Deep Learning With Caffe

Zhe Li
Zhuoning Yuan
Prof. Tianbao Yang

Department of
Computer Science

Agenda

- Introduction to Caffe
- Demo
  - ❑ Installation Caffe
  - ❑ Getting Start with Caffe

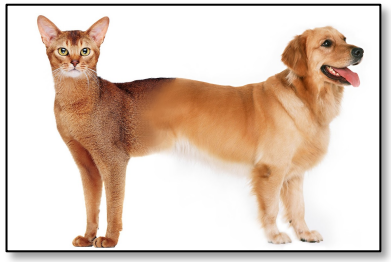
Claim

- hand-on Installation of Caffe on Argon Cluster
- Help You Get Start with Caffe
- Will Not be survey on Deep Learning

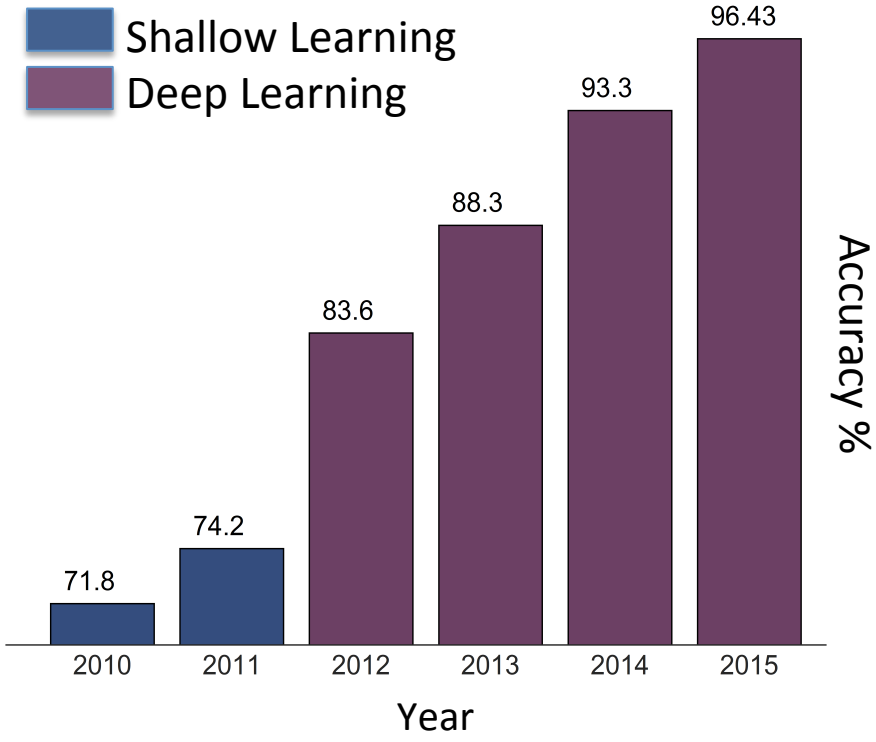
Deep Learning

Cat

Dog

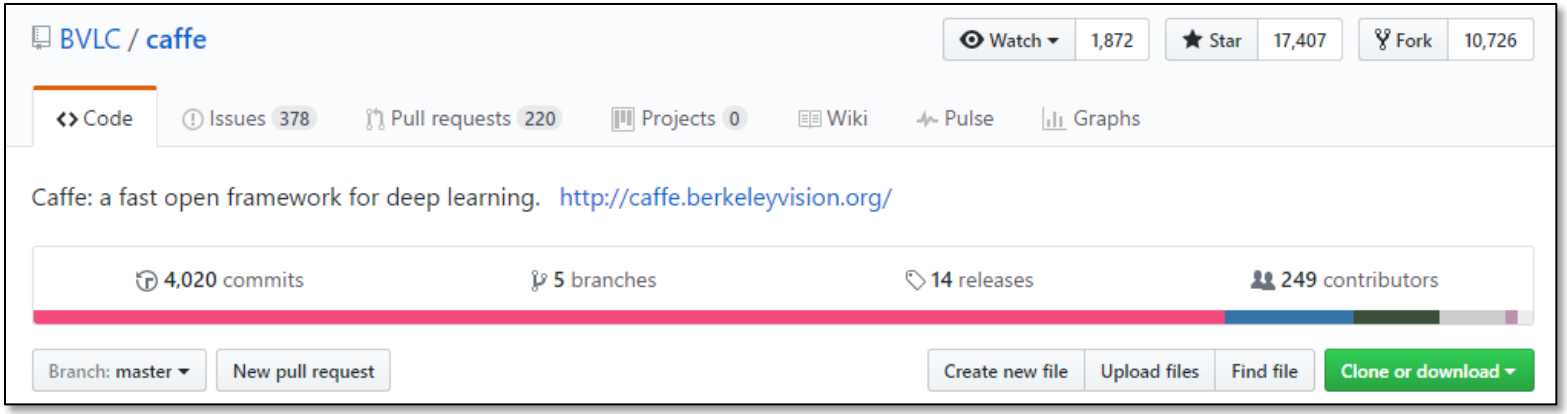


Dog or Cat?



What is Caffe?

- Open Deep Learning Framework (C++, CUDA)
- Well-trained Models
- Worked Example for Deep Learning
- Seamless Switch between GPU and CPU



Installation Requirements

System Library
BOOST
CUDA
OPENCV
OpenBLAS
HDF5
Python

VS

User Library
MDB
Snappy
LevelDB
Gflags
Glog
Protobuf

Dependencies Installation

- Create Folders for Installed Libraries

```
$mkdir .usr
$cd .usr
$mkdir local
$cd local
$mkdir bin include lib man share src
```

```
$mkdir caffe-dependency-package
$cd caffe-dependency-package
```

Dependencies Installation

- MDB (Database Library)

```
$git clone https://gitorious.org/mdb/mdb.git
$cd mdb/libraries/liblmdb
```

```
$vim Makefile
prefix  = /Users/[YourAccountName]/.usr/local
```

```
$make
$make install
```

Dependencies Installation

- Snappy Library (Compression/Decompression)

```
$git clone https://github.com/google/snappy.git
$cd snappy
$bash autogen.sh
$./configure --prefix=/Users/[YourAccountName]/.usr/local/
```

```
$vim Makefile
dist_doc_DATA = ChangeLog COPYING INSTALL NEWS README
format_description.txt framing_format.txt
```

```
$make
$make install
```

Dependencies Installation

- Leveldb (Database Library)

```
