Deep Learning:
and Deep Data-Science

12 May 2015

roelof@kth.se
www.csc.kth.se/~roelof/

Graph Technologies R&D
roelof@graph-technologies.com

slides online at:
https://www.slideshare.net/roelofp/deep-learning-as-a-catdog-detector
BUT FIRST...

are you a...

CAT PERSON?

DOG PERSON?
in the next few minutes
we’ll be making a

CAT vs DOG

DETECTOR
main Libraries

• scikit-learn (machine learning)
  http://scikit-learn.org

• caffe (deep learning) – for training deep neural nets
  (for today: loading a pre-trained one)
  http://caffe.berkeleyvision.org

• theano (efficient gpu-powered math)
  http://www.deeplearning.net/software/theano/

• ipython notebook
  http://ipython.org/notebook.html
scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

— Examples

Regression
Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

— Examples

Clustering
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

— Examples

Dimensionality reduction
Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization.

— Examples

Model selection
Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics.

— Examples

Preprocessing
Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction.

— Examples
Caffe

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley. Caffe is released under the BSD 2-Clause license.

Check out our web image classification demo!

Why Caffe?

Expressive architecture encourages application and innovation. Models and optimization are defined by configuration without hard-coding. Switch between CPU and GPU by setting a single flag to train on a GPU machine then deploy to commodity clusters or mobile devices.

Extensible code fosters active development. In Caffe's first year, it has been forked by over 1,000 developers and had many significant changes contributed back. Thanks to these contributors the framework tracks the state-of-the-art in both code and models.

Speed makes Caffe perfect for research experiments and industry deployment. Caffe can process over 60M images per day with a single NVIDIA K40 GPU*: That's 1 ms/image for inference and 4 ms/image for learning. We believe that Caffe is the fastest convnet implementation available.

Community: Caffe already powers academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Join our community of brewers on the caffe-users group and Github.

* With the ILSVRC2012-winning SuperVision model and caching IO. Consult performance details.
Welcome

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. (float32 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.

Theano has been powering large-scale computationally intensive scientific investigations since 2007. But it is also approachable enough to be used in the classroom (IFT6266 at the University of Montreal).

News

- We support [cuDNN](https://www.cudnn.org) if it is installed by the user.
- Open Machine Learning Workshop 2014 [presentation](#).
- Colin Raffel [tutorial on Theano](#).
- Ian Goodfellow did a [12h class with exercises on Theano](#).
The IPython Notebook

The IPython Notebook is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media, as shown in this example session:

It aims to be an agile tool for both exploratory computation and data analysis, and provides a platform to support **reproducible research**, since all inputs and outputs may be stored in a one-to-one way in notebook documents.
Code is ahead, soon...
I promise :}

BEAR
WITH
ME
“Data science is clearly a blend of the hackers' art, statistics and machine learning...”

—Hilary Mason & Chris Wiggins, 2010
Hacking Skills  

Machine Learning  

Math & Statistics Knowledge  

Data Science  

Danger Zone!  

Traditional Research  

Substantive Expertise  

(Drew Connoway 2010)
> Features = Awesomeness

1 feature
$> \text{Features} = \text{Awesomeness}$

1 feature

2 features
1 feature

2 features
too few features/dimensions = overfitting
Features = Awesomeness

1 feature

2 features

too few features/dimensions = overfitting

3 features
More Features = Awesomeness!

1 feature

2 features

3 features
++ Data Needs also grow!
(picture by Dato)
Deep Learning?

• A host of statistical machine learning techniques
• Enables the automatic learning of feature hierarchies
• Generally based on artificial neural networks
Deep Learning

(picture by Dato)
Deep Learning?

- Manually designed features are often over-specified, incomplete and take a long time to design and validate.

- **Learned Features** are easy to adapt, fast to learn.

- Deep learning provides a very flexible, (almost?) universal, learnable framework for **representing** world, visual and linguistic information.

- Deep learning can learn **unsupervised** (from raw text/audio/images/whatever content) and **supervised** (with specific labels like positive/negative).

(as summarised by Richard Socher 2014)
2006+: The Deep Learning Conspirators

Stacked Autoencoders

Université de Montréal

Bengio

Google

Hinton

University of Toronto

Restricted Boltzmann Machine

Facebook

LeCun

Sparse Representations
Audio Recognition

![Chart showing error rates for traditional and deep learning audio recognition over time. The chart indicates a decrease in error rates for both methods, with deep learning showing a more significant improvement.] (chart by Clarifai)
Image Recognition

(chart by Clarifai)
Natural Language Processing
Natural Language Processing

cloudy days don’t last forever.
DL? How?

almost at the code...
Deep neural networks learn hierarchical feature representations
Coding time!

our ingredients…

(picture by Dato)
Kaggle’s Cat vs Dog dataset (25k dog/cat pictures)

(picture by Dato)
Create an algorithm to distinguish dogs from cats

In this competition, you’ll write an algorithm to classify whether images contain either a dog or a cat. This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.

