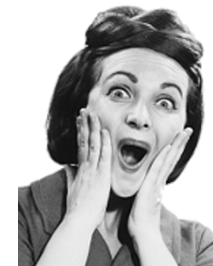


A Brief Introduction to the DNN/CNN Toolbox

Xiaopeng HONG
CMV, OU

Many impl. for DNN!!!

- [Theano](#) – CPU/GPU symbolic expression compiler in python (from LISA lab at University of Montreal)
- [Pylearn2](#) - Pylearn2 is a library designed to make machine learning research easy.
- [Torch](#) – provides a Matlab-like environment for state-of-the-art machine learning algorithms in lua (from Ronan Collobert, Clement Farabet and Koray Kavukcuoglu)
- [DeepLearnToolbox](#) – A Matlab toolbox for Deep Learning (from Rasmus Berg Palm)
- [Cuda-Convnet](#) – A fast C++/CUDA implementation of convolutional (or more generally, feed-forward) neural networks. It can model arbitrary layer connectivity and network depth. Any directed acyclic graph of layers will do. Training is done using the back-propagation algorithm.
- [Deep Belief Networks](#). Matlab code for learning Deep Belief Networks (from Ruslan Salakhutdinov).
- [matrbm](#). Simplified version of Ruslan Salakhutdinov's code, by Andrej Karpathy (Matlab).
- [deepmat](#)- Deepmat, Matlab based deep learning algorithms.
- [Estimating Partition Functions of RBM's](#). Matlab code for estimating partition functions of Restricted Boltzmann Machines using Annealed Importance Sampling (from Ruslan Salakhutdinov).
- [Learning Deep Boltzmann Machines](#) Matlab code for training and fine-tuning Deep Boltzmann Machines (from Ruslan Salakhutdinov).
- [Eblearn.lsh](#) is a LUSH-based machine learning library for doing Energy-Based Learning. It includes code for "Predictive Sparse Decomposition" and other sparse auto-encoder methods for unsupervised learning. [Koray Kavukcuoglu](#) provides Eblearn code for several deep learning papers on this [page](#).
- [Nengo](#)-Nengo is a graphical and scripting based software package for simulating large-scale neural systems.
- [Eblearn](#) is a C++ machine learning library with a BSD license for energy-based learning, convolutional networks, vision/recognition applications, etc. Eblearn is primarily maintained by [Pierre Sermanet](#) at NYU.
- [cudamat](#) is a GPU-based matrix library for Python. Example code for training Neural Networks and Restricted Boltzmann Machines is included.
- [Gnumpy](#) is a Python module that interfaces in a way almost identical to numpy, but does its computations on your computer's GPU. It runs on top of cudamat.
- The [CUV Library](#) (github [link](#)) is a C++ framework with python bindings for easy use of Nvidia CUDA functions on matrices. It contains an RBM implementation, as well as annealed importance sampling code and code to calculate the partition function exactly (from [AIS lab](#) at University of Bonn).
- [3-way factored RBM](#) and [mcRBM](#) is python code calling CUDAMat to train models of natural images (from [Marc'Aurelio Ranzato](#)).
- Matlab code for training [conditional RBMs/DBNs](#) and [factored conditional RBMs](#) (from [Graham Taylor](#)).
- [CXXNET](#) – An implementation of deep convolution neural network in C++.
- [mPoT](#) is python code using CUDAMat and gnumpy to train models of natural images (from [Marc'Aurelio Ranzato](#)).
- [neuralnetworks](#) is a java based gpu library for deep learning algorithms.
- [ConvNet](#) is a matlab based convolutional neural network toolbox.
- [Caffe](#) is a C++/CUDA deep learning framework by [Yangqing Jia](#)
- et c ...



Two relative simple implementation

- Rasmus Berg Palm's toolbox
- Andrew Ng's script

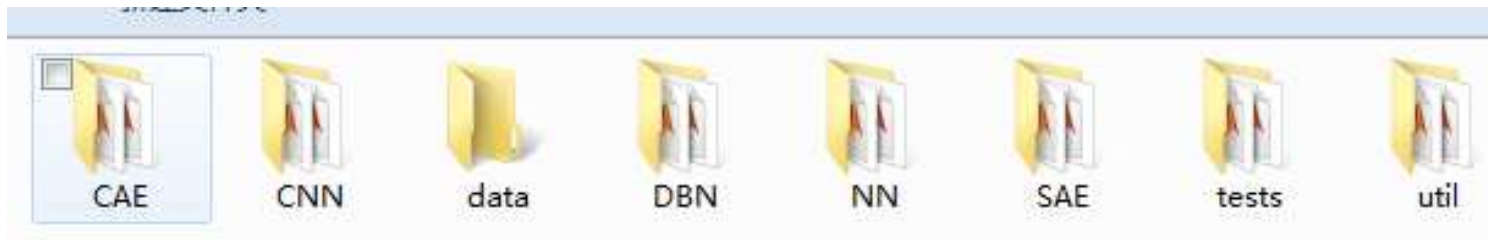


Rasmus Berg Palm's toolbox

- Link for Downloading
- <https://github.com/rasmusbergpalm/DeepLearnToolbox>
- Link for Palm's master thesis
- `@MASTERSTHESIS\{IMM2012-06284, author = "R. B. Palm", title = "Prediction as a candidate for learning deep hierarchical models of data", year = "2012",}`
- http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6284)

Setup

- 1. Download.