$git clone https://github.com/google/leveldb.git
$make
$mv out-shared/libleveldb.* /Users/[YourAccountName]/.usr/local/lib/
$mv out-static/libleveldb.* /Users/[YourAccountName]/.usr/local/lib/
```

Dependencies Installation

- Gflag (Logging & Command Lines Utility)

```
$git clone https://github.com/schuhschuh/gflags.git
$cd gflags
$mkdir build
$cd build
$cmake -DCMAKE_INSTALL_PREFIX=/Users/[YourAccountName]/.usr/
local/ ..
```

```
$vim CMakeCache.txt
...
CMAKE_CXX_FLAGS:STRING=-fPIC
BUILD_SHARED_LIBS:BOOL = ON
BUILD_STATIC_LIBS:BOOL = ON
...
```

```
$make
$make install
```

Dependencies Installation

- Glog (Logging & Command Lines Utility)

```
$git clone https://github.com/google/glog.git
$cd glog
$mkdir build
$cd build
```

```
$module load cmake/3.7.2
$cmake -DCMAKE_INSTALL_PREFIX=/Users/
[YourAccountName]/.usr/local/ -DBUILD_SHARED_LIBS=ON ..
```

```
$make
$make install
```

Dependencies Installation

- Protobuf (Define Data Structure Library)

```
$wget
https://github.com/google/protobuf/releases/download/v2.5.0/
protobuf-2.5.0.tar.gz
$tar -xzvf protobuf-2.5.0.tar.gz
$cd protobuf-2.5.0
$bash autogen.sh
$./configure --prefix=/Users/[YourAccountName]/.usr/local/
$make
$make install
```

Dependencies Installation

- OpenCV

```
$git clone https://github.com/Itseez/opencv.git
$mkdir build
$cd build
$cmake -DCMAKE_INSTALL_PREFIX=/Users/[YourAccountName]/.usr/
local/ ..
$make
$make install
```

System Library

- Load Modules

```
$mkdir modulefiles
$cd modulefiles
$vim common

module load python/2.7.13_parallel_studio-2017.1
module load cuda/8.0.44
module load matlab/R2016b
module load boost/1.63.0
module load OpenBLAS/0.2.19_gcc-5.4.0
module load hdf5/1.8.18
```

