What is Caffe?
Open Source Framework for Deep Learning

- Developed by the Berkeley Vision and Learning Center (BVLC)
- C++/CUDA architecture
- Command line, Python, MATLAB interfaces
- Pre-processing and deployment tools, reference models and examples
- Image data management
- Seamless GPU acceleration
- Large community of contributors to the open-source project

http://github.com/BVLC/caffe
What is Caffe?
End-to-end Deep Learning for the Practitioner and Developer

Prototype

Train

Deploy
Caffe Features
Data Pre-processing and Management

Data ingest formats
- LevelDB or LMDB database
- In-memory (C++ and Python only)
- HDF5
- Image files

Pre-processing tools
- LevelDB/LMDB creation from raw images
- Training and validation set creation with shuffling
- Mean-image generation

Data transformations
- Image cropping, resizing, scaling and mirroring
- Mean subtraction

$CAFFE_ROOT/build/tools
Caffe Features

Deep Learning Model Definition

- Protobuf model format
  - Strongly typed format
  - Human readable
  - Auto-generates and checks Caffe code
  - Developed by Google
  - Used to define network architecture and training parameters
  - No coding required!

```python
name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"
convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
        type: "xavier"
    }
}
```
Caffe Features
Deep Learning Model Definition

- Loss functions:
  - Classification
    - Softmax
    - Hinge loss
  - Linear regression
    - Euclidean loss
  - Attributes/multiclassification
    - Sigmoid cross entropy loss
  - and more...

- Available layer types:
  - Convolution
  - Pooling
  - Normalization

- Activation functions:
  - ReLU
  - Sigmoid
  - Tanh
  - and more...
Network training also requires no coding - just define a “solver” file

net: “lenet_train.prototxt”
base_lr: 0.01
momentum: 0.9
max_iter: 10000
snapshot_prefix: “lenet_snapshot”
solver_mode: GPU

> caffe train –solver lenet_solver.prototxt –gpu 0

Multiple optimization algorithms available: SGD (+momentum), ADAGRAD, NAG
Caffe Features

Monitoring the Training Process

Output to stdout:

To visualize - pipe, parse and plot or use DIGITS
Caffe Features

Deep Neural Network Deployment

Standard, compact model format

    caffe train produces a binary .caffemodel file

Easily integrate trained models into data pipelines

    Deploy against new data using command line, Python or MATLAB interfaces

Deploy models across HW and OS environments

    .caffemodel files transfer to any other Caffe installation (including DIGITS)
Caffe Features

Deep Neural Network Sharing

Caffe Model Zoo hosts community shared models

Benefit from networks that you could not practically train yourself

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

Caffe comes with unrestricted use of BVLC models:

- AlexNet
- R-CNN
- GoogLeNet
Caffe Features

Extensible Code

```python
import caffe
import numpy as np

class EuclideanLoss(caffe.Layer):
    def setup(self, bottom, top):
        # check input pair
        if len(bottom) != 2:
            raise Exception("Need two inputs to compute distance.")

    def reshape(self, self, bottom, top):
        # check input dimensions match
        if bottom[0].count != bottom[1].count:
            raise Exception("Inputs must have the same dimension.")
        # difference is shape of inputs
        self.diff = np.zeros_like(bottom[0].data, dtype=np.float32)
        # loss output is scalar
        top[0].reshape(1)

    def forward(self, self, bottom, top):
        self.diff[:] = bottom[0].data - bottom[1].data
        top[0].data[...] = np.sum(self.diff**2) / bottom[0].num / 2.

    def backward(self, self, top, propagate_down, bottom):
        for i in range(2):
            if not propagate_down[i]:
                continue
            if i == 0:
                sign = 1
            else:
                sign = -1
            bottom[i].diff[:] = sign * self.diff / bottom[i].num
```

Layer Protocol == Class Interface

Define a class in C++ or Python to extend Layer
Include your new layer in a network prototxt

```protobuf
layer {
  type: "Python"
  python_param {  
    module: "layers"
    layer: "EuclideanLoss"
  }
}
```
Caffe Features

Extend Caffe from C++

Start from the caffe::Layer class

Implement the member functions

LayerSetUp
Reshape
Forward_cpu/Forward_gpu
Backward_cpu/Backward_gpu
Caffe Features

Extend Caffe from C++

Write the CUDA kernel

```
template <typename Dtype>
void MaxPoolForward(const int nthreads, const Dtype* bottom_data,
                     const int num, const int channels, const int height, const int width,
                     const int pooled_height, const int pooled_width,
                     const int kernel_h, const int kernel_w, const int stride_h, const int stride_w,
                     const int pad_h, const int pad_w, Dtype* top_data,
                     int* mask, Dtype* top_mask) {

  int pw = index % pooled_width;
  int ph = (index / pooled_width) % pooled_height;
  int c = (index / pooled_width / pooled_height) % channels;
  int n = index / pooled_width / pooled_height / channels;
  int hstart = ph * stride_h - pad_h;
  int wstart = pw * stride_w - pad_w;
  int hend = min(hstart + kernel_h, height);
  int wend = min(wstart + kernel_w, width);
  hstart = max(hstart, 0);
  wstart = max(wstart, 0);
  Dtype maxval = -FLT_MAX;
  int maxidx = -1;
  bottom_data[0] = channels + c) * height = width;
  for (int h = hstart; h < hend; ++h) {
    for (int w = wstart; w < wend; ++w) {
      if (bottom_data[h * width + w] > maxval) {
        maxidx = h * width + w;
        maxval = bottom_data[maxidx];
      }
    }
  }
  top_data[index] = maxval;
  if (mask) {
    mask[index] = maxidx;
  } else {
    top_mask[index] = maxidx;
  }
}
```