`NN/` - A library for Feedforward Backpropagation Neural Networks

`CNN/` - A library for Convolutional Neural Networks

`DBN/` - A library for Deep Belief Networks

`SAE/` - A library for Stacked Auto-Encoders

`CAE/` - A library for Convolutional Auto-Encoders

`util/` - Utility functions used by the libraries

`data/` - Data used by the examples `tests/` - unit tests to verify toolbox is working

- 2. `addpath(genpath('DeepLearnToolbox'));`

test_example_CNN

```
%% ex1 Train a 6c-2s-12c-2s Convolutional neural network
%will run 1 epoch in about 200 second and get around 11% error.
%With 100 epochs you'll get around 1.2% error
rand('state', 0)
cnn.layers = {
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', 6, 'kernelsize', 5) %convolution layer
    struct('type', 's', 'scale', 2) %sub sampling layer
    struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5) %convolution layer
    struct('type', 's', 'scale', 2) %subsampling layer
};
cnn = cnnsetup(cnn, train_x, train_y);

opts.alpha = 1;
opts.batchsize = 50;
opts.numepochs = 1;

cnn = cnnttrain(cnn, train_x, train_y, opts);

[er, bad] = cnntest(cnn, test_x, test_y);

%plot mean squared error
figure; plot(cnn.rL);
```

- CNN is closed to our every day research
- Not that difficult to use (as a beginner)



Andrew Ng's script

- http://deeplearning.stanford.edu/wiki/index.php/MATLAB_Modules
- Autor Encoder is the main focus

```
%% =====  
%% Step 0a: Load data  
% Here we provide the code to load natural image data into x.  
% x will be a 144 * 10000 matrix, where the kth column x(:, k) corresponds to  
% the raw image data from the kth 12x12 image patch sampled.  
% You do not need to change the code below.  
  
x = sampleIMAGESRAW();  
figure('name','Raw images');  
randset = randi(size(x,2),200,1); % A random selection of samples for visualization  
display_network(x(:,randset));  
  
%% =====  
%% Step 0b: Zero-mean the data (by row)  
% You can make use of the mean and repmat/bsxfun functions.  
  
% ----- YOUR CODE HERE -----  
  
%% =====  
%% Step 1a: Implement PCA to obtain xRot  
% Implement PCA to obtain xRot, the matrix in which the data is expressed  
% with respect to the eigenbasis of sigma, which is the matrix U.  
  
% ----- YOUR CODE HERE -----  
xRot = zeros(size(x)); % You need to compute this  
  
%% =====  
%% Step 1b: Check your implementation of PCA  
% The covariance matrix for the data expressed with respect to the basis U  
% should be a diagonal matrix with non-zero entries only along the main  
% diagonal. We will verify this here.  
% Write code to compute the covariance matrix, covar.  
% When visualised as an image, you should see a straight line across the  
% diagonal (non-zero entries) against a blue background (zero entries).  
  
% ----- YOUR CODE HERE -----  
covar = zeros(size(x, 1)); % You need to compute this  
  
7 % Visualise the covariance matrix. You should see a line across the  
% diagonal against a blue background.  
figure('name','Visualisation of covariance matrix');
```



Main focus of this course

- CNN
- Why?
 - "from now on, deep learning with **CNN** has to be considered as the **primary candidate** in essentially any visual recognition task." (Razavian et al, 2014)



Widely used Framework

- Theano/Pylearn2
 - LISA U. Montreal
 - Python & Numpy
- Torch 7
 - NYU
 - Lua/C++
 - Used by FAIR & Google Deep mind & Twitter, etc.
- Cuda-Convnet2
 - U. Toronto
 - C++ / CUDA library
- Caffe/Decaf
 - BVLC, UCB
 - C++ / CUDA
- MatCovnet
 - VGG, U. Oxford
 - Matlab/C++



Widely used Framework

- Theano/Pylearn2
- Torch 7
- Cuda-Convnet2
- Caffe/Decaf
- MatCovnet



Theano/Pylearn2

- LISA U. Montreal



Yoshua BENGIO

I believe that the recent surge of interest in NNets just means that the **machine learning community wasted many years** not exploring them, in the 1996-2006 decade, mostly.

There is also hype, especially if you consider the media. That is unfortunate and dangerous, and will be exploited especially by companies trying to make a quick buck.

The danger is to see another bust when **wild promises** are not followed by outstanding results. Science mostly moves by small steps and we should stay humble.