System Library

- Configure Bash File

```
$vim .bashrc

#add the following to .bashrc
module use -a /Users/[YourAccountName]/modulefiles
module load common
```

Set Environment Variable

```
$cd modulefiles
$mkdir depends_[YourAccountShortName]
$cd depends_[YourAccountShortName]
```

```
$vim 1.0
#add the following to the 1.0 file
set root /Users/[YourAccountName]/.usr/local/
prepend-path    PATH                $root/bin
prepend-path    CPLUS_INCLUDE_PATH      $root/include
prepend-path    C_INCLUDE_PATH          $root/include
prepend-path    LD_LIBRARY_PATH         $root/lib
prepend-path    LIBRARY_PATH            $root/lib
prepend-path    MANPATH                 $root/share
```

```
$vim .bash_profile
module load depends_[YourAccountShortName]/1.0
```


Time for
a BREAK

Caffe Installation

- Download Caffe

```
$git clone https://github.com/BVLC/caffe.git
$cd caffe
$cp Makefile.config.example Makefile.config
```

- Environment Variable Configuration

```
$vim Makefile.config

OPENCV_VERSION := 3
CUDA_DIR := /opt/apps/cuda/8.0.44
#-gencode arch=compute_20,code=sm_20 \
#-gencode arch=compute_20,code=sm_21 \
BLAS := open
BLAS_INCLUDE := /opt/apps/OpenBLAS/0.2.19_gcc-5.4.0/include
BLAS_LIB := /opt/apps/OpenBLAS/0.2.19_gcc-5.4.0/lib
```

Caffe Installation

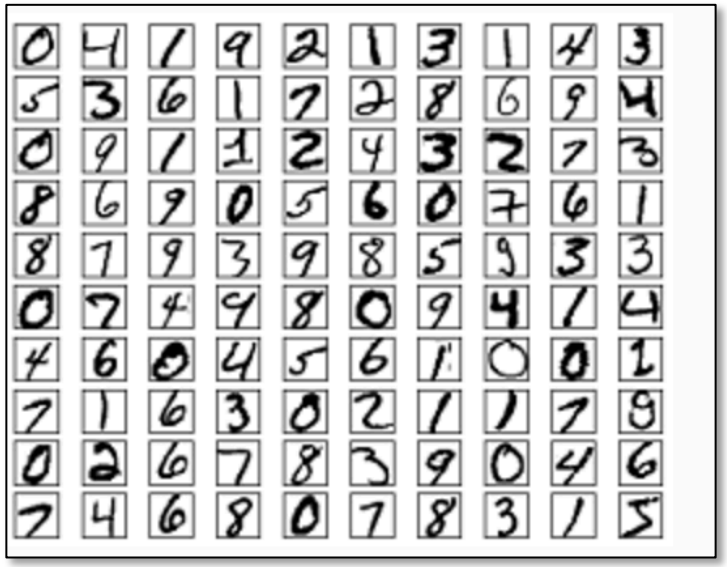
```
...

MATLAB_DIR := /opt/apps/matlab/R2016b

PYTHON_INCLUDE := /usr/include/python2.7\
                  /opt/apps/python/2.7.13_parallel_studio-2017.1/lib/python2.7/site-
packages/numpy-1.11.2-py2.7-linux-x86_64.egg/numpy/core/include

PYTHON_LIB := /usr/lib

INCLUDE_DIRS := $(PYTHON_INCLUDE) /Users/[YourAccountName]/.usr/local/
include

LIBRARY_DIRS := $(PYTHON_LIB) /Users/[YourAccountName]/.usr/local/lib /opt/
apps/boost/1.63.0/lib/ /opt/apps/hdf5/1.8.18/lib/ /usr/lib64
```

Compile Caffe

```
$make all
$make test
$make runtest
$make distribute
$make pycaffe
```

Get start with Caffe

- Prepare Data - MINST Handwriting Dataset
  - ❑ Each image with size 28*2
  - ❑ 50,000 training image and 10,000 test images



Get start with Caffe

- Prepare Data - MINST Handwriting Dataset

```
$cd caffe
$bash data/mnist/get_mnist.sh
```