Forward_gpu launches the CUDA kernel

```
template <typename Dtype>
void PoolingLayer<Dtype>::Forward_gpu(const vector< Blob<Dtype>* >& bottom,
                                       const vector< Blob<Dtype>* >& top) {
  const Dtype* bottom_data = bottom[0]->gpu_data();
  Dtype* top_data = top[0]->mutable_gpu_data();
  int count = top[0]->count();
  // We'll output the mask to top[1] if it's of size > 1.
  const bool use_top_mask = top.size() > 1;
  int* mask = NULL;
  Dtype* top_mask = NULL;
  switch (this->layer_param_.pooling_param().pool()) {
    case PoolingParameter_PoolMethod_MAX:
      if (use_top_mask) {
        top_mask = top[1]->mutable_gpu_data();
      } else {
        mask = max_idx_.mutable_gpu_data();
      }
      // NOLINTNEXTLINE(whitespace/indents)
      MaxPoolForward<Dtype>::<<<CAFFE_GET_BLOCKS(count), CAFFE_CUDA_NUM_THREADS>>>(
        count, bottom_data, bottom[0]->num(), channels,
        height, width, pooled_height, pooled_width, kernel_h,
        kernel_w, stride_h, stride_w, pad_h, pad_w, top_data,
        mask, top_mask);
      break;
  }
```
Caffe Features

Extend Caffe from C++

Extend the Protobuffer file

```protobuf
text
```
Example applications

Use Case 1: Classification of Images

Object

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

Open source demo code:

$CAFFE_ROOT/examples/web_demo

Scene

http://places.csail.mit.edu/

B. Zhou et al. NIPS 14

Style

http://demo.vislab.berkeleyvision.org/

Karayev et al. Recognizing Image Style. BMVC14
Example applications

Use Case 2: Localization

(Fast) Region based Convolutional Networks (R-CNN)
Ross Girshick, Microsoft Research
https://github.com/rbgirshick/fast-rcnn
Example applications

Use Case 3: Pixel Level Classification and Segmentation

http://fcn.berkeleyvision.org
Example applications

Use Case 4: Sequence Learning

- Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTM)
  - Video
  - Language
  - Dynamic data

- Current Caffe pull request to add support
  - https://github.com/BVLC/caffe/pull/1873
  - http://arxiv.org/abs/1411.4389

A group of young men playing a game of soccer.

Jeff Donahue et al.
Example applications

Use Case 5: Transfer Learning

Lots of data

ImageNet

CNN

New data

Transfer weights

Object Classifier

Dog vs. Cat

Just change a few lines in the model prototxt file

```
layer {
  name: "data"
  type: "Data"
  data_param {
    source: "ilsvrc12_train"
  }
}
...}
...
...
layer {
  name: "fc8"
  type: "InnerProduct"
  inner_product_param {
    num_output: 1000
  }
}
```
Caffe Setup and Performance
Caffe Setup

- Tried and tested by BVLC on Ubuntu 14.04/12.04 and OS X 10.8+
- Also demonstrated to compile on RHEL, Fedora and CentOS
- Download source from https://github.com/BVLC/caffe
- Unofficial 64-bit Windows port https://github.com/niuzhiheng/caffe
- Linux setup (see http://caffe.berkeleyvision.org/installation.html)
  - Download
  - Install pre-requisites
  - Install CUDA and cuDNN for GPU acceleration
  - Compile using make
GPU Acceleration

-gpu N flag tells caffe which gpu to use

Alternatively, specify solver_mode: GPU in solver.prototxt

Benchmark: Train Caffenet model, 20 iterations, 256x256 images, mini-batch size 256
cuDNN integration
http://developer.nvidia.com/cudnn

Drop-in support

Install cuDNN, uncomment `USE_CUDNN :=1` in `Makefile.config` before build
Caffe Mobile Deployment

- Jetson TX1
  - Up to 340 img/s on Alexnet (FP16)
  - No need to change code
- Compile Caffe
- Copy a trained .caffemodel to TX1
- Run

*Source: http://petewarden.com/2014/10/25/how-to-run-the-caffe-deep-learning-vision-library-on-nvidias-jetson-mobile-gpu-board/*
Hands-on lab Preview

bit.ly/dlnvlab3

- Use data pre-processing tools
- Edit a network definition
- Train a model
- Improve classification accuracy by modifying network parameters
- Visualize trained network weights
- Deploy a model using Python
Hands-on Lab

1. Create an account at nvidia.qwiklab.com
2. Go to “LHC 2015 Workshop” lab
3. Start the lab and enjoy!

Only requires a supported browser, no NVIDIA GPU necessary!