

Part 1: Convolutional neural networks in caffe

Part 2: CNNs for retinal vessels segmentation

Mitko Veta

IMAG/e, Eindhoven University of Technology

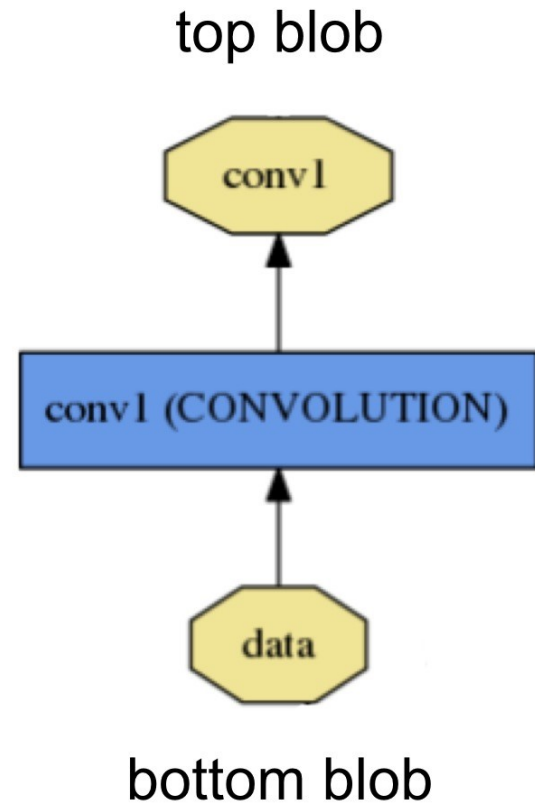


Caffe

- “Caffe is a deep learning framework made with expression, speed, and modularity in mind.”
- Developed by Berkeley Vision and Learning Center (BVLC)
- C++ with Python and MATLAB wrappers
- Open source
 - <https://github.com/BVLC/caffe>

Caffe

- Modular design
- Data is passed as “blobs”
- Most layer types have one bottom (input) and one top (output blob)

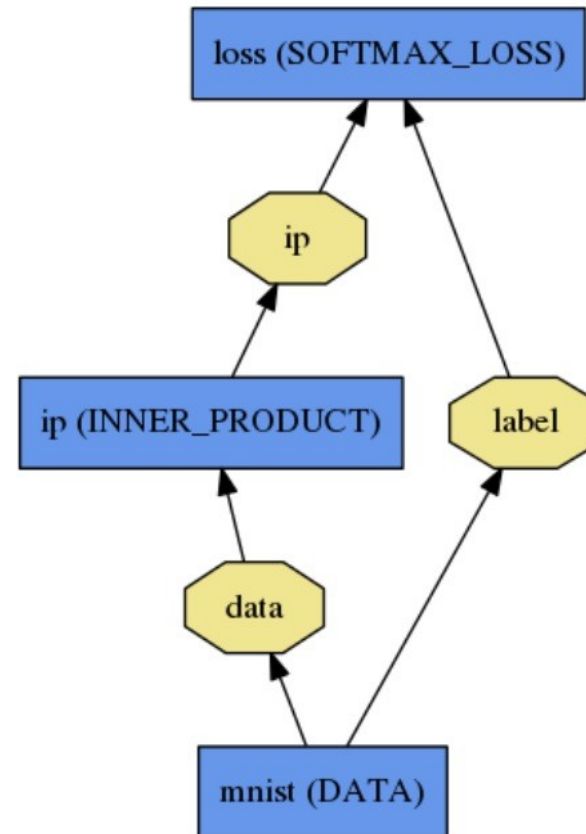


Caffe

- Two configurations need to be defined:
 - Network architecture
 - Solver
 - Both as protocol buffers
 - Both configurations can be put in a single .prototxt file

Logistic regression example

```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 2
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip"
  bottom: "label"
  top: "loss"
}
```



