPU support but not with python layers - no distributed training

Model deployment

TensorFlow	Theano	Keras	Torch	Caffe
TF Serving for self-hosted web, Google Cloud Platform for easy web	Although compiled in C++/CUDA, can't be deployed without Python	Cf Theano and TensorFlow	Require LuaJIT to run models. (Can be problematic for integration more than performance)	+ C++ based + stable library - many forks + can be compiled in variety of devices

Extra features

TensorFlow	Theano	Keras	Torch	Caffe
+Written in C++ +Bindings for Python, C++, Java…	+Python	Python	+ Written in Lua and C/CUDA	Written in C++ Python and matlab interface
+Multi-GPU +Distributed +Windows, Android support	+Multi-GPU	+Multi-GPU +Android support	+ multi-GPU + distributed learning (torch-distlearn) + Stability : testing libraries	+ allows cross-platform (including windows) + very stable

Which framework to choose when...

Which framework to choose when ..?

1. You are a PhD student on DL itself?
2. You want to use DL only to get features?
3. You work in industry?
4. You started your 2 month internship?
5. You want to give practise works to your students?
6. You are curious about deep learning?
7. You don’t even know python?

Which framework to choose when ..?

1. You are a PhD student on DL itself: **TensorFlow, Theano, Torch**
2. You want to use DL only to get features: **Keras, Caffe**
3. You work in industry: **TensorFlow, Caffe**
4. You started your 2 month internship: **Keras, Caffe**
5. You want to give practise works to your students: **Keras, Caffe**
6. You are curious about deep learning: **Caffe**
7. You don’t even know python: **Keras, Torch**

Docker

Docker is a virtualization solution (similar to virtual machine). You can download container (or “image”) containing all the frameworks you need.

Why is is useful for DL?

- Installing all the DL frameworks takes time, so download a docker image instead.
- You are sure to have the same running environment on two different machines
- You cannot be root on the cluster.
- Don’t share the code only. Share your docker image also.

Thank you for your attention