Data Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Available Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>sampleSubmission</td>
<td>.csv (86.82 kb)</td>
</tr>
<tr>
<td>test</td>
<td>.zip (271.15 mb)</td>
</tr>
<tr>
<td>train</td>
<td>.zip (543.16 mb)</td>
</tr>
</tbody>
</table>
Pretrained Convolutional Neural Net (CNN)
97% accuracy in < 1h

MultiLayer Perceptron

(picture by Dato)
1 Load Pretrained Net

2 Extract features for all training images

3. train MLP on those features
Cooking Instructions

1. Load Pretrained Net
No Free Lunch… But Free Models!

Model Zoo
Liwei Wang edited this page 3 days ago · 18 revisions

Trained models are posted here as links to Github Gists. Check out the model zoo documentation for details.

To acquire a model:

1. download the model gist by `./scripts/download_model_from_gist.sh <gist_id> <dirname>` to load the model metadata, architecture, solver configuration, and so on. (`<dirname>` is optional and defaults to caffe/models).
2. download the model weights by `./scripts/download_model_binary.py <model_dir>` where `<model_dir>` is the gist directory from the first step.

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

https://github.com/BVLC/caffe/wiki/Model-Zoo
# imports

```python
%matplotlib inline
import logging
from glob import glob
from random import shuffle
import pickle

# Make sure that caffe is on the python path:
caffe_root = '../'
import sys
sys.path.insert(0, caffe_root + 'python')
import caffe

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```
# load pretrained deep neural net

```python
def png_to_np(basedir, fetch_target=False):
    logging.getLogger().setLevel(logging.INFO)
    caffe.set_mode_gpu()
    net = caffe.Classifier(MODEL_FILE, PRETRAINED,
                           mean=np.load(caffe_root + 'python/caffe/imagenet/
                            ilsvrc_2012_mean.npy').mean(1).mean(1),
                           channel_swap=(2,1,0),
                           raw_scale=255,
                           image_dims=(256, 256))
```

(convnet from Krizhevsky et al.'s NIPS 2012 ImageNet classification paper)
1 Load Pretrained Net

2 Extract features for all training images
# feed image into the network and return internal feature representation of layer fc6

def activate(net, im):
    input_image = caffe.io.load_image(im)
    # Resize the image to the standard (256, 256) and oversample net input sized crops.
    input_oversampled = caffe.io.oversample([caffe.io.resize_image(input_image, net.image_dims)], net.crop_dims)
    # 'data' is the input blob name in the model definition, so we preprocess for that input.
    caffe_input = np.asarray([net.transformer.preprocess('data', in_) for in_ in input_oversampled])
    # forward() takes keyword args for the input blobs with preprocessed input arrays.
    predicted = net.forward(data=caffe_input)
    # Activation of all convolutional layers and first fully connected
    feat = net.blobs['fc6'].data[0]
    return feat
# extract features from images

def activate(net, file):
    feature_info = activate(net, files[0])
    feature_count = feature_info.shape[0]
    feature_dtype = feature_info.dtype
    data = np.zeros((len(files), feature_count), dtype=feature_dtype)
    for n, im in enumerate(files):
        data[n, :] = activate(net, im)
        if n % 1000 == 0:
            print('Reading in image', n)
    return data, target, files
# dump features as pickle file

```python
x, y, filenames = png_to_np(
    '/mnt/pet/train/', fetch_target=True
)
pickle.dump(x, open('saved_x_v2.pkl', 'wb'))
pickle.dump(y, open('saved_y_v2.pkl', 'wb'))
pickle.dump(filenames, open('saved_filenames_v2.pkl', 'wb'))
```

Reading in image 0
Reading in image 1000
Reading in image 2000
Reading in image 3000
Reading in image 4000
Reading in image 5000
Reading in image 6000
Reading in image 7000
Reading in image 8000
(...)