Computer Vision Achievement

- **Maxout nets set the state of the art on MNIST, SVHN, CIFAR-10, and CIFAR-100** ("Maxout Networks", Goodfellow et al 2013)
- **Won Emotiiv face recognition challenge** ("Combining Modality Specific Deep Neural Network Models for Emotion Recognition in Video", Ebrahimi et al 2013)
- **Won Transfer Learning Challenge at NIPS workshops** ("Scaling up spike-and-slab models for unsupervised feature learning", Goodfellow et al 2013)
- **Won DARPA's Transfer Learning Challenge** ("Unsupervised and Transfer Learning Challenge: A Deep Learning Approach", Mesnil et al 2012)
- **State of the art face recognition on Toronto Face Database** ("Disentangling Factors of Variation for Facial Expression Recognition," Rifai et al 2012)
- **No ImageNet results yet**

Slides from Ian Goodfellow in his talk 'Theano and Pylearn2 for deep learning applied to computer vision'



Theano

- ‘Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.’
- Theano features:
 - **tight integration with NumPy** – Use *numpy.ndarray* in Theano-compiled functions.
 - **transparent use of a GPU** – Perform data-intensive calculations up to 140x faster than with CPU. (for 32-bit floating point only)
 - **efficient symbolic differentiation** – Theano does your derivatives for function with one or many inputs.
 - **speed and stability optimizations** – Get the right answer for $\log(1+x)$ even when x is really tiny.
 - **dynamic C code generation** – Evaluate expressions faster.
 - **extensive unit-testing and self-verification** – Detect and diagnose many types of mistake.
- <http://deeplearning.net/software/theano/>
- https://github.com/goodfeli/theano_exercises



Theano Deep Learning Toolbox

- <https://github.com/lisa-lab/DeepLearningTutorials>
- This toolbox includes some of the most important deep learning algorithms such as Auto Encoder, DBN, CNN, and RBM
- <http://deeplearning.net/tutorial/deeplearning.pdf>

Deep Learning Tutorial

Release 0.1

```

class LeNetConvPoolLayer(object):
    """Pool Layer of a convolutional network """

    # initialize weights with random weights
    W_bound = numpy.sqrt(6. / (fan_in + fan_out))
    self.W = theano.shared(
        numpy.asarray(
            rng.uniform(low=-W_bound, high=W_bound, size=filter_shape),
            dtype=theano.config.floatX
        ),
        borrow=True
    )

    # the bias is a 1D tensor -- one bias per output feature map
    b_values = numpy.zeros((filter_shape[0],), dtype=theano.config.floatX)
    self.b = theano.shared(value=b_values, borrow=True)

    # convolve input feature maps with filters
    conv_out = conv.conv2d(
        input=input,
        filters=self.W,
        filter_shape=filter_shape,
        image_shape=image_shape
    )

    # downsample each feature map individually, using maxpooling
    pooled_out = downsample.max_pool_2d(
        input=conv_out,
        ds=poolsize,
        ignore_border=True
    )

```

Pylearn2

- Pylearn2 is a machine learning library.
- Most of its functionality is built on [Theano](#)
 - write Pylearn2 new models, algorithms, etc using mathematical expressions
 - Theano will optimize and stabilize those expressions, and
 - compile them to a backend of your choice (CPU or GPU).'
- <http://deeplearning.net/software/pylearn2/>
- <https://github.com/lisa-lab/pylearn2/>

Pylearn2

- A machine learning toolbox for easy scientific experimentation.
- Researchers add features as they need them. We avoid getting bogged down by too much top-down planning in advance.
- Pylearn2 may wrap other libraries such as scikit-learn when this is practical
- To provide great flexibility and make it possible for a researcher to do almost anything
- Many deep learning reference implementations
- Small framework for all what is needed for one normal MLP/RBM/SDA/Convolution experiments.

Setup

- <http://deeplearning.net/software/pylearn2/>

Download and installation

There is no PyPI download yet, so Pylearn2 cannot be installed using e.g. `pip`. You must check out the bleeding-edge/development version from GitHub using:

```
git clone git://github.com/lisa-lab/pylearn2.git
```

To make Pylearn2 available in your Python installation, run the following command in the top-level `pylearn2` directory (which should have been created by the `git clone` command):

```
python setup.py develop
```

You may need to use `sudo` to invoke this command with administrator privileges. If you do not have such access (or would rather not add Pylearn2 to the global `site-packages` directory) you can instead install it in a local directory by issuing the command:

```
python setup.py develop --user
```

This command will also compile the Cython extensions required for e.g. `pylearn2.train_extensions.window_flip`.

Alternatively, you can make Pylearn2 available by adding the installation directory to your `PYTHONPATH` environment variable, but note that changing your `PYTHONPATH` to make sure the extension `.so` files are built and accessible within the source tree, e.g. with `python setup.py build_ext --inplace`.

Data path

For some tutorials and tests you will also need to set your `PYLEARN2_DATA_PATH` variable. On Linux, the best way to do this is to add a line to your `~/.bashrc` file:

```
export PYLEARN2_DATA_PATH=/data/lisa/data
```

Note that this is only an example, and if you are not in the LISA lab, you will need to choose a directory path that is valid on your filesystem. Simply choose a path with Pylearn2.



How to get start

- <http://daemonmaker.blogspot.ca/2014/10/a-first-experiment-with-pylearn2.html>

A First Experiment with Pylearn2

Vincent Dumoulin recently wrote a great blog post titled *Your models in Pylearn2* that shows how to quickly implement a new model idea in Pylearn2. However Pylearn2 has a fair number of models already implemented. This post is meant to compliment his post by explaining how to setup and run a basic experiment using existing components in Pylearn2.