Optimization

```
solver_mode: GPU  
solver_type: NESTEROV
```

```
base_lr: 0.01  
momentum: 0.9  
weight_decay: 0.0001
```

```
lr_policy: "fixed"
```

```
max_iter: 200000
```

```
snapshot: 1000  
snapshot_prefix: "path/to/output/folder"
```

```
test_iter: 100  
test_interval: 1000
```

```
display: 100  
average_loss: 100
```

- The solver defines the optimization method and parameters
 - Default is SGD

Input

```
layer {  
  name: "images"  
  type: "Data"  
  top: "data"  
  top: "label"  
  transform_param {  
    mirror: true  
    mean_value: 111  
    scale: 0.015  
  }  
  data_param {  
    source: "/path/to/training/training/db"  
    backend: LMDB  
    batch_size: 256  
  }  
  include: { phase: TRAIN }  
}
```

- Caffe supports several input formats
 - Databases (LevelDB, LMDB)
 - Files on disk (HDF5, common image formats)
- The data layer handles the reading of the input in batches and basic preprocessing (scaling, mean subtraction)

Convolutional layer

```
layer {  
  name: "conv1"  
  type: "Convolution"  
  bottom: "data"  
  top: "conv1"  
  param {  
    lr_mult: 1  
    decay_mult: 1  
  }  
  param {  
    lr_mult: 2  
    decay_mult: 0  
  }  
  convolution_param {  
    num_output: 48  
    kernel_size: 6  
    stride: 1  
    weight_filler {  
      type: "xavier"  
    }  
    bias_filler {  
      type: "constant"  
    }  
  }  
}
```

- The number of inputs is implicitly defined with the bottom blob
- `weight_filler` and `bias_filler` define the initialization method
 - Weights and biases can have different learning rates and regularization strengths

Nonlinearity and max-pooling

```
layer {  
  name: "nonlin1"  
  type: "TanH"  
  bottom: "conv1"  
  top: "conv1"  
}
```

```
layer {  
  name: "pool1"  
  type: "Pooling"  
  bottom: "conv1"  
  top: "pool1"  
  pooling_param {  
    pool: MAX  
    kernel_size: 2  
    stride: 2  
  }  
}
```

- The nonlinearity can operate “in place”
 - Saves memory
- No trainable parameters → no initialization needed

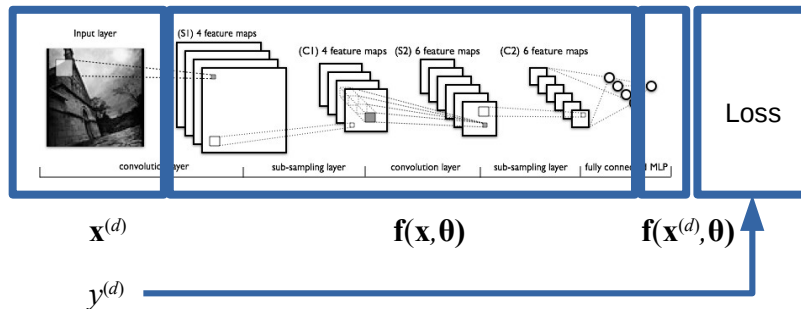
Fully connected layer

```
layer {  
  name: "ip1"  
  type: "InnerProduct"  
  bottom: "pool4"  
  top: "ip1"  
  param {  
    lr_mult: 1  
    decay_mult: 1  
  }  
  param {  
    lr_mult: 2  
    decay_mult: 0  
  }  
  inner_product_param {  
    num_output: 100  
    weight_filler {  
      type: "xavier"  
    }  
    bias_filler {  
      type: "constant"  
    }  
  }  
}
```

- Called “inner product” for obvious reasons
- The initialization is defined in a similar manner to convolutional layers

Loss layer

```
layer {  
  name: "loss"  
  type: "SoftmaxWithLoss"  
  bottom: "ip2"  
  bottom: "label"  
  top: "loss"  
}
```



- Softmax and loss layers are combined in one layer because this way the “gradient computation is more numerically stable”
- At test time, this can be replaced by a softmax layer

