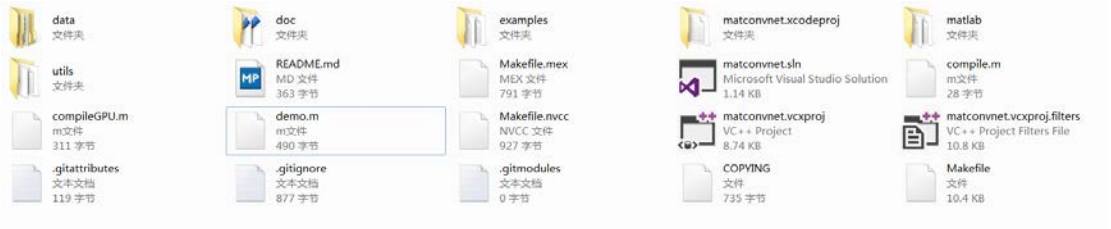
MatConvNet 使用指南

MatConvNet 是牛津大学视觉组深度学习的工具，主页：http://www.vlfeat.org/matconvnet/。
Windows 下编译还是很顺利的，至少比 Caffe 要省很多事，不需要配那么多的第三方库。不过对 matlab 的版本有要求，我选择的是 matlab2015a，C++编译器是 VS2013.
MatConvNet 在 1.16 时有过重大更改，其后的模型不兼容之前的模型文件，这点特别需要注意，为了紧跟时代的发展，这里选择的是最新的 1.20 版本，下载地址：点此下载。里面的文件大致如下图所示：



有几个文件是我为了方便测试而加进去的。
新建一 compile.m 文件，用来编译 CPU 版本，内容如下：

```
addpath matlab
vl_compilenn
```

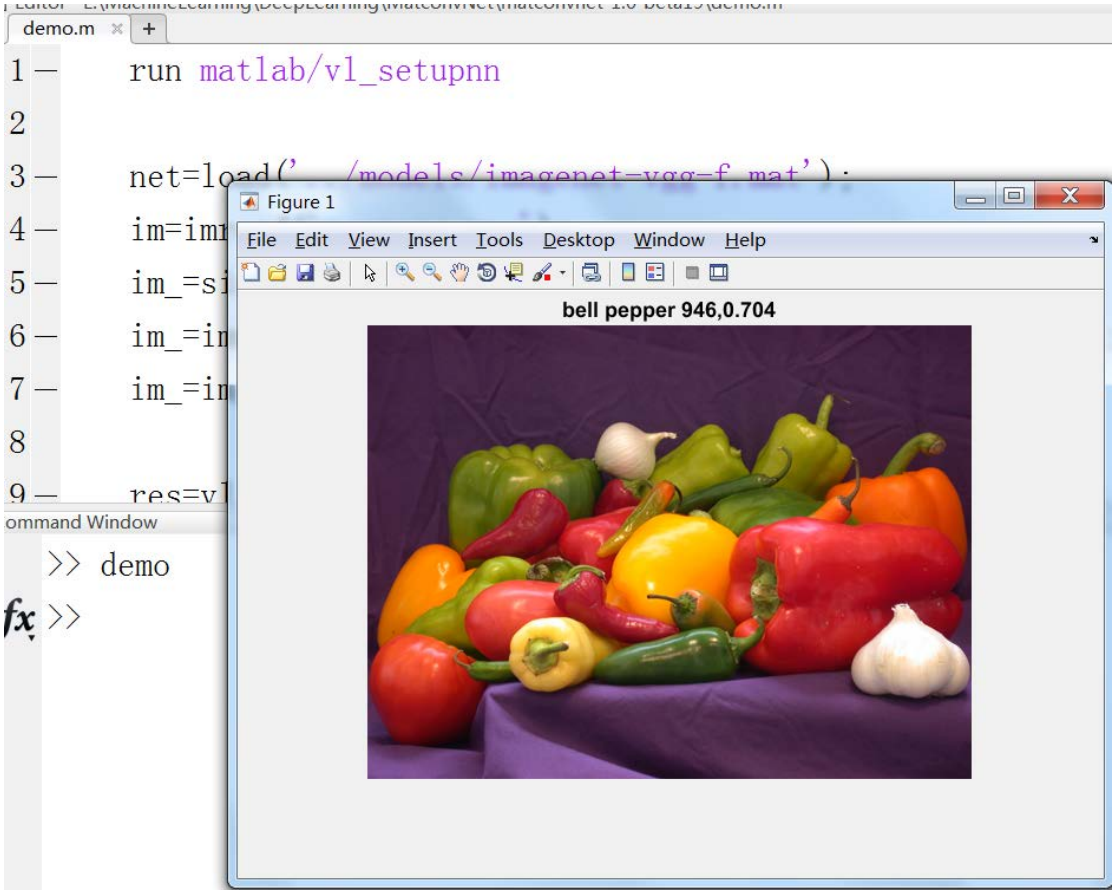
运行，不需要几分钟就可以编译完成。
新建一 compileGPU.m 文件，用来编译 GPU 版本，内容如下：

```
addpath matlab
vl_compilenn('enableGpu', true, ...
                    'cudaRoot', 'C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.0', ...
                    'cudaMethod', 'nvcc');%,...
%                       'enableCudnn', 'true',...
%
'cudnnRoot','E:\MachineLearning\DeepLearning\CuDNN\CUDNNv4') ;
```