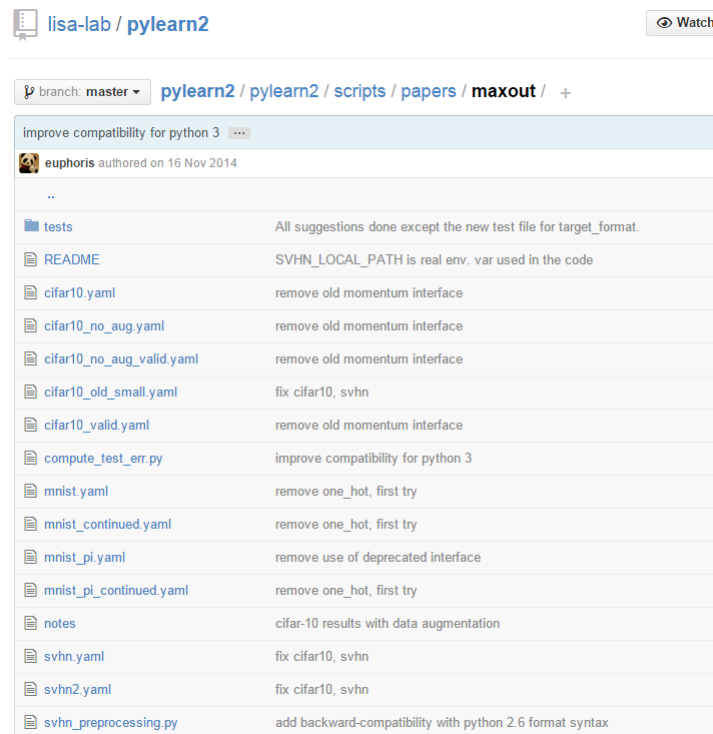
In this tutorial we will train a very simple single layer softmax regression model on MNIST, a database of handwritten digits. Softmax is a generalization of a binary predictor called logistic regression to the prediction of one of many classes. The task will be to identify which digit was written, i.e. classify the image into the classes 0-9.

This same task is addressed in the *Softmax regression Pylearn2 tutorial*. This post will borrow from that tutorial. However Pylearn2 is feature rich allowing one to control everything from which model to train and which dataset to train it on to fine grained control over the training and the ability to monitor and save statistics about an experiment. For the sake of simplicity and understanding we will not be using most of them and as such this tutorial will be simpler.



A Pylearn2 example

- Training a convolutional network on CIFAR-10 with maxout units:
 - <https://github.com/lisa-lab/pylearn2/tree/master/pylearn2/scripts/papers/maxout>



lisa-lab / pylearn2 Watch

branch: master pylearn2 / pylearn2 / scripts / papers / maxout / +

improve compatibility for python 3 ...

euphoris authored on 16 Nov 2014

..

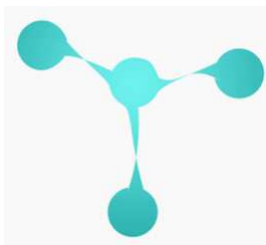
tests	All suggestions done except the new test file for target_format.
README	SVHN_LOCAL_PATH is real env. var used in the code
cifar10.yaml	remove old momentum interface
cifar10_no_aug.yaml	remove old momentum interface
cifar10_no_aug_valid.yaml	remove old momentum interface
cifar10_old_small.yaml	fix cifar10, svhn
cifar10_valid.yaml	remove old momentum interface
compute_test_err.py	improve compatibility for python 3
mnist.yaml	remove one_hot, first try
mnist_continued.yaml	remove one_hot, first try
mnist_pi.yaml	remove use of deprecated interface
mnist_pi_continued.yaml	remove one_hot, first try
notes	cifar-10 results with data augmentation
svhn.yaml	fix cifar10, svhn
svhn2.yaml	fix cifar10, svhn
svhn_preprocessing.py	add backward-compatibility with python 2.6 format syntax



Widely used Framework

- Theano/Pylearn2
- **Torch 7**
- Cuda-Convnet2
- Caffe/Decaf
- MatCovnet





Torch

- **Torch** is an [open source deep learning](#) library for the [Lua](#) programming language and a [scientific computing](#) framework for [machine learning](#) algorithms. It uses [LuaJIT](#) and an underlying [C](#) implementation.
 - For Mac OS X and Linux
 - <http://torch.ch/docs/getting-started.html#>
 - FAIR & ‘Deep mind’ use it!
 - Always aimed large-scale learning
 - Includes lots of packages for neural networks, optimization, graphical models, image processing → More than 50,000 downloads, universities and major industrial labs

1. ^ "Torch: a modular machine learning software library" [↗](#). 30 October 2002. Retrieved 24 April 2014.

