Tutorial 4. MatConvNet

Xingyu ZENG(xyzeng@ee.cuhk.edu.hk)

MatConvNet

- A open-source Matlab toolbox for Convolution Network

- You can download in

  https://github.com/vlfeat/matconvnet

- *If you use MatConvNet in your work, please cite:*

  "MatConvNet - Convolutional Neural Networks for MATLAB", A. Vedaldi and K. Lenc, arXiv:1412.4564, 2014.

MatConvNet

- Advantage
  - Matlab, easy to use
  - Pretrained models(VGG, AlexNet)
  - Support GPU
- Disadvantage
  - Complicated than DeepLearnToolbox
  - Support only Convolution Network

MatConvNet

- Installing and compiling the library
  - Matlab
    - run <path to MatConvNet>/matlab/vl_setupnn
    - vl_compilenn();
    - vl_compilenn('enableGpu', true); % for gpu
  - Shell
    - make ARCH=<your arch> MATLABROOT=<path to MATLAB>
    - make ENABLE_GPU=y ARCH=<your arch> MATLABROOT=<path to MATLAB> CUDAROOT=<path to CUDA>

Notes: the gpu version requre at least MATLAB2013

MatConvNet

- Overview
  1. Core functions
     - Different type of layers, including convolution, dropout
  2. Simple CNN functions
     - A simple wrapper for CNN, including training, testing, display
  3. Utility functions
     - Some helper function to initialize the toolbox

MatConvNet

- Pretrained Models (VGG, AlexNet)

Usage:

1. Download a pre-trained CNN
2. net=load('cnn.mat');
3. im=im-net.normalization.averageImage;
4. res=vl_simplenn(net,im);

Notes:

1. res(i).x, output for i-th layer
2. Each model has one averageImage value

MatConvNet

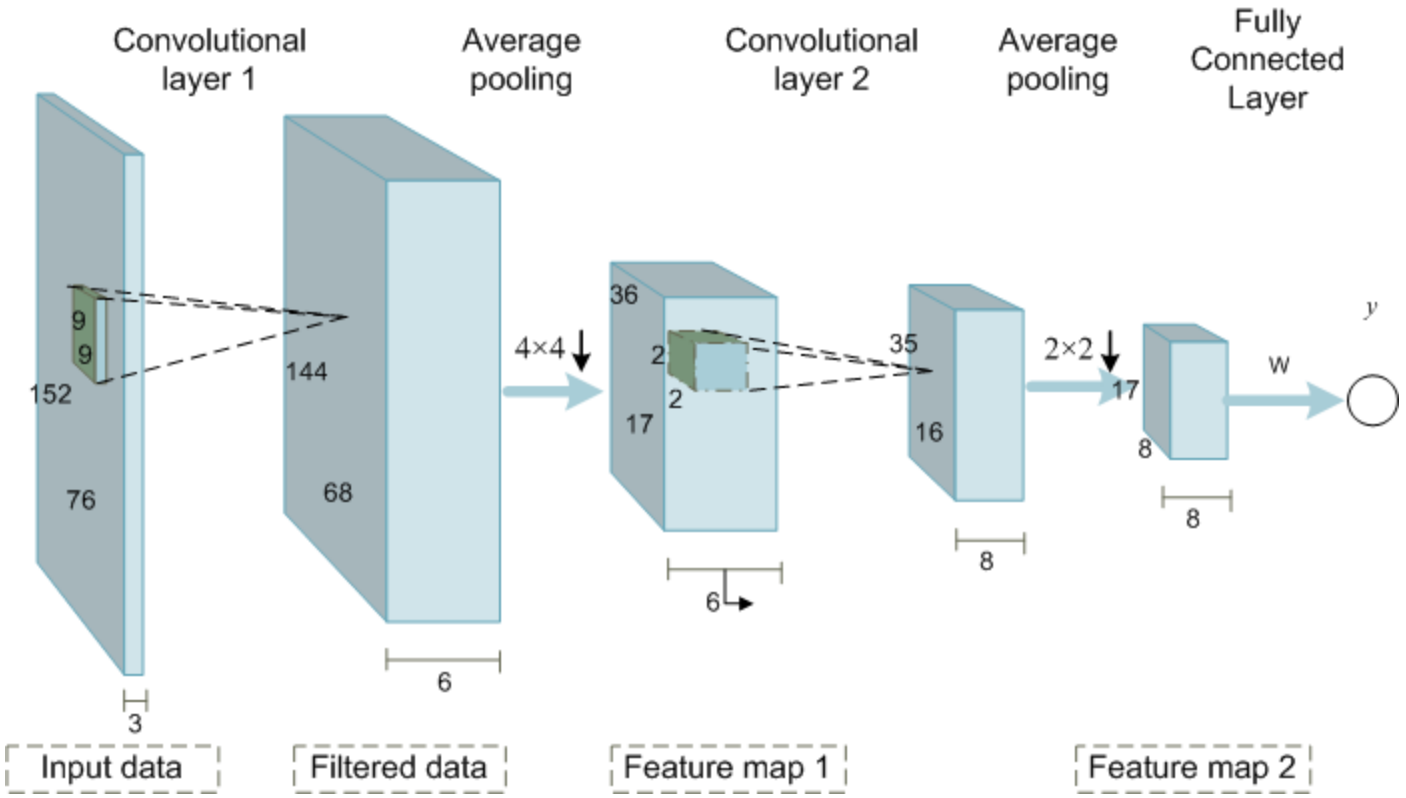
- Notes:
  1. Format: Height*Width*Channels*Num
  2. Pretrained Models:
     - VGG models