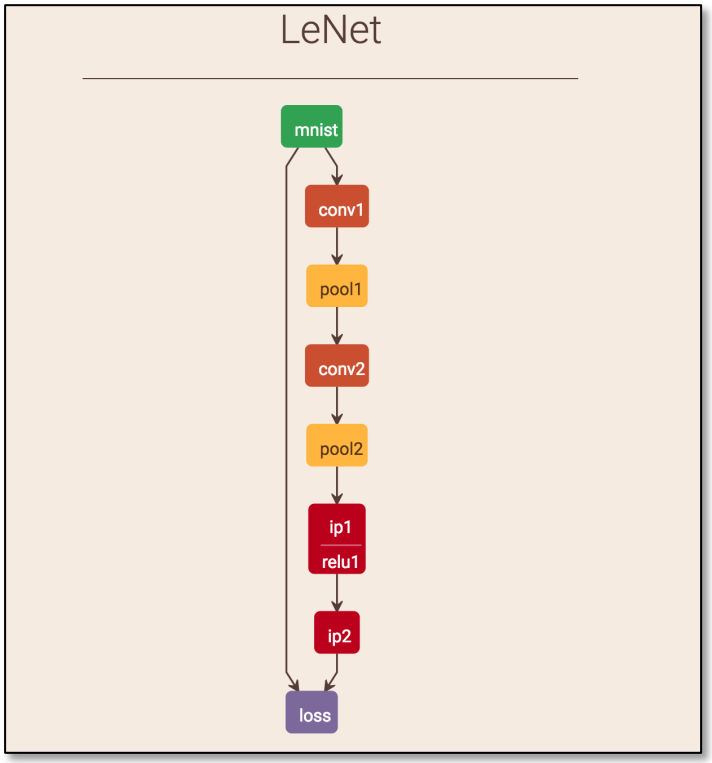
```
[zli79@argon-mm-p100-compute-1-37 caffe]$ ls data/mnist/
get_mnist.sh            t10k-labels-idx1-ubyte   train-labels-idx1-ubyte
t10k-images-idx3-ubyte  train-images-idx3-ubyte
```

```
$bash examples/mnist/create_mnist.sh
```

```
[zli79@argon-mm-p100-compute-1-37 caffe]$ ls examples/mnist/
convert_mnist_data.cpp              mnist_autoencoder_solver.prototxt
create_mnist.sh                     mnist_autoencoder_solver_adadelta.prototxt
lenet.prototxt                      mnist_autoencoder_solver_adagrad.prototxt
lenet_adadelta_solver.prototxt      mnist_autoencoder_solver_nesterov.prototxt
lenet_auto_solver.prototxt          mnist_test_lmdb
lenet_consolidated_solver.prototxt  mnist_train_lmdb
```

Get start with Caffe

- Define a Network Structure



- 1 Data Layer
- 2 Convolution Layers
- 2 Pooling Layers
- 2 Fully Connected Layers
- 1 Loss Layer

Get start with Caffe

- Define Network Structure
  - Network structure defined by user
  - Different problems may have different network structure
  - Parameters in network structure is from fine-tuning

Get start with Caffe

- Define Network Structure

```
name: "LeNet"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    scale: 0.00390625
  }
  data_param {
    source: "examples/mnist/mnist_train_lmdb"
    batch_size: 64
    backend: LMDB
  }
}
```

Figure 1. lenet_train_test.prototxt

Get start with Caffe

- Define Solver
  - ❑ Choose algorithm to update model
  - ❑ Learning rate strategy
  - ❑ Where to store model
  - ❑ ……

Get start with Caffe

- Define Solver

```
# The train/test net protocol buffer definition
net: "examples/mnist/lenet_train_test.prototxt"
# test_iter specifies how many forward passes the test should carry out.
# In the case of MNIST, we have test batch size 100 and 100 test iterations,
# covering the full 10,000 testing images.
test_iter: 100
# Carry out testing every 500 training iterations.
test_interval: 500
# The base learning rate, momentum and the weight decay of the network.
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
# The learning rate policy
lr_policy: "inv"
gamma: 0.0001
power: 0.75
# Display every 100 iterations
display: 100
# The maximum number of iterations
max_iter: 10000
# snapshot intermediate results
snapshot: 5000
snapshot_prefix: "examples/mnist/lenet"
# solver mode: CPU or GPU
solver_mode: GPU
```

Figure 2. lenet_solver.prototxt

Get start with Caffe

- Call Train Function

```
#need to use gpu-Node and go to GPU-node
$bash examples/mnist/train_lenet.sh
```

THANK YOU !