Getting started

- Torch's main site and resources:
- www.torch.ch
- Tutorials for Torch:
 - <http://torch.madbits.com>
- 'Everything Torch'
- <https://github.com/torch/torch7/wiki/Cheatsheet>
- Lua: <http://www.lua.org>
- LuaJIT: <http://luajit.org/luajit.html>

Packages for DL

- **Neural Network Package for Torch**
 - <http://torch.cogbits.com/doc/nn/index.html>
- **DP Package**
 - <https://github.com/nicholas-leonard/dp>



fbcunn - Deep Learning CUDA Extensions from FAIR

- Fbcunn contains highly engineered deep learning modules for GPUs, to accelerate your own deep learning endeavors.
- It plugs into the [Torch-7](#) framework and installs seamlessly via luarocks, and is fully compatible with torch's [nn](#) package.'
- <https://facebook.github.io/fbcunn/fbcunn/>



fbcunn

- Fast spatial convolution modules that use FFT to accelerate convolutions. [We wrote a paper about them](#) if you'd like to read more.
- Fast Temporal convolutions that are 1.5x to 10x faster compared to Torch's cunn implementations.
- nn.DataParallel and nn.ModelParallel containers. Plug your model in them and see it accelerate over multiple GPUs
- Wrappers to use FFT/IFFT as nn modules.
- Fast LookupTable that is used for Neural Language Models and word embeddings. Much faster than the one in torch/nn
- Hierarchical SoftMax module, now classifying 1 million classes is a practically viable strategy
- LP and Max Pooling over feature maps (usable for MaxOut).
- more goodies. Full documentation and spec is here: <https://facebook.github.io/fbcunn/fbcunn/>

Fast Convolutional Nets With fbfft: A GPU Performance Evaluation

[Nicolas Vasilache](#), [Jeff Johnson](#), [Michael Mathieu](#), [Soumith Chintala](#), [Serkan Piantino](#), [Yann LeCun](#)

Widely used Framework

- Theano/Pylearn2
- Torch 7
- **Cuda-Convnet2**
- Caffe/Decaf
- MatCovnet



Cuda-Convnet2

- By Alex Krizhevsky
- <https://code.google.com/p/cuda-convnet2/>
- Very fast on state-of-the-art GPUs with Multi-GPU parallelism

This is an update to [cuda-convnet](#).

This project has three major new features relative to cuda-convnet:

1. Improved training times on Kepler-generation Nvidia GPUs (Geforce Titan, K20, K40).
2. Multi-GPU training support implementing data parallelism, model parallelism, and the hybrid approach described in [One weird trick for parallelizing convolutional neural networks](#).
3. Less-polished code and incomplete (but improving) documentation.

Documentation

Usage

- [Compiling](#) -- how to compile the code
- [Data](#) -- how to generate training data
- [TrainingExample](#) -- how to train an example network
- [LayerParams](#) -- how to specify a custom network
- [MultiGPU](#) -- how to train multi-GPU networks
- [ShowNet](#) -- how to look inside trained networks



The screenshot shows the GitHub repository page for 'cuda-convnet2'. It features a blue house icon with a white '{P}' symbol. The text 'cuda-convnet2' is in a large, bold, grey font, with the tagline 'Fast convolutional neural networks in C++/CUDA' below it. At the bottom, there is a navigation bar with links for 'Project Home', 'Wiki', 'Issues', and 'Source'.



Cuda-Convnet1

This is a fast C++/CUDA implementation of convolutional (or more generally, feed-forward) neural networks. It can model arbitrary layer connectivity and network depth. Any directed acyclic graph of layers will do. Training is done using the back-propagation algorithm.

Fermi-generation GPU (GTX 4xx, GTX 5xx, or Tesla equivalent) required.

Documentation

- [Compiling](#) -- how to check out and compile this code.
- [Data](#) -- what kind of data this net can train on.
- [LayerParams](#) -- how to specify an architecture for the net.
- [NeuronTypes](#) -- types of hidden unit nonlinearities.
- [TrainingNet](#) -- how to train the net.
- [Options](#) -- the command-line arguments that the net takes.
- [ViewingNet](#) -- how to look inside the checkpoints saved by the net.
- [CheckingGradients](#) -- how to numerically test the gradients for correctness.

Fast results

- 11% error on [CIFAR-10](#) in **75 minutes**, with image translations and horizontal reflections ([def](#), [params](#)).
- 13% error on CIFAR-10 in **25 minutes**, with image translations and horizontal reflections ([def](#), [params](#)).
 - See [Methodology](#) for details of training.

Filters learned by this net:



- 18% error on CIFAR-10 in **20 minutes**, without any image translations/transformations/preprocessing ([def](#), [params](#)).
- 26% error on CIFAR-10 in **80 seconds**, without any image translations/transformations/preprocessing ([def](#), [params](#)).



ConvNet

- By Sergey Demyanov
- <https://github.com/sdemyanov/ConvNet>
- Convolutional Neural Networks for Matlab, including Invariant Backpropagation algorithm (IBP). Has versions for GPU and CPU, written on CUDA, C++ and Matlab. All versions work identically. The GPU version uses kernels from Alex Krizhevsky's library 'cuda-convnet2'.
-



Some other CUDA DNN toolbox

- **NVIDIA cuDNN**
- **CUDA CNN**

NVIDIA cuDNN


- [cuDNN](#) is an NVIDIA library with functionality used by deep neural network.
- <http://deeplearning.net/software/theano/library/sandbox/cuda/dnn.html>

CUDA CNN 2

- CudaCnn is a Convolutional neural networks library written on C++/CUDA with Matlab frontend.
- <http://www.mathworks.com/matlabcentral/fileexchange/24291-cnn-convolutional-neural-network-class>




CUDA CNN 1



CNN - Convolutional neural network class

by Mihail Sirotenko
28 May 2009 (Updated 24 Oct 2012)

This project provides matlab class for implementation of convolutional neural networks.

