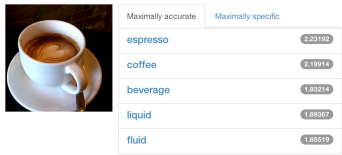
New slides from our ECCV tutorial is out!
Please go to the new slides:

https://docs.google.com/presentation/d/1UeKXVgRvvxg9OUdh_UiC5G71UMscNPlvArsWER41PsU/edit#slide=id.p



Brewing Deep Networks With Caffe

Yangqing Jia

Look for additional notes for the slides

What is, and Why Caffe?

- Pure C++/CUDA Implementation
- Fast, well-tested code
- Tools, demos, and recipes
- Seamless switch between CPU and GPU
  - `Caffe::set_mode(Caffe::GPU);`

Prototype

Training

Deployment

Statistics...

- Speed with Krizhevsky's 2012 model:
  - K40 / Titan: 2 ms/image, K20: 2.6ms
  - (40 million images / day)
  - 8-core CPU: ~20 ms/image
- ~ 8K lines of C/C++ code
  - with unit test: ~14k

* Not counting image I/O time. Details at http://caffe.berkeleyvision.org/performance_hardware.html

Do I want Caffe If I...

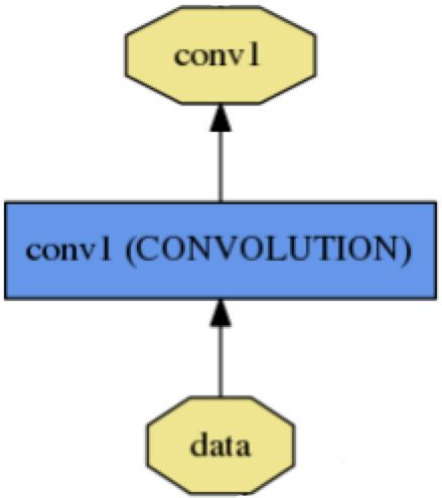
- Have small or medium scale applications?
  - Scripting languages may save engineering time indeed.
- Prefer simpler scripting languages?
  - We now provide Python and Matlab wrappers.
- Hate tricky compilation issues?
  - Recipes on Caffe webpage, and github.
  - Virtualbox / EC2 images to be provided soon.

A Caffe Layer

```
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

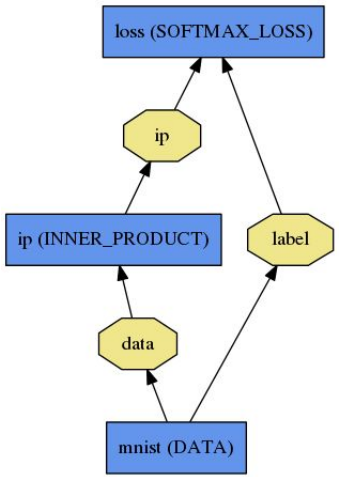


*Link to the Google Protobuffer Documentation

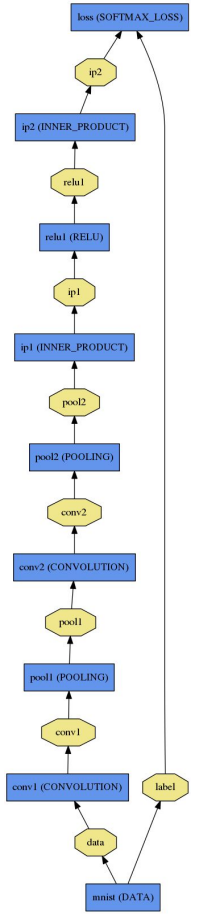
A Caffe Network

- 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" …}
```



LogReg ↑



LeNet →

ImageNet, Krizhevsky 2012 →

Training a Caffe Net

Write a solver protobuffer:

```
train_net: "lenet_train.prototxt"
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
max_iter: 10000
snapshot_prefix: "lenet_snapshot"
solver_mode: GPU
```

All you need to run things on the GPU.

End to End Recipe...

