艾斯1213

深度卷积网络CNN与图像语义分割

转载请注明出处: http://xiahouzuoxin.github.io/notes/html/深度卷积网络CNN与图像语义分割.html

说好的要笔耕不缀，这开始一边实习一边找工作，还摊上了自己的一点私事困扰，这几个月的东西都没来得及总结一下。这就来记录一下关于CNN、Caffe、Image Sematic Segmentation相关的工作，由于公司技术保密的问题，很多东西没办法和大家详说只能抱歉了。在5月份前，我也是一个DL和CNN的门外汉，自己试着看tutorials、papers、搭Caffe平台、测试CNN Net，现在至少也能改改Caffe源码（Add/Modify Layer）、基于Caffe写个Demo。这里希望把学习的过程分享给那些在门口徘徊的朋友。没法事无巨细，但希望能起到提点的作用！“乍可刺你眼，不可隐我脚”。

级别1：DL快速上手

1. UFLDL： http://deeplearning.stanford.edu/tutorial/

   这是stanford Ng老师的教材，也刚好是以CNN为主，Ng老师教材的特色就是简洁明白。一遍看不懂多看两遍，直到烂熟于心，顺便把里面的Matlab Exercises完成了。

2. http://deeplearning.net/tutorial/

   PRML作者给的Python入门DL的tutorial，基于Theano Framework，有些偏向于RNN的东西。

一句简单的话描述：“深度学习就是多层的神经网络”。神经网络几十年前就有了，而且证明了“2层（1个隐层）的神经网络可以逼近任意的非线性映射”，意思就是说，只要我的参数能训练好，2层神经网络就能完成任意的分类问题（分类问题就是将不同类通过非线性映射划分到不同的子空间）。但2层神经网络存在的问题是：

> 如果要逼近非常非常复杂的非线性映射，网络的权值W就会很多，造成Train时候容易出现的问题就是Overfitting。所以大事化小，将复杂问题进行分割，用多层网络来逼近负责的非线性映射，这样每层的参数就少了。自然而然的网络就从2层变成了多层，浅网络(shallow)就变成了深网络(deep)。

但科研界的大牛们会这么傻吗，十几年前会想不到用多层网络来进行非线性映射？看看CNN最早的工作：http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf 那是98年的，Train了一个5层的CNN来进行MINIST数据集的数字图片分类。多层神经网络一直不火我觉得有这么两个原因：

1. 神经网络中非线性的映射的极值优化问题本身是一个非凸问题，本身数学理论上的就对非凸优化问题缺少严格有效最优化方法的支撑。直到现在也依然对各层Layer的输出解释不清不楚，但效果就是好，这还得归功于各种大神藏之捏之的各种Tricks
2. 数据与计算能力的问题。十来年前哪来随随便便就这么大的硬盘，哪里去找像ImageNet这样1000类的数据集。“大数据是燃料，GPU是引擎”，正是因为大数据的出现和GPU编程的出现带动了DL的进展，这些在10年前是做不来的。我在CPU与GPU上跑自己简化的Googlenet，GPU比CPU快10倍。

DL只是一个概念而已。对于做图像和视觉的就该一头扎到CNN(Convolutional Neural Netwok)，做自然语言的就该投入到RNN(Recurrent Neural Network)。我是做图像的。CNN的学习资料除了上面Ng的tutorial外，还有一个Stanford Li Fei-Fei教授的课程cs231：Convolutional Neural Networks for Visual Recognition，http://cs231n.github.io/convolutional-networks/ 是Notes中一份关于CNN非常详细的资料。

级别2：从Caffe着手实践

先看看这个热个身：贾扬清：希望Caffe成为深度学习领域的Hadoop，增加点学习的欲望，毕竟现在多少人靠着Hadoop那玩意儿挣着大钱。

接着请认准Caffe官方文档： http://caffe.berkeleyvision.org/ 和Github源码： https://github.com/BVLC/caffe 。毫不犹豫fork一份到自己的Github。然后就是照着INSTALL来Complie和Config Caffe了，值得注意的是，安装OpenCV的时候推荐使用源码安装。

先自己熟悉Caffe的架构，主要参考资料就是官网文档，我自己刚开始的时候也写了个小的ppt笔记：Diving into Caffe.pptx

还有两份不错的Caffe tutorials：

1. Standford CS231课程的Caffe tutorial
2. Oxford的Caffe tutorial

接着就是要自己动手，实打实地分析一个CNN，比如LeNet、AlexNet，自己在纸上画一画，像下面那样

LeNet

AlexNet

搞明白下面这些玩意的作用，没见过的就google一个个的查：

1. Convolution
2. Pooling
3. ReLU： Rectified Linear Units
4. LRN: Local Response Normalization
5. finetune
6. dropout

搞明白SGD中weight的更新方法，可以参考Caffe网站上tutorial的Solver部分：http://caffe.berkeleyvision.org/tutorial/solver.html 。