 Watch this File

- <http://www.mathworks.com/matlabcentral/fileexchange/24291-cnn-convolutional-neural-network-class>

Widely used Framework

- Theano/Pylearn2
- Torch 7
- Cuda-Convnet2
- **Caffe/Decaf**
- MatCovnet



Caffe

- ‘Caffe is a deep learning framework developed with cleanliness, readability, and speed in mind. It was created by [Yangqing Jia](#) during his PhD at UC Berkeley, and is in active development by the Berkeley Vision and Learning Center ([BVLC](#)) and by community contributors.’
- Tested on Linux and Mac OS X
- <http://caffe.berkeleyvision.org/>
- <http://caffe.berkeleyvision.org/tutorial/>

```
@article{jia2014caffe,  
  Author = {Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan  
and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor},  
  Journal = {arXiv preprint arXiv:1408.5093},  
  Title = {Caffe: Convolutional Architecture for Fast Feature Embedding},  
  Year = {2014}  
}
```



DIY Deep Learning for Vision: a Hands-On Tutorial with Caffe

- A network is a set of layers connected as a DAG:

```
name: "dummy-net"  
  
layers { name: "data" ...}  
layers { name: "conv" ...}  
layers { name: "pool" ...}  
... more layers ...  
layers { name: "loss" ...}
```

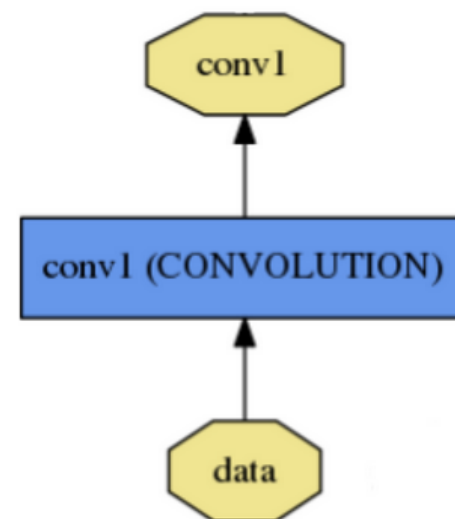
Slides by Evan Shelhamer, Jeff Donahue, Yangqing Jia, Ross Girshick
In 'Brewing Deep Networks With Caffe'

DIY Deep Learning for Vision: a Hands-On Tutorial with Caffe

```
name: "conv1"  
type: CONVOLUTION  
bottom: "data"  
top: "conv1"  
convolution_param {  
  num_output: 20  
  kernel_size: 5  
  stride: 1  
  weight_filler {  
    type: "xavier"  
  }  
}
```

name, type, and the connection structure (input blobs and output blobs)

layer-specific parameters



- Every layer type defines

- Setup
- Forward
- Backward

* Nets + Layers are defined by [protobuf](#) schema



The Caffe Model Zoo

- - open collection of deep models to share innovation
 - VGG ILSVRC14 + Devil models
 - Network-in-Network / CCCP
 - MIT Places scene recognition
 - ...

http://caffe.berkeleyvision.org/model_zoo.html

Berkeley-trained models

- [Finetuning on Flickr Style](#): same as provided in [models/](#), but listed here as a Gist for an example.
- [BVLC GoogleNet](#)

Network in Network model

The Network in Network model is described in the following [ICLR-2014 paper](#):

Network In Network
M. Lin, Q. Chen, S. Yan
International Conference on Learning Representations, 2014 (arXiv:1409.1556)

please cite the paper if you use the models.

Models:

- [NIN-Imagenet](#): a small(29MB) model for imagenet, yet performs slightly better than AlexNet, and fast to train.
- [NIN-CIFAR10](#): NIN model on CIFAR10, originally published in the paper [Network in Network](#). The error rate of this model is 10.4% on CIFAR10.

Models from the BMVC-2014 paper "Return of the Devil in the Details: Delving Deep into Convolutional Nets"



DeCaf



Decaf

Decaf is a general python framework for deep convolutional neural networks, relying on a set of scientific computation modules (such as numpy/scipy) to efficiently run CNN models without the need of a GPU. Decaf is still under development but an imagenet classification demo could be checked out [here](#).

- ▶ Home
- ▶ Research
- ▶ Publications
- ▶ Software
- ▶ Teaching
- ▶ CV



Yangqing Jia (贾扬清)

jiayq@google.com

I am currently a research scientist at Google working on computer vision. I have recently finished my Ph.D. Berkeley, advised by Prof. [Trevor Darrell](#).

During my graduate study I've worked/interned at the National University of Singapore, Microsoft Research Google Research. I obtained my bachelor and master degrees at Tsinghua University, China.

I am the author of [Caffe](#), which is now a [BVLC](#) maintained, open-source deep learning framework.

I do have a [LinkedIn profile](#), and a somewhat outdated [CV](#).