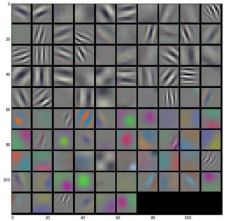
- Convert the data to Caffe-format
  - leveldb, hdf5/.mat, list of images, LMDB, etc.
- Write a Network Definition
- Write a Solver Protobuffer text
- Train with the provided train_net tool
  - build/tools/train_net.bin solver.prototxt

- Examples are your friends
  - `caffe/examples/mnist,cifar10,imagenet`
  - `caffe/tools/*.bin`

Peeking into Networks

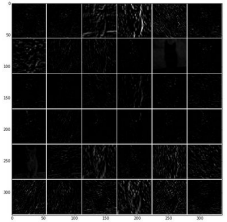
The first layer filters, conv1

In [8]:
```
# the parameters are a list of [weights, biases]
filters = net.params['conv1'][0].data
vis_square(filters.transpose(0, 2, 3, 1))
```

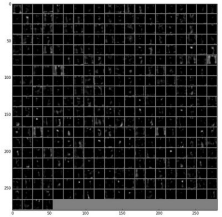


The first layer output, conv1 (rectified responses of the filters above, first 36 only)

In [9]:
```
feat = net.blobs['conv1'].data[4, :36]
vis_square(feat, padval=1)
```

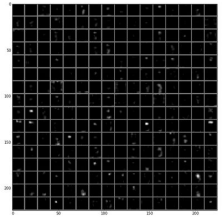


In [13]:
```
feat = net.blobs['conv4'].data[4]
vis_square(feat, padval=0.5)
```



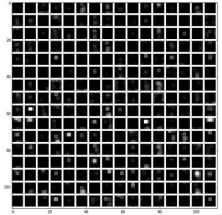
The fifth layer output, conv5 (rectified, all 256 channels)

In [14]:
```
feat = net.blobs['conv5'].data[4]
vis_square(feat, padval=0.5)
```

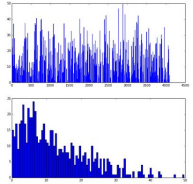


The fifth layer after pooling, pool5

In [15]:
```
feat = net.blobs['pool5'].data[4]
vis_square(feat, padval=1)
```

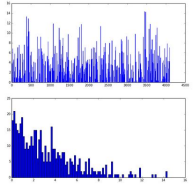


In [16]:
```
feat = net.blobs['fc6'].data[4]
plt.subplot(2, 1, 1)
plt.plot(feat.flat)
plt.subplot(2, 1, 2)
_ = plt.hist(feat.flat[feat.flat > 0], bins=100)
```



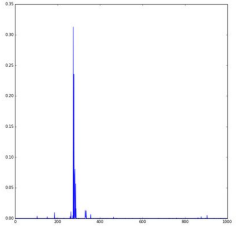
The second fully connected layer, fc7 (rectified)

In [17]:
```
feat = net.blobs['fc7'].data[4]
plt.subplot(2, 1, 1)
plt.plot(feat.flat)
plt.subplot(2, 1, 2)
_ = plt.hist(feat.flat[feat.flat > 0], bins=100)
```



In [18]:
```
feat = net.blobs['prob'].data[4]
plt.plot(feat.flat)
```

Out[18]: [<matplotlib.lines.Line2D at 0x12b260710>]



A Quick Sip of Brewed Models

http://demo.caffe.berkeleyvision.org/

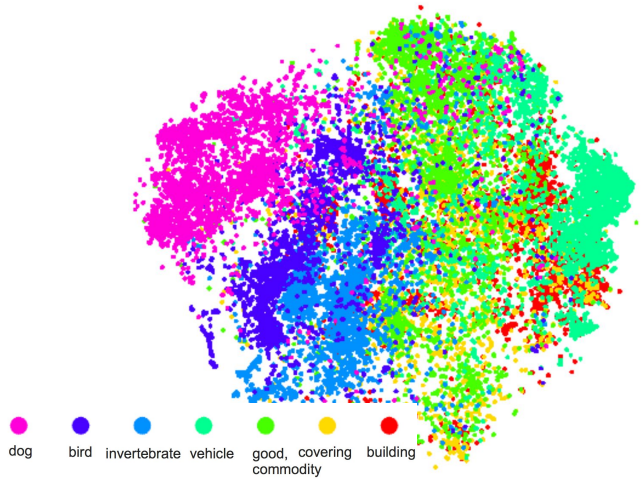
(demo code to be open-sourced soon)

Maximally accurate | Maximally specific

cat	1.80727
domestic cat	1.74727
feline	1.72787
tabby	0.99133
domestic animal	0.78542

