2016/11/15, Tuesday, Cloudy

```
By Jincheng Su, mail: jcsu14@fudan.edu.cn
```

Keep on making `/path/to/bcnn/run/run_experiments_bcnn_train.m` run:

1. Attemptation 2.

Recall that when I compiled 'matconvnet', I simply run `> vl_compilenn` , which is only with CPU support.

By enabling GPU support, one should use

```
 > vl_compilenn('enableGpu', true)
```

or

```
 > vl_compilenn('enableGpu', true, 'cudaRoot',
'/path/to/cuda/install/directory') .
```

Howerver, the cuda8.0 which is installed by double click on '*.deb' downloaded from website just cannot be found. Maybe I should download the '*run' bin file, and give it a shot…

2. Analysis `function y = vl_nnbilinearpool(x, varargin)` for symmetric bcnn

```
function y = vl_nnbilinearpool(x, varargin)
% ------------------------------------------------------------------
-----------
% functionality:
%   implementation of the feed forward pass and backward pass of 'bilinear
pool', which
%   is a layer connected next to a pretrained CNN model (or two same model
)
%
% $x$: input feature of size [height, width, channels, batch size].
% $varargin$: $dzdy$ when in backward pass.
% $y$:
%   forward pass:
%       Self outer product of $x$.
%       For each image with size `[height, width, channels]`, firstly
```

```
%       reshape it into size `[height * width, channels]`, and then
%       compute the output
%                   $$y = \frac{1}{height * width} x^T x$$
%       which gives $y$ the size `[channels, channels]`, reshape it again
to a vector.
%   backward pass:
%       gradient of $y$ w.r.t $x$.
%       $y$ is the same size as $x$, i.e., `[height, width, channels, batc
h_size]`.
%       For each image, reshape $dzdy$ to size `[channels, channels]`
%                       reshape $x$ to size `[height * width, channels]`.
%       $dydx$ is caculated as:
%                   $$y = \frac{1}{height * width} x * dzdy$$
%       which gives $y$ the size `[height * width, channels]`,
%       Reshape $y$ to `[height, width, channels]` as output.
% ----------------------------------------------------------------------
-----------

% if the number of elements in `varargin` > 0 and the first element is not
 a string
backMode = numel(varargin) > 0 && ~isstr(varargin{1})

% if in backward mode, take out the `dzdy` in `varargin`
if backMode dzdy = varargin{1}; end

% if `x` is a `gpuArray`, it is in `gpuMode`
gpuMode = isa(x, 'gpuArray');

% unpack the size of x into height, width, number of channel, and batch si
ze
[h, w, ch, bs] = size(x);

% backward mode
if backMode
    if gpuMode
        y = gpuArray(zeros(size(x), 'single'));
    else
        y = zeros(size(x), 'single'));
    end
    % for each image / for each feature map with `ch` channels
    for b = 1:bs
        dzdy_b = reshape(dzdy(1, 1, :, b), [ch, ch]);
        a = reshape (x(:, :, :, b), [h*w, ch]);
```

```
        % caculate dydx
        y(:, :, :, b) = reshape(a * dzdy_b, [h, w, ch]) / (h * w);
    end
else
    if gpuMode
        y = gpuArray(zeros([1, 1, ch * ch, bs], 'single'));
    else
        y = zeros([1, 1, ch * ch, bs], 'single');
    for b = 1:bs
        a = reshape(x(:, :, :, b), [h * w, ch]);
        % caculate output
        y(1, 1, :, b) = reshape(a'*a, [1, ch*ch]) / (h * w);
    end
end
```

Qustion remaining: where is the pooling?