       `Very Deep Convolutional Networks for Large-Scale Image Recognition', Karen Simonyan and Andrew Zisserman, arXiv technical,report, 2014
     - AlexNet

       `ImageNet Classification with Deep Convolutional Neural Networks', Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, NIPS, 2012

MatConvNet

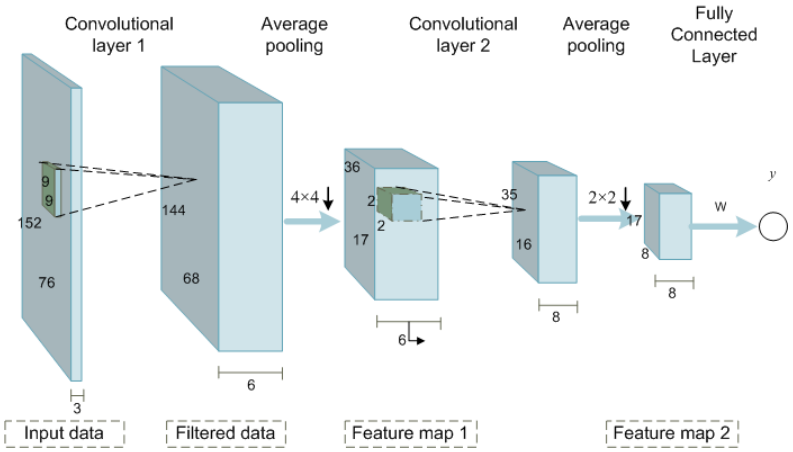
- Suppose input size: 152*76*3



MatConvNet

- Construct one CNN

```
net.layers = {} ;
net.layers{end+1} = struct('type', 'conv', ...
                           'filters', f*randn(9,9,3,6, 'single'), ...
                           'biases', zeros(1, 6, 'single'), ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [4 4], ...
                           'stride', 4, ...
                           'pad', 0) ;
……..
net.layers{end+1} = struct('type', 'softmaxloss') ;
```



MatConvNet

- Layers
  1. Convolution Layer
  2. Pooling Layer
  3. RELU Layer
  4. Dropout Layer
  5. Softmax Layer
  6. Log-loss layer
  7. Softmax-log-loss-layer
  8. Custom Layer
     1. layer.type = 'custom'
     2. layer.forward: a function handle computing the block.
     3. layer.backward: a function handle computing the block derivative.

MatConvNet

- Training one CNN

Usage:

1. opts.train.batchSize=100;
2. opts.train.numEpochs=100;
3. opts.train.useGpu=false;
4. opts.train.learningRate=0.001;
5. [net,info]=cnn_train(net,imdb,@getBatch,opts);

Notes:

1. getBatch: one function, [im,labels]=getBatch(imdb,ind);
2. info: error, objective, topFiveError….

MatConvNet

- More details can be found in

http://www.vlfeat.org/matconvnet/matconvnet-manual.pdf

- More examples can be found in

https://github.com/vlfeat/matconvnet/tree/master/examples

- The homepage of MatConvNet:

http://www.vlfeat.org/matconvnet/