好了，现在试着去跑跑${CAFFE_ROOT}/example下LeNet的example吧。

随笔分类

随笔档案

最新评论

1. Re:ubuntu15.10 源码安装 tensorflow

请问这个怎么解决，谢谢

--zbtf

阅读排行榜

评论排行榜

推荐排行榜

级别3：读paper，网络Train起来

当去搜索ICRL、CVPR、ICCV这些最前沿的计算机视觉、机器学习会议的时候，只要是涉及图像相关的深度学习实验，大都是基于Caffe来做的。所以，只要抓住1~2篇popular的paper深入，把论文中的CNN在Caffe上复现了，就能找到一些感觉了。在这期间，下面一些经典论文是至少要读一读的：

1. LeNet-5: Gradient-Based Learning Applied to Document Recognition CNN首篇paper，虽然是1998年的文章，但依然值得仔细一读，熟悉下CNN这玩意儿。
2. AlexNet: ImageNet Classification with Deep Convolutional Neural Networks 自我感觉是促进CNN的扛鼎之作，似乎很多所谓的Tricks在这篇文章中能找到，看这篇文章就是来学Tricks的。你在这里能看到诸如Overlap Pooling、LRN、带momentum的SGD等等。
3. Googlenet: Going Deeper with Convolutions ImageNet竞赛Number1，有效的Inception结构给我映像深刻。
4. VGGNet: Very Deep Convolutional Networks for Large-Scale Image Recognition ImageNet竞赛Number2，典型的卷积+Pooling方式构建深层网络，但是由于没有Googlenet中Inception的1x1的convolution用于减小网络厚度，时间上要比Googlenet慢一些。论文中从A-E由浅到深训练深度网络的方法值得在搭建自己的网络时学习，这个后面再表。
5. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks 这篇paper至少告诉我一件事，深度网络提取的特征对图片Classification、Detection、Segmentation等都有效。也就是说，我拿一个Net到ImageNet去Train一个1000类的分类model出来，我又要把这个Net用于图片Detection，这篇paper告诉你：好了，不用再Train一个Detection model了，我的Classification model直接给你用，你除了需要把后端的Softmax改一改之外，其它啥都不用改，这个Net照样跑得和Classification任务中一样的好。

具体到用CNN做Sematic Segmentation，利用到全卷积网络，对下面两篇进行了精读，并且都Caffe上复现过并用于分割任务，

1. FCNN: Fully Convolutional Networks for Semantic Segmentation
2. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs

下面是几个月前我看过这两篇paper后做得ppt：

1. FCN for Sematic Segmentation.pptx
2. Semantic Image Segmentation With Deep Convolutional Nets and Fully Connected CRFs.ppt

这两篇paper有一个共同点，都用到了multiscale的信息（也就是将High Layer的输出与Low Layer的输出进行结合），这对促进分割效果有很大的作用。值得一提的是，在Train multiscale的model时，由于存在反卷积或上采样的操作，Layer相对复杂，很难一大锅扔进去还Trian得很好，所以通常会先Train一个无multiscale的model，再用该model去finetune含multiscale的Net。

级别4：Demo跑起来

读一些源码玩玩

1. caffe.proto
2. Convolution Layer

   CNN里面最重要的运算就是卷积，要知道Caffe是怎么做卷积的，就看看src/caffe/layers/conv_layer.cpp。Caffe里的卷积计算：

   

   Caffe里的卷积操作

   Caffe计算卷积时先将图片中用于卷积的每个patch拉成一个行向量，若stride=1，则卷积的patch个数刚好是W*H，则将原图片保存成一个[WxH,KxKxD]矩阵，同理，滤波器也存成一个[M,KxKxD]矩阵，卷积计算只要计算这两个矩阵的乘积，矩阵乘积的工作刚好可以交给BLAS来进行优化，也就是为了用BLAS优化而进行这种转化，从中可以看出，Caffe里的卷积还是挺耗内存空间的！

3. SoftmaxLossLayer

   所谓的输出的Loss计算使用的一般是Softmax或Sigmoid的交叉谱，具体可看这里SoftmaxWithLossLayer

4. DataLayer

5. 自己添加一个Layer（比如Segmentation中常用的精度指标是IoU而不是简单的Accuracy）
   - 在caffe.proto中的message LayerParameter部分增加Layer Type和layerParameter，并在caffe.proto中声明该message LayerParameter（protobuf里的message类似C里的struct）
   - 在include的某个头文件中声明LayerClass
   - 在src/caffe/layers/中新建一个.cpp，实现LayerSetup、Shape、Forward_cpu以及Backward_cpu
   - 为建立LayerClass与proto中Layer声明的关联，在上面的.cpp中还要使用INSTALL_CLASS(...)和REGISTER_LAYER_CLASS(...)