Deployment

- Caffe outputs two types of files
 - .caffemodel: Contains the weights of the model
 - This is what you deploy!
 - .solverstate: Snapshot of the solver, can be used to resume training
- Separate deployment .prototxt needs to be defined

Running caffe

```
$ caffe.bin train -solver solver.prototxt
```

```
...
```

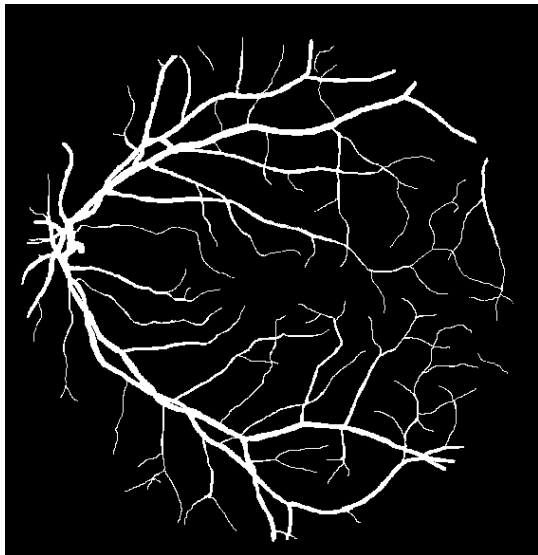
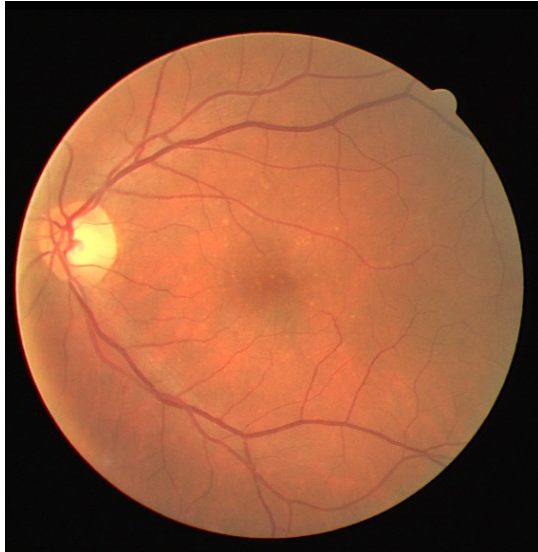
```
I0911 13:31:32.394986 5078 layer_factory.hpp:74] Creating layer images
I0911 13:31:32.395014 5078 net.cpp:90] Creating Layer images
I0911 13:31:32.395026 5078 net.cpp:368] images -> data
I0911 13:31:32.395061 5078 net.cpp:368] images -> label
I0911 13:31:32.395078 5078 net.cpp:120] Setting up images
I0911 13:31:32.395531 5078 db_lmdb.cpp:22] Opened lmdb db/training
I0911 13:31:32.395733 5078 data_layer.cpp:52] output data size: 256,1,65,65
I0911 13:31:32.396944 5078 net.cpp:127] Top shape: 256 1 65 65 (1081600)
I0911 13:31:32.396968 5078 net.cpp:127] Top shape: 256 (256)
I0911 13:31:32.396981 5078 layer_factory.hpp:74] Creating layer conv1
I0911 13:31:32.397017 5078 net.cpp:90] Creating Layer conv1
I0911 13:31:32.397027 5078 net.cpp:410] conv1 <- data
I0911 13:31:32.397049 5078 net.cpp:368] conv1 -> conv1
I0911 13:31:32.397069 5078 net.cpp:120] Setting up conv1
I0911 13:31:32.441848 5078 net.cpp:127] Top shape: 256 48 60 60 (44236800)
I0911 13:31:32.441915 5078 layer_factory.hpp:74] Creating layer nonlin1
I0911 13:31:32.441941 5078 net.cpp:90] Creating Layer nonlin1
I0911 13:31:32.441956 5078 net.cpp:410] nonlin1 <- conv1
I0911 13:31:32.441977 5078 net.cpp:357] nonlin1 -> conv1 (in-place)
I0911 13:31:32.441994 5078 net.cpp:120] Setting up nonlin1
I0911 13:31:32.442090 5078 net.cpp:127] Top shape: 256 48 60 60 (44236800)
```

```
...
```