里面改成自己安装 CUDA 的位置，不过一般默认的话这个就可以，注释掉的是 cudnn V4 的支持，可以自己加上。
运行，比上面那个稍长时间就可以编译好。可能会有一大堆的警告，不过没有关系。
为了测试编译的到底能不能用，新建 demo.m 文件，内容如下：

```
run matlab/vl_setupnn
net=load('../models/imagenet-vgg-f.mat');%此处换成自己下载模型存储的位置
im=imread('peppers.png');
im_=single(im);
im_=imresize(im_,net.meta.normalization.imageSize(1:2));%缩放到 224*224 大小
im_=im_-net.meta.normalization.averageImage;
```

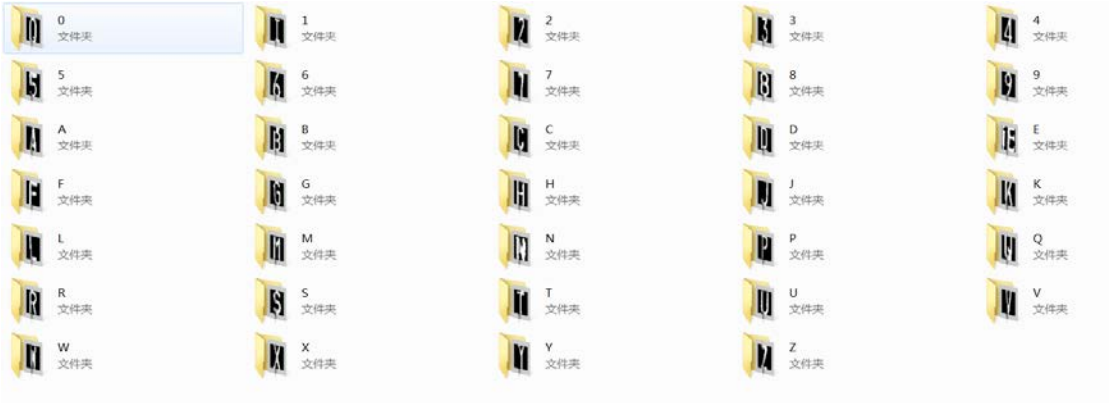
```
res=vl_simplenn(net,im_);
y=res(end).x;
x=gather(res(end).x);
scores=squeeze(gather(res(end).x));
[bestScore,best]=max(scores);
figure(1);
clf;
imshow(im);
title(sprintf('%s %d,%.3f',...
    net.meta.classes.description{best},best,bestScore));
```