3. Analysis `function y = vl_nnbilinearclpool(x, varargin)` for asymmetric bcnn

```
function y = vl_nnbilinearclpool(x1, x2, varargin)
% ------------------------------------------------------------------------
-----------
% functionality:
%   implementation of the feed forward pass and backward pass of 'bilinear
clpool', which
%   is a layer connected next to a pretrained CNN model (or two same model
)
%
% $x$: input feature of size [height, width, channels, batch size].
% $varargin$: $dzdy$ when in backward pass.
% $y$:
%   forward pass:
%       Self outer product of $x$.
%       For each image with size `[height, width, channels]`, firstly
%       reshape it into size `[height * width, channels]`, and then
%       compute the output
%                   $$y = \frac{1}{height * width} x^T x$$
%       which gives $y$ the size `[channels, channels]`, reshape it again
to a vector.
```

```
%   backward pass:
%       gradient of $y$ w.r.t $x$.
%       $y$ is the same size as $x$, i.e., `[height, width, channels, batc
h_size]`.
%       For each image, reshape $dzdy$ to size `[channels, channels]`
%                       reshape $x$ to size `[height * width, channels]`.
%       $dydx$ is caculated as:
%                   $$y = \frac{1}{height * width} x * dzdy$$
%       which gives $y$ the size `[height * width, channels]`,
%       Reshape $y$ to `[height, width, channels]` as output.
% ----------------------------------------------------------------------
-----------

% if the number of elements in `varargin` > 0 and the first element is not
 a string
backMode = numel(varargin) > 0 && ~isstr(varargin{1})

% if in backward mode, take out the `dzdy` in `varargin`
if backMode dzdy = varargin{1}; end

% if `x` is a `gpuArray`, it is in `gpuMode`
gpuMode = isa(x1, 'gpuArray');

% unpack the size of x into height, width, number of channel, and batch si
ze
[h1, w1, ch1, bs] = size(x1);
[h2, w2, ch2, ~] = size(x2);

% resize the CNN output to the same size
if w1 * h1 <= w2 * h2
    % downsample feature 2
    x2 = array_resize(x2, w1, h1);
else
    % downsample feature 1
    x1 = array_resize(x1, w2, h2);
end
h = size(x1, 1); w = size(x1, 2);

% backward mode
if backMode
    if gpuMode
        y = gpuArray(zeros(size(x), 'single'));
    else
```

```
        y = zeros(size(x), 'single'));
    end
    % for each image / for each feature map with `ch` channels
    for b = 1:bs
        dzdy_b = reshape(dzdy(1, 1, :, b), [ch1, ch2]);
        A = reshape (x1(:, :, :, b), [h*w, ch1]);
        B = reshape (x2(:, :, :, b), [h*w, ch2]);
        dB = reshape(A * dzdy_b, [h, w, ch2]);
        dA = reshape(B * dzdy_b', [h, w, ch1]); %'
        if w1 * h1 <= w2 * h2
            % B is downsampled
        else
            % A is downsampled
    end
% feed forward pass
else
    if gpuMode
        y = gpuArray(zeros([1, 1, ch1 * ch2, bs], 'single'));
    else
        y = zeros([1, 1, ch1 * ch2, bs], 'single');
    for b = 1:bs
        xa = reshape(x1(:, :, :, b), [h * w, ch1]);
        xb = reshape(x2(:, :, :, b), [h * w, ch2]);
        y(1, 1, :, b) = reshape(xa'*xb, [1, ch1*ch2]); % why not '/(h *w)'
?
    end
end

function Ar = array_resize(A, w, h)
%----------------------------------------
% downsample A with size `[w, h]`
%----------------------------------------
```

4. Analysis `function y = vl_nnsqrt(x, param, varargin)` for bcnn

```
    function y = vl_nnsqrt(x, param, varagin)
    % ------------------------------------------------------------------
-
    % functionality: perform square root normalization for the input featu
res
```

```
    %                at each location
    %
    % x: the input features of size [height, width, channels, batch_size]
    % param: the threshold to prevent large value when close to 0
    % varargin: dzdy, only needed in backward pass
    % y:
    %     forward pass:
    %         y = sign(x) .* sqrt(|x|)
    %     backward pass:
    %         dydx = 0.5 ./ sqrt(|x| + param)
    %         y = dydx .* dzdy % the chain rule
    % ----------------------------------------------------------------
-
```

5. Analysis `function y = vl_nnl2norm(x, param, varargin)` for `bcnn`

```
    function y = vl_nnl2norm(x, param, varagin)
    % ----------------------------------------------------------------
-
    % functionality: perform square root normalization for the input featu
res
    %                at each location
    %
    % x: the input features of size [height, width, channels, batch_size]
    % param: the threshold to prevent large value when the norm is close t
o 0
    % varargin: dzdy, only needed in backward pass
    % y:
    %     forward pass:
    %         y = x ./ ||x||  % note: ||x|| is l-2 norm
    %     backward pass:
    %         \frac{d}{dx_j}(x./||x||) = \frack{1}{||x||^3}(x_1^2 + ... +
x_{j-1}^2 + x_{j+1}^2 + ... + x_n^2)
    %         (d/dx_j)(x./||x||) = 1 / ||x|| - x_j^2 / ||x|| ^ 3
    %         gradient(x./||x||) = 1 / ||x|| - x.^2 / ||x|| ^ 3
    % ----------------------------------------------------------------
-
```

6. Shift `gcc/g++` between version `4.7/4.8`

setting `gcc` :

```
> sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.7 50
```

info:

```
update-alternatives: using /usr/bin/gcc-4.7 to provide /usr/bin/gcc (gcc)
in auto mode
```

```
> sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 40
> sudo update-alternatives --config gcc
```

info:

```
There are 2 choices for the alternative gcc (providing /usr/bin/gcc).

  Selection    Path              Priority   Status
------------------------------------------------------------
* 0            /usr/bin/gcc-4.7   50        auto mode
  1            /usr/bin/gcc-4.7   50        manual mode
  2            /usr/bin/gcc-4.8   40        manual mode

Press enter to keep the current choice[*], or type selection number:
```