Running caffe

```
...
I0911 13:31:32.453774 5078 solver.cpp:294] Iteration 0, Testing net (#0)
I0911 13:31:36.487558 5078 solver.cpp:343] Test net output #0: accuracy = 0.463281
I0911 13:31:36.487620 5078 solver.cpp:343] Test net output #1: loss = 0.699408 (* 1 = 0.699408 loss)
I0911 13:31:36.531695 5078 solver.cpp:214] Iteration 0, loss = 0.681241
I0911 13:31:36.531749 5078 solver.cpp:229] Train net output #0: loss = 0.681241 (* 1 = 0.681241 loss)
I0911 13:31:36.531764 5078 solver.cpp:486] Iteration 0, lr = 0.01
I0911 13:31:50.624618 5078 solver.cpp:214] Iteration 100, loss = 0.37712
I0911 13:31:50.624673 5078 solver.cpp:229] Train net output #0: loss = 0.408127 (* 1 = 0.408127 loss)
I0911 13:31:50.624686 5078 solver.cpp:486] Iteration 100, lr = 0.01
I0911 13:32:04.677295 5078 solver.cpp:214] Iteration 200, loss = 0.315125
I0911 13:32:04.677389 5078 solver.cpp:229] Train net output #0: loss = 0.278361 (* 1 = 0.278361 loss)
I0911 13:32:04.677403 5078 solver.cpp:486] Iteration 200, lr = 0.01
...
I0911 13:33:57.053082 5078 solver.cpp:361] Snapshotting to models/nfbia_iter_1000.caffemodel
I0911 13:33:57.055656 5078 solver.cpp:369] Snapshotting solver state to models/nfbia_iter_1000.solverstate
I0911 13:33:57.056545 5078 solver.cpp:294] Iteration 1000, Testing net (#0)
I0911 13:34:01.060945 5078 solver.cpp:343] Test net output #0: accuracy = 0.922148
I0911 13:34:01.061064 5078 solver.cpp:343] Test net output #1: loss = 0.206845 (* 1 = 0.206845 loss)
...
```

DRIVE



- 40 images of the retina with ground truth vessel segmentation
- 20 training, 20 testing
 - First 14 training images used for training
 - Last 6 training images used for validation
- We want to do pixel classification

Preprocessing and data augmentation

- Preprocessing
 - Average the 3 RGB color channels to obtain a grayscale image
 - Retina mask
- Data augmentation
 - From each image sample (with replacement) 300K samples of size 65x65 pixels
 - Random rotation between 0 and 2π
 - Label “1” if the central pixel belongs to a vessel, “0” otherwise
 - 4.2M training samples, 1.8M validation samples
 - Only ~12% of the training samples are positive



DRIVE notebook

- Now, let's look at the notebook with the DRIVE example...

DRIVE notebook

- Input 65x65
- Convolution 1x6x6x48
- Hyperbolic tangent nonlinearity
- Max pooling 2x2
- Convolution 48x5x5x48
- Hyperbolic tangent nonlinearity
- Max pooling 2x2
- Convolution 48x4x4x48
- Hyperbolic tangent nonlinearity
- Max pooling 2x2
- Convolution 48x2x2x48
- Hyperbolic tangent nonlinearity
- Max pooling 2x2
- Fully connected (48x2x2)x100
- Hyperbolic tangent nonlinearity
- Fully connected 100x2
- Softmax

$$\text{output_size} = (\text{input_size} - \text{kernel_size})/\text{stride} + 1$$

$$(65-6)/1 + 1 = 60$$

$$(60-2)/2 + 1 = 30$$

$$(30-5)/1 + 1 = 26$$

$$(26-2)/2 + 1 = 13$$

$$(13-4)/1 + 1 = 10$$

$$(10-2)/2 + 1 = 5$$

$$(5-2)/1 + 1 = 4$$

$$(4-2)/2 + 1 = 2$$