Research

My current research topics include:

- Learning better structures for image feature extraction.
- Explaining human generalization behavior with visually grounded cogscience models.
- Making large-scale vision feasible and affordable.

(Most recent publications to be added)

Recent Publications

Links to: [\[Full List\]](#) [\[Google Scholar\]](#)



DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition

J Donahue, Y Jia, O Vinyals, J Hoffman, N Zhang, E Tzeng, T Darrell. arXiv preprint.

[\[ArXiv Link\]](#) [\[Live Demo\]](#) [\[Software\]](#) [\[Pretrained ImageNet Model\]](#)

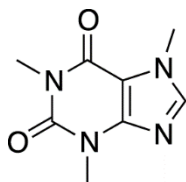
- <http://daggerfs.com/>



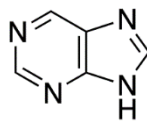


NUS Purine

- **Purine: A Bi-graph based deep learning framework**



caffeine



purine

In purine the math functions and core computations are adopted from Caffe.

Pictures from Min LIN, Shuo LI, Xuan LUO, Shuicheng YAN's slides

https://docs.google.com/presentation/d/1_LtkveeE1XTTsCthIruLb2bL4pcquhx2mRqXgbiBJsw/edit#slide=id.p26

Purine: A bi-graph based deep learning framework

Min Lin, Shuo Li, Xuan Luo, Shuicheng Yan

(Submitted on 19 Dec 2014 (v1), last revised 20 Jan 2015 (this version, v3))

In this paper, we introduce a novel deep learning framework, termed Purine. In Purine, a deep network is expressed as a bipartite graph (bi-graph), which is composed of interconnected operators and data tensors. With the bi-graph abstraction, networks are easily solvable with event-driven task dispatcher. We then demonstrate that different parallelism schemes over GPUs and/or CPUs on single or multiple PCs can be universally implemented by graph composition. This eases researchers from coding for various parallelization schemes, and the same dispatcher can be used for solving variant graphs. Scheduled by the task dispatcher, memory transfers are fully overlapped with other computations, which greatly reduce the communication overhead and help us achieve approximate linear acceleration.

Widely used Framework

- Theano/Pylearn2
- Torch 7
- Cuda-Convnet2
- Caffe/Decaf
- **MatCovnet**



Oxford Impl. MatConvNet

- <http://www.vlfeat.org/matconvnet/>
- Andrea Vedaldi, Karel Lenc. MatConvNet - Convolutional Neural Networks for MATLAB. <http://arxiv.org/pdf/1412.4564v1.pdf>

MatConvNet: CNNs for MATLAB

MatConvNet: CNNs for MATLAB

MatConvNet is a MATLAB toolbox implementing *Convolutional Neural Networks* (CNNs) for computer vision applications. It is simple, efficient, and can run and learn state-of-the-art CNNs. Several example CNNs are included to classify and encode images.

- **Obtaining MatConvNet**
 - Tarball for version 1.0-beta8
 - GIT repository
 - PDF manual (see also MATLAB inline help).
- **Getting started**
 - Installation instructions
 - Using pre-trained models
 - Training your own models
 - Working with GPU accelerated code
- **What's new**
 - Changes

Citing. If you use MatConvNet in your work, please cite: "MatConvNet - Convolutional Neural Networks for MATLAB", A. Vedaldi and K. Lenc, arXiv:1412.4564, 2014.

```
@article{arxiv:1412.4564,  
  author = {A. Vedaldi and K. Lenc},  
  title = {MatConvNet -- Convolutional Neural Networks for MATLAB},  
  journal = {CoRR},  
  volume = {abs/1412.4564},  
  year = {2014},  
}
```



Andrea Vedaldi, Ph.D. vedaldi@robots.ox.ac.uk
Associate Professor in Engineering Science
[Information Engineering Building](#) 30.05, Parks Road, Oxford, OX1 3PJ
[Visual Geometry Group \(directions\)](#)
Tel. +44 1865 273 127

[Résumé](#) [Google Scholar](#)

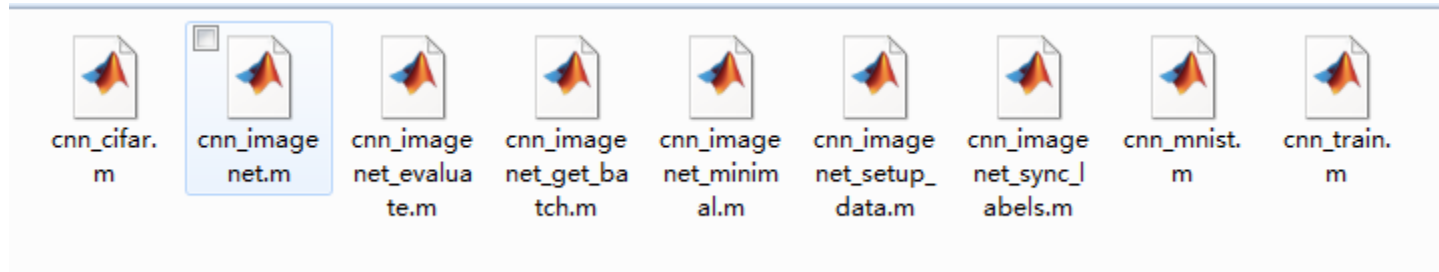


Achievement

This is a summary of the performance of these models on the ILSVRC 2012 validation data:

model	top-1 err.	top-5 err.	images/s
caffe-ref	42.4	19.6	132.9
caffe-alex	42.6	19.6	131.4
vgg-s	36.7	15.5	120.0
vgg-m	37.8	16.1	127.6
vgg-f	41.9	19.3	147.0
vgg-verydeep-19	30.5	11.3	40.3
vgg-verydeep-16	30.9	11.2	46.2





```

% Define a network similar to LeNet
f=1/100 ;
net.layers = {} ;
net.layers{end+1} = struct('type', 'conv', ...
    'filters', f*randn(5,5,1,20, 'single'), ...
    'biases', zeros(1, 20, 'single'), ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
    'method', 'max', ...
    'pool', [2 2], ...
    'stride', 2, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
    'filters', f*randn(5,5,20,50, 'single'),...
    'biases', zeros(1,50,'single'), ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
    'method', 'max', ...
    'pool', [2 2], ...
    'stride', 2, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
    'filters', f*randn(4,4,50,500, 'single'),...
    'biases', zeros(1,500,'single'), ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'conv', ...
    'filters', f*randn(1,1,500,10, 'single'),...
    'biases', zeros(1,10,'single'), ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'softmaxloss') ;

```



Using the pre-trained model (1/2)

Download the pretrained models

- VGG models from the [Very Deep Convolutional Networks for Large-Scale Visual Recognition](#) project.

Citation: 'Very Deep Convolutional Networks for Large-Scale Image Recognition', *Karen Simonyan and Andrew Zisserman*, arXiv technical report, 2014, ([paper](#)).

- [imagenet-vgg-verydeep-16](#)
- [imagenet-vgg-verydeep-19](#)

- VGG models from the [Return of the Devil](#) paper (v1.0.1).

Citation: 'Return of the Devil in the Details: Delving Deep into Convolutional Networks', *Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman*, BMVC 2014 ([BibTex](#) and [paper](#)).

- [imagenet-vgg-f](#)
- [imagenet-vgg-m](#)
- [imagenet-vgg-s](#)
- [imagenet-vgg-m-2048](#)
- [imagenet-vgg-m-1024](#)
- [imagenet-vgg-m-128](#)

- Berkeley [Caffe reference models](#) (version downloaded on September 2014).

Citation: Please see [Caffe homepage](#).

- [imagenet-caffe-ref](#)
- [imagenet-caffe-alex](#)



Using the pre-trained model (2/2)

Using the pretrained models

In order to run, say, `imagenet-vgg-s` on a test image, use:

```
% setup MtConvNet in MATLAB
run matlab/vl_setupnn

% download a pre-trained CNN from the web
urlwrite('http://www.vlfeat.org/sandbox-matconvnet/models/imagenet-vgg-f.mat', ...
    'imagenet-vgg-f.mat');
net = load('imagenet-vgg-f.mat');

% obtain and preprocess an image
im = imread('peppers.png');
im_ = single(im); % note: 255 range
im_ = imresize(im_, net.normalization.imageSize(1:2));
im_ = im_ - net.normalization.averageImage;

% run the CNN
res = vl_simplenn(net, im_);

% show the classification result
scores = squeeze(gather(res(end).x));
[bestScore, best] = max(scores);
figure(1); clf; imagesc(im);
title(sprintf('%s (%d), score %.3f', ...
    net.classes.description{best}, best, bestScore));
```

<http://www.vlfeat.org/matconvnet/pretrained/>

Java Implementation

- <https://github.com/ivan-vasilev/neuralnetworks>
- By Ivan Vasilev
- This is a Java implementation of some of the algorithms for training deep neural networks. GPU support is provided via the OpenCL and Aparapi. The architecture is designed with modularity, extensibility and pluggability in mind.
-

Some other toolbox

- Overfeat
- NUS Purine



OverFeat: Object Recognizer, Feature Extractor

- <http://cilvr.nyu.edu/doku.php?id=software:overfeat:start>
- OverFeat is a Convolutional Network-based image features extractor and classifier.
- The underlying system is described in the following paper:
- Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun: “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, International Conference on Learning Representations (ICLR 2014), April 2014. ([OpenReview.net](#)), ([arXiv:1312.6229](#)), ([BibTeX](#)).

OverFeat: Object Recognizer, Feature Extractor

- It is primarily a feature extractor in that data augmentation for better classification is not provided. Different data augmentation schemes can be implemented to improve classification results, see the OverFeat paper for more details.
- OverFeat was trained on the ImageNet dataset and participated in the ImageNet 2013 competition.
- This package allows researchers to use OverFeat to recognize images and extract features.
- A library with C++ source code is provided for running the OverFeat convolutional network, together with wrappers in various scripting languages (Python, Lua, Matlab coming soon).
- OverFeat was trained with the Torch7 package (<http://www.torch.ch>). This package provides tools to run the network in a standalone fashion. The training code is not part of this package.



Comparisons and Summary

- All
 - Implements deep CNN
 - Well Supports GPU
 - Open-source
- No strict winners
- Try and choose the one that best fits your work