Setting `g++` :

```
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.8 40
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.7 50
sudo update-alternatives --config g++
```

info:

```
There are 2 choices for the alternative g++ (providing /usr/bin/g++).

  Selection    Path              Priority   Status
------------------------------------------------------------
* 0            /usr/bin/g++-4.7   50        auto mode
  1            /usr/bin/g++-4.7   50        manual mode
  2            /usr/bin/g++-4.8   40        manual mode
```

```
Press enter to keep the current choice[*], or type selection number: 1
```

After change `gcc/g++` to version 4.7, the package `matconvnet` compiles with no complaint of `gcc/g++ -4.8` not being supported.

7. Attemptation 3 on making `run_experiments_bcnn_train.m` run.

```
remove `/path/to/bcnn/data/checkgpu`
> run_experiments_bcnn_train()
```

Error info:

```
...
199: 199.Winter_Wren (train:    30, test:    30 total:    60)
200: 200.Common_Yellowthroat (train:    30, test:    30 total:    60)
 Inf:       **total** (train: 11788, test: 11788 total: 11788)
dataset: there are 11788 images (5994 trainval 5794 test)
Initialization: extracting bcnn feature of batch 1/185
Error using vl_argparse (line 99)
Unknown parameter 'cropSize'

Error in vl_argparse (line 79)
      opts = vl_argparse(opts, vertcat(params,values)) ;

Error in imdb_get_batch_bcnn (line 35)
    opts(i) = vl_argparse(opts(i), {varargin{1}(i),varargin{2:end}});

Error in getBatchSimpleNNWrapper>getBatchSimpleNN (line 10)
im = imdb_get_batch_bcnn(images, opts, ...

Error in getBatchSimpleNNWrapper>@(imdb,batch)getBatchSimpleNN(imdb,batch,
opts) (line 4)
fn = @(imdb, batch) getBatchSimpleNN(imdb, batch, opts) ;

Error in initializeNetworkSharedWeights (line 117)
            [im, labels] = getBatchFn(imdb, batch) ;

Error in imdb_bcnn_train_dag (line 65)
net = initNetFn(imdb, encoderOpts, opts);
```

```
Error in run_experiments_bcnn_train (line 81)
          imdb_bcnn_train_dag(imdb, opts);
```

Err…Such disgusting!

8. Install caffe.

```
> cd /path/to/caffe_root
> cp Makefile.config.example Makefile.config
```

Adjust Makefile.config:

1. uncomment to build with cuDNN, add the `include path` and `lib path` of your cuDNN.

   > ***USE_CUDNN := 1***
   > *# Whatever else you find you need goes here.*
   > *INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include* ***/usr/local/cuda/include***
   > *LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib* ***/usr/local/cuda/lib64***

2. add `cuda installation directory` :

   > ***CUDA_DIR := /usr/local/cuda-8.0***

3. uncomment to use OpenCV 3

   > ***OPENCV_VERSION := 3***

4. Uncomment and set the right path — to compile the matlab interface

   > *# MATLAB directory should contain the mex binary in /bin.*
   > ***MATLAB_DIR := /usr/local/MATLAB/R2015b***

5. Uncomment and set the right path — to use Python 3

   > ***PYTHON_LIBRARIES := boost_python3 python3.4m***
   > ***PYTHON_INCLUDE := /usr/include/python3.4m ***
   > ***/usr/lib/python3/dist-packages/numpy/core/include***

6. installing…

   ```
   > sudo make all
   ```

   No error!

```
> sudo make test
```

No error!

```
> sudo make runtest
```

Error, how disgusting!

```
 nvcc warning : The 'compute_20', 'sm_20', and 'sm_21' architectures
are deprecated, and may be removed in a future release (Use -Wno-depr
ecated-gpu-targets to suppress warning).
 LD -o .build_release/lib/libcaffe.so.1.0.0-rc3
 CXX/LD -o .build_release/test/test_all.testbin src/caffe/test/test_c
affe_main.cpp
 .build_release/tools/caffe
 .build_release/tools/caffe: error while loading shared libraries: li
bcudnn.so.5: cannot open shared object file: No such file or director
y
 make: *** [runtest] Error 127
```

Cannot open shared object file: libcudnn.so.5 !!
Obviously, cuDNN was not installed correctly!

9. Attemptation 4 on making `run_experiments_bcnn_train.m` run.

Maybe some file is corrupt? But the packages are all install successfully… Anyway, just give it a shot…

…

After reinstall `bcnn` completely from scratch, nonthing changed!

…

10. Attemptation 5 on making `run_experiments_bcnn_train.m` run.

—By analysing source code.