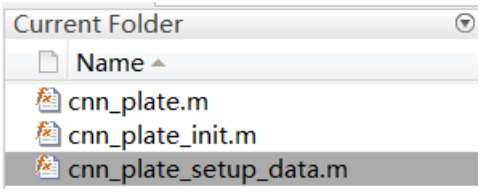
非常成功的把图片正确分类。

如果还想继续深入，就可以看 examples 里的例子了，有 mnist、cifar 和 imagenet 的相关文件，需要下载相应的数据，imagenet 的数据相当大，有数百 G 之多，没必要亲自来一遍。

下面是最关心的问题，如何训练自己的数据。这里以车牌识别 EasyPR 项目的车牌字符识别为例，我们下载对应的车牌字符（在浏览器新建下载就好了，把网址贴进去 https://github.com/liuruoze/EasyPR/blob/master/resources/train/ann.7z，解压后就能看到），文件结构如下所示：



我只选取了数字和字母的文件，每个文件夹下存放对应的图片。
我们工作的目录结构如下所示：



首先新建 cnn_plate_setup_data.m 文件，读取相应的图片和标签，并减去均值。

```
function imdb =cnn_plate_setup_data(datadir)
inputSize =[20,20,1];
subdir=dir(datadir);
imdb.images.data=[];
imdb.images.labels=[];
imdb.images.set = [] ;
imdb.meta.sets = {'train', 'val', 'test'} ;
image_counter=0;
trainratio=0.8;
for i=3:length(subdir)
        imgfiles=dir(fullfile(datadir,subdir(i).name));
        imgpercategory_count=length(imgfiles)-2;
        disp([i-2 imgpercategory_count]);
        image_counter=image_counter+imgpercategory_count;
        for j=3:length(imgfiles)
            img=imread(fullfile(datadir,subdir(i).name,imgfiles(j).name));
            img=imresize(img, inputSize(1:2));
            img=single(img);
%               [~,~,d]=size(img);
%               if d==3
%                   img=rgb2gray(img);
%                   continue;
%               end
            imdb.images.data(:,:,:,end+1)=single(img);
            imdb.images.labels(end+1)= i-2;
            if j-2<imgpercategory_count*trainratio
```

```
                imdb.images.set(end+1)=1;
            else
                imdb.images.set(end+1)=3;
            end
        end
end
dataMean=mean(imdb.images.data,4);
imdb.images.data = single(bsxfun(@minus,imdb.images.data, dataMean)) ;
imdb.images.data_mean = dataMean;
end
```

还需要设计网络的结构，在 cnn_plate_init.m 文件里实现：

```
function net =cnn_plate_init()
rng('default');
rng(0) ;

f=1/100 ;
net.layers = {};
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(3,3,1,20, 'single'), zeros(1, 20, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(3,3,20,100, 'single'),zeros(1,100,'single')}}, ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'conv', ...
    'weights', {{f*randn(3,3,100,1000, 'single'),zeros(1,1000,'single')}}, ...
    'stride', 1, ...
    'pad', 0) ;
net.layers{end+1} = struct('type', 'softmaxloss') ;
```