```
1. Load Pretrained Net

2. Extract features for all training images

3. train MLP on those features
Pylearn2:
Multilayer Perceptron (MLP) on top of extracted features

```python
# imports
demo

1 from pylearn2.models import mlp
2 from pylearn2.costs.mlp.dropout import Dropout
3 from pylearn2.training_algorithms import sgd, learning_rule
4 from pylearn2.termination_criteria import EpochCounter
5 from pylearn2.datasets import DenseDesignMatrix
6 from pylearn2.train import Train
7 from pylearn2.train_extensions import best_params
8 from pylearn2.space import VectorSpace
9
10 import pickle
11 import numpy as np
```
# load earlier extracted features and labels
# convert to input that pylearn understands

def demo:
    x = pickle.load(open('saved_x_v2.pkl', 'rb'))
    y = pickle.load(open('saved_y_v2.pkl', 'rb'))
    filenames = pickle.load(open('saved_filenames_v2.pkl', 'rb'))
    y = to_one_hot(y)
    in_space = VectorSpace(dim=x.shape[1])
    full = DenseDesignMatrix(X=x, y=y)
# create layers of MLP

# with softmax as final layer

```
# trainer initialized with SGD, momentum, dropout

l1 = mlp.RectifiedLinear(layer_name='l1',
                        sparse_init=12,
                        dim=5000,
                        max_col_norm=1.)

l2 = mlp.RectifiedLinear(layer_name='l2',
                        sparse_init=12,
                        dim=5000,
                        max_col_norm=1.)

l3 = mlp.RectifiedLinear(layer_name='l3',
                        sparse_init=12,
                        dim=5000,
                        max_col_norm=1.)

output = mlp.Softmax(layer_name='y',
                     n_classes=2,
                     irange=.005)

layers = [l1, l2, l3, output]

mdl = mlp.MLP(layers,
              input_space=in_space)

lr = .0001
epochs = 100
trainer = sgd.SGD(learning_rate=lr,
                  batch_size=128,
                  learning_rule=learning_rule.Momentum(.5),
                  # Remember, default dropout is .5
                  cost=Dropout(input_include_probs={'l1': .8},
                                input_scales={'l1': 1.}),
                  termination_criterion=EpochCounter(epochs),
                  monitoring_dataset={'train': full})
```
```python
# no sklearn.cross_validation > train_test_split
# own test/train split so we can also link filenames
splitter = round(len(x)*0.8)
X_train, X_test = x[:splitter], x[splitter:]
y_train, y_test = y[:splitter], y[splitter:]
filenames_train, filenames_test = filenames[:splitter], filenames[splitter:]
pickle.dump(X_train, open('saved_feat_x_train_v2.pkl', 'wb'))
pickle.dump(X_test, open('saved_feat_x_test_v2.pkl', 'wb'))
pickle.dump(y_train, open('saved_feat_y_train_v2.pkl', 'wb'))
pickle.dump(y_test, open('saved_feat_y_test_v2.pkl', 'wb'))
pickle.dump(filenames_train, open('saved_feat_filenames_train_v2.pkl', 'wb'))
pickle.dump(filenames_test, open('saved_feat_filenames_test_v2.pkl', 'wb'))
```
#start our MLP (pylearn experiment method)

def liftoff():
    trn = DenseDesignMatrix(X=X_train, y=y_train)
    tst = DenseDesignMatrix(X=X_test, y=y_test)
    trainer.monitoring_dataset = {'valid': tst, 'train': trn}
    experiment.main_loop()

In [*]:

    trn = DenseDesignMatrix(X=X_train, y=y_train)
    tst = DenseDesignMatrix(X=X_test, y=y_test)
    trainer.monitoring_dataset = {'valid': tst, 'train': trn}
    experiment.main_loop()

Parameter and initial learning rate summary:
- l1_W: 9.99999974738e-05
- l1_b: 9.99999974738e-05
- l2_W: 9.99999974738e-05
- l2_b: 9.99999974738e-05
- l3_W: 9.99999974738e-05
- l3_b: 9.99999974738e-05
- softmax_b: 9.99999974738e-05
- softmax_W: 9.99999974738e-05

Compiling sgd_update...
Compiling sgd_update done. Time elapsed: 1.686666 seconds
compiling begin_record_entry...
compiling begin_record_entry done. Time elapsed: 0.548132 seconds
Monitored channels:
- learning_rate
- momentum
- total_seconds_last_epoch

(...)

already after 5 min: valid_y_misclass: 0.06199999989867
In [319]:
plt.plot(score_train[2:])
plt.plot(score_test[2:])

Out[319]: [<matplotlib.lines.Line2D at 0x7f517c694450>]

accuracy

50%

90%

1 hour
In [319]:
    plt.plot(score_train[2:])
    plt.plot(score_test[2:])
Out[319]:
    [<matplotlib.lines.Line2D at 0x7f517c694450>]

94%

97%

accuracy

start at iteration #2
1 hour

(...)