Transfer Learned Knowledge

- Taking a pre-trained model and finetune it for related tasks [Zeiler-Fergus] [DeCAF] [OverFeat]



*Dog and cat image Copyright kaggle.com

Dogs vs Cats: top 10% in 10 minutes

- Simply change a few lines in the layer definition

```
layers {
  name: "data"
  type: DATA
  data_param {
    source: "ilsvrc12_train_leveldb"
    mean_file: "../../data/ilsvrc12/i
    batch_size: 256
    crop_size: 227
    mirror: true
  }
```

```
layers {
  name: "data"
  type: DATA
  data_param {
    source: "dogs-vs-cats-leveldb"
    mean_file: "../../data/ilsvrc1
    batch_size: 256
    crop_size: 227
    mirror: true
  }
```

Input:
A different source

```
layers {
  name: "fc8"
  type: INNER_PRODUCT
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 1000
```

```
layers {
  name: "fc8-dogcat"
  type: INNER_PRODUCT
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
  inner_product_param {
    num_output: 2
```

Last Layer:
A different classifier

Dogs vs Cats: top 10% in 10 minutes

build/tools/finetune_net.bin dogcat_solver.prototxt pretrained_imagenet_model

Under the hood (loosely speaking):

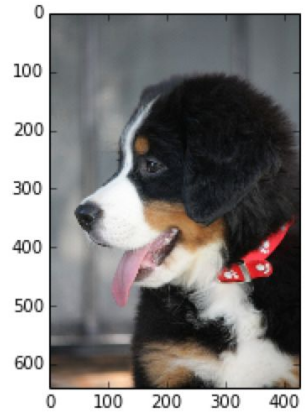
```
net = new Caffe::Net(
    "dogcat_solver.prototxt");
net.CopyTrainedNetFrom(
    pretrained_model);
solver.Solve(net);
```

Example code to be made available at

caffe/examples/dogs-vs-cats/

```
plt.imshow(image)
scores = net.predict([image]).flatten()
if scores[1] > scores[0]:
    print 'Woof it is a DOG!'
else:
    print 'Yiss it is a CAT!'
```

```
Woof it is a DOG!
```



Object Detection

R-CNN: Regions with Convolutional Neural Networks

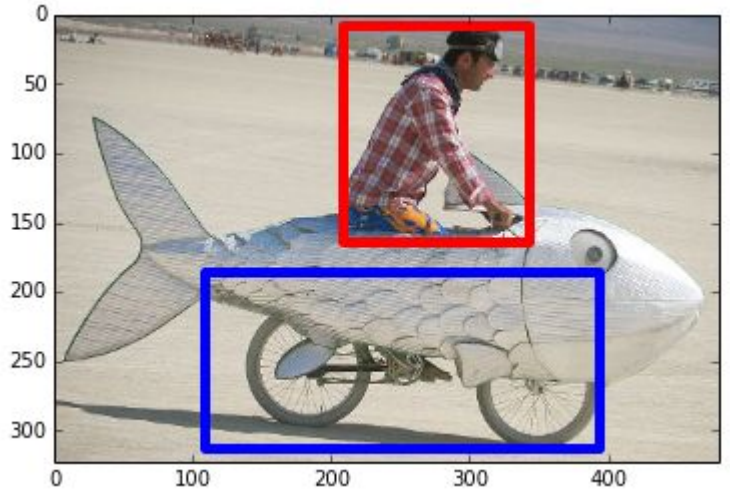
http://nbviewer.ipython.org/github/BVLC/caffe/blob/dev/examples/detection.ipynb