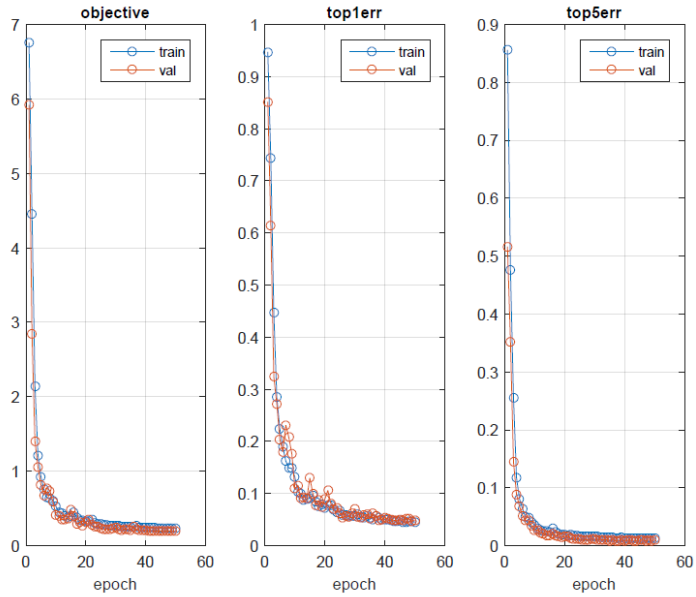
```
% Meta parameters
net.meta.inputSize = [20 20 1] ;
net.meta.trainOpts.learningRate = logspace(-3, -5, 100);
net.meta.trainOpts.numEpochs = 50 ;
net.meta.trainOpts.batchSize = 1000 ;

% Fill in defaul values
net = vl_simplenn_tidy(net) ;

end
```
特别注意网络的结构，不行的话多调几个试试。
最后是把这些串联起来，进行训练：
```
function [net, info] = cnn_plate()
run(fullfile(fileparts(mfilename('fullpath')),...
  '..', '..', 'matlab', 'vl_setupnn.m')) ;
%datadir='E:\MachineLearning\caffe\caffe-windows-
master\platerecognition\data\platerecognition\chars2';
datadir='E:\PatternRecognition\PlateRecognition\EasyPR\EasyPR-
1.4\resources\train\ann';
opts.expDir = fullfile(vl_rootnn, 'data', 'plate-baseline') ;
opts.imdbPath = fullfile(opts.expDir, 'imdb.mat');
if exist(opts.imdbPath,'file')
      imdb=load(opts.imdbPath);
else
      imdb=cnn_plate_setup_data(datadir);
      mkdir(opts.expDir) ;
      save(opts.imdbPath, '-struct', 'imdb') ;
end
net=cnn_plate_init();
net.meta.normalization.averageImage =imdb.images.data_mean ;
opts.train.gpus=1;
[net, info] = cnn_train(net, imdb, getBatch(opts), ...
  'expDir', opts.expDir, ...
  net.meta.trainOpts, ...
  opts.train, ...
  'val', find(imdb.images.set == 3)) ;


function fn = getBatch(opts)
% --------------------------------------------------------------------
      fn = @(x,y) getSimpleNNBatch(x,y) ;
end
function [images, labels]    = getSimpleNNBatch(imdb, batch)
```

```
        images = imdb.images.data(:,:,:,batch) ;
        labels = imdb.images.labels(1,batch) ;
        if opts.train.gpus > 0
            images = gpuArray(images) ;
        end
    end
end
```



GPU 训练起来还是很快的。

训练好后自己写个 demo 识别下吧：

```
run(fullfile(fileparts(mfilename('fullpath')),...
  '..', '..', 'matlab', 'vl_setupnn.m')) ;
addpath ../../data/plate-baseline;
%datadir='E:\MachineLearning\caffe\caffe-windows-
master\platerecognition\data\platerecognition\chars2';
datadir='E:\PatternRecognition\PlateRecognition\EasyPR\EasyPR-
1.4\resources\train\ann';
class=47;index=1;
subdir=dir(datadir);
imgfiles=dir(fullfile(datadir,subdir(class+2).name));
img=imread(fullfile(datadir,subdir(class+2).name,imgfiles(index+2).name));
imshow(img);
net=load('net-epoch-50.mat');
net=net.net;
im_=single(img);
im_=imresize(im_,net.meta.inputSize(1:2));
im_=im_ - net.meta.normalization.averageImage;
opts.batchNormalization = false ;
net.layers{end}.type = 'softmax';
res=vl_simplenn(net,im_);
```

```
scores=squeeze(gather(res(end).x));
[bestScore,best]=max(scores);
disp([subdir(best+2).name ' ' bestScore]);
```

```
1 —     run(fullfile(fileparts(mfilename('fullpath')),.
2         '..', '..', 'matlab', 'vl_setupnn.m')) ;
3 —     addpath ../../data/plate-baseline;
4       %datadir='E:\MachineLearning\caffe\caffe-windo
5 —     datadir='E:\PatternRecognition\PlateRecognitio
6 —     class=47;index=1;
7 —     subdir=dir(datadi
8 —     imgfiles=dir(full
9 —     img=imread(fullfi
0 —     imshow(img);
```

```
ommand Window
    layers: {1x8 cell}
      meta: [1x1 struc

>> demo
zh_liao
```