1 hour
In [380]:
input_image = caffe.io.load_image('google-glasses-cat-2.jpg')
plt.imshow(input_image)

Out[380]:
<matplotlib.image.AxesImage at 0x7f519a017d410>

In [381]:
feat = getfeat_single_image('google-glasses-cat-2.jpg')  # run image through cnn
x = feat
y = f([x])  # run feature through DBN > out: prediction
if y:
    print "WOOF!"
else:
    print "MEOW!"

MEOW!
So are YOU more like a Dog or Cat?

CAT VS DOG

DETECTOR
I might put it up as a Flask site online, if people are interested?

What about me?

(CAT or DOG, that's the question...)
In [391]:

```
wget http://i.imgur.com/oMJyDO0.jpg
```

```
--2015-05-12 12:12:00--  http://i.imgur.com/oMJyDO0.jpg
Resolving i.imgur.com (i.imgur.com)... 199.27.76.193
Connecting to i.imgur.com (i.imgur.com)|199.27.76.193|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 58456 (57K) [image/jpeg]
Saving to: 'oMJyDO0.jpg'

100%[============================================================>] 58,456  --.-K/s  in 0.02s

2015-05-12 12:12:00 (2.97 MB/s) - 'oMJyDO0.jpg' saved [58456/58456]

In [393]:

```
input_image = caffe.io.load_image('oMJyDO0.jpg')
plt.imshow(input_image)
```

Out[393]: `<matplotlib.image.AxesImage at 0x7f51ae6e5e50>`
feat = getfeat_single_image('rQ4bKra.jpg')  # run image to get feature
x = feat
y = f([x])  # run feature through DBN > out: prediction
if y:
    print "WOOF I'm a Dog!"
else:
    print "MEOW I'm a Cat!"

WOOF I'm a Dog!
```
In [395]:
feat = getfeat_single_image('rQ4bKra.jpg')  # run image
x = feat
y = f([x])  # run feature through DBN > out: prediction
if y:
    print "WOOF I'm a Dog!"
else:
    print "MEOW I'm a Cat!"
```

WOOF I'm a Dog!
THAT'S ALL!

EASY PIEZY...
as PhD candidate KTH/CSC: “Always interested in discussing Machine Learning, Deep Architectures, Graphs, and Language Technology”

roelof@kth.se
www.csc.kth.se/~roelof/

roelof@graph-systems.com
www.graph-technologies.com
Welcome!

SCHEDULE A NEW MEETUP

Upcoming  Past  Calendar

April 20 · 6:00 PM
Deep Learning for Bioinformatics

103 Datamaniacs | 🌟🌟🌟 | 40 Photos

Agenda: • 18:00 - 18:15 Grab a coffee/beer and get ready to rumble... • 18:15 - 18:30 Welcome • 18:30 - 19:00 Roelof Pieters: Deep Learning, a birds eye view • 19:00 ~... LEARN MORE

March 10 · 6:00 PM
Kickoff! Deep Learning: Revolution or Hype in Data-Science?

144 Datamaniacs | 🌟🌟🌟 | 35 Photos

Deep Learning is kicking off everywhere (see this front page article in the New York Times for example)! There is good reason to be excited about deep learning, as it's... LEARN MORE
Wanna Play? General Deep Learning

• Theano - CPU/GPU symbolic expression compiler in python (from LISA lab at University of Montreal).  
  http://deeplearning.net/software/theano/

• Pylearn2 - library designed to make machine learning research easy.  
  http://deeplearning.net/software/pylearn2/

• Torch - Matlab-like environment for state-of-the-art machine learning algorithms in lua (from Ronan Collobert, Clement Farabet and Koray Kavukcuoglu)  
  http://torch.ch/

• more info:  http://deeplearning.net/software links/
Wanna Play? NLP

• RNNLM (Mikolov)
  http://rnnlm.org

• NB-SVM
  https://github.com/mesnilgr/nbsvm

• Word2Vec (skipgrams/cbow)
  https://code.google.com/p/word2vec/ (original)
  http://radimrehurek.com/gensim/models/word2vec.html (python)

• GloVe
  http://nlp.stanford.edu/projects/glove/ (original)
  https://github.com/maciejkula/glove-python (python)

• Socher et al / Stanford RNN Sentiment code:
  http://nlp.stanford.edu/sentiment/code.html

• Deep Learning without Magic Tutorial:
  http://nlp.stanford.edu/courses/NAACL2013/
Wanna Play ? Computer Vision

- cuda-convnet2 (Alex Krizhevsky, Toronto) (c++/CUDA, optimized for GTX 580)
  https://code.google.com/p/cuda-convnet2/

- Caffe (Berkeley) (Cuda/OpenCL, Theano, Python)
  http://caffe.berkeleyvision.org/

- OverFeat (NYU)
  http://cilvr.nyu.edu/doku.php?id=code:start