Full R-CNN scripts available at

https://github.com/rbgirshick/rcnn

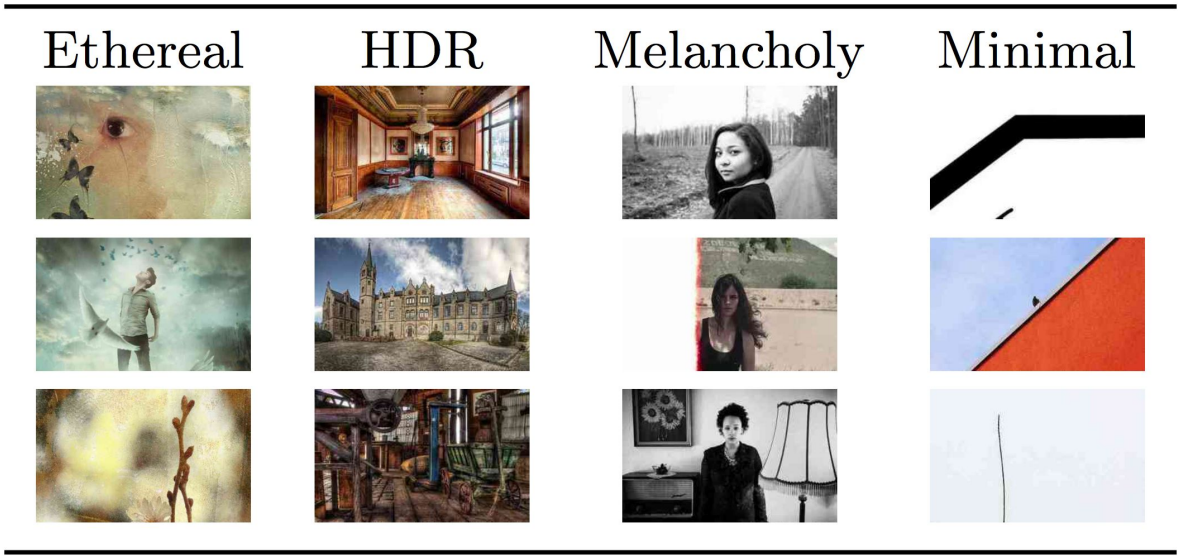
Ross Girshick et al.
Rich feature hierarchies for accurate object detection and semantic segmentation

Oral Session 2A, Tue 1:30 pm



Visual Style Recognition

Sergey Karayev, http://vislab.berkeleyvision.org/, demo available online



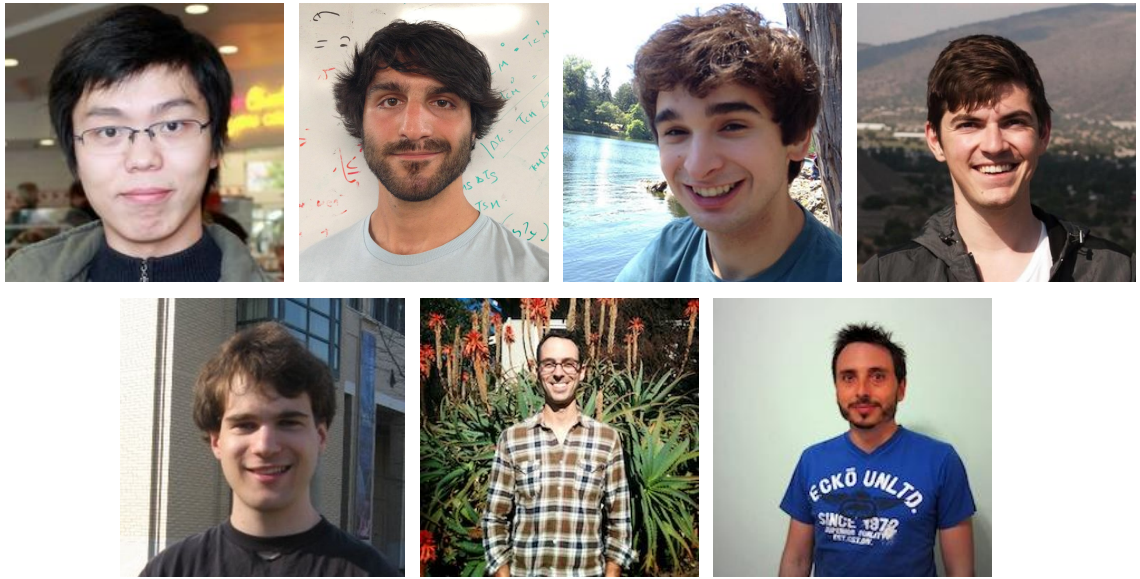
Other Styles:

Vintage
Long Exposure
Noir
Pastel
Macro
… and so on.

In One Sip

Caffe...

- is C++/CUDA friendly
- is fast
- is state-of-the-art
- has tips, recipes, demos
- all available under an open-source initiative



Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev
Jonathan Long, Ross Girshick, Sergio Guadarrama

https://github.com/UCB-ICSI-Vision-Group/caffe-paper/raw/arxiv/caffe.pdf