   上面仅仅是我随手写在草稿纸上的随笔，其实自己看看源码，依葫芦画瓢写一个简单的Layer就知道了。

熟悉Caffe接口，写Demo这是硬功夫

Caffe提供了好用的接口，包括matlab、C++、Python！由于特殊原因，我不能公开我C++和matlab的Demo源码以及其中的一些后处理技术，暂且只能给大家看一些分割的结果：

Sematic Segmentation Result

还有一个视频语义分割的结果，大家看看，热闹热闹就好，

分析各层Layer输出特征

我一开始以为看看各层Layer的输出，能帮助我改进Net，可却发现错了，除了前几层还能看出点明亮或边缘信息外，网络后端Layer的输出压根就没办法理解。extract_featmat.cpp是我基于extract_features.cpp改的一个Caffe tool，放到tools目录下编译就好了，使用方法看help：

```
void print_help(void) {
  LOG(ERROR)<<
  "This program takes in a trained network and an input image, and then\n"
  " extract features of the input data produced by the net.\n"
  "Usage: extract_featmat [options]\n"
  "  -model     [pretrained_net_param]\n"
  "  -proto     [feature_extraction_proto_file]\n"
  "  -img       [rgb_image_name]\n"
  "  -blobs     [extract_feature_blob_name1[,name2,...]],refrence .prototxt with"
  "            \"blob:\". [all] for all layers. \n"
  "  -wr_prefix [output_feat_prefix1[,prefix2,...]], conrespond to -blobs\n"
  "  -wr_root   [output_feat_dir], optional, default \"./\", make sure it exist\n"
  "  -gpu       [gpu_id],optional,if not specified,default CPU\n";
}
```

下面图是一些Layer的输出blob，从结果可以看出，前面的layer还能看到一些边缘信息，后面的layer就完全看不出原图像相关的信息了，

不同Layers的Feature变化

级别5：何不自己搭个CNN玩玩

虽然还是个新手，关于搭建CNN，还在慢慢在找感觉。我觉得从两方面：

1. 利用已有的网络，使劲浑身解数找它们的缺点，改进它们

   熟读Googlenet和VGGnet那两篇paper，两者的CNN结构如下： 

   GoogleNet

   

   VGGNet

   VGG不是Weight Filter不是非常厚么，卷积操作复杂度就高。而Googlenet通过Inception中1x1的Convolution刚好是为了减少Weight Filter的厚度，我最近一段时间在尝试做的事就是将VGG中的Layer用Googlenet中的Inception的方式去替代，希望至少在时间上有所改进。

2. 从头搭建一个CNN用于解决实际问题。一个词：搭积木。

   先搭一个简单的，比如说就3层：卷积-Pooling-卷积-Pooling-卷积-Pooling，先把这个简单的网络训练以来，效果不好没关系，我们接着往上加，直到满意为止。不明白的看看上面的VGG Net，先在ImageNet上Train出一个A网络，然后增加A的深度(粗体的部分)变成B，再用A去初始化B网络，就逐级增加网络深度。 这里面有一个finetune的技巧，那就是用浅层的网络训练weight结果去初始化或finetune深层网络。这也是为什么不直接一开始就搭建深层网络的原因，前面说过，深度网络的Train是个非凸问题，是个至今难解决的大问题，网络初始化对其收敛结果影响很大，finetune就这样作为Deep Network中一项最重要的tricks而存在了。finetune除了由浅至深逐级初始化帮助收敛外，还有一个作用：将自己的网络在一个非常非常大的数据集上(现在最大的ImageNet)进行Train，这个Train的结果再拿去作为实际要解决的问题中用于初始化，这样能增加网络的泛化能力。 关于更多的问题，现在也是盲人摸象，暂且搁下不提。

Train CNN时关于数据集的一些注意事项

1. 论数据集的重要性

Train一个Net的经验也很重要，还是那句话，“数据是燃料”，CNN训练一定要保证足够的数据量（就我现在知道，Train一个深层的全卷积至少要4万张图片以上），否则很容易出现过拟合或者说泛化能力特别差，就像下面那样。下面分别是DatasetSize=4K与DatasetSize=40K同一Net Train出来的Prediction结果。 在训练时，仅从精度上来看，两个Net训练时得到的差距不大，IoU都在90%左右，但实际predict时，4K train出的model是如此的难看！

论Train CNN数据集大小的重要性

2. CNN对目标出现在图片的位置、大小、方向等信息非常敏感，为增加网络泛化能力，最好对数据进行先multiscale、mirror、crop等操作。比如下面就是我用来对图片进行scale的一段matlab代码（这段代码主要是通过resize、填白来scale，但不改变图片长宽比）。当然，scale的方式不局限于此，也可以直接resize，改变图片长宽比。

```
function ImageScale(in_dir,out_dir)
% Author: zuoxin,xiahou
if ~exist(out_dir,'dir')
   mkdir(out_dir);
end
ims = dir(fullfile(in_dir,'*.jpg'));
```

```
gts = dir(fullfile(in_dir,'*.bmp'));
N = length(ims);
if N ~= length(gts)
   error('NUM(Images)~=NUM(GroudTruth)');
end
for i=1:N
   im = imread(fullfile(in_dir,ims(i).name));
   gt = imread(fullfile(in_dir,gts(i).name));
   % Check for size errors
   if ndims(im) < 3
      [h,w] = size(im);
      k=1;
   else
      [h,w,k] = size(im);
   end
   [h_gt,w_gt,k_gt] = size(gt);
   if h~=h_gt || w~=w_gt
      error('size(im)~=size(gt)');
   end
   if h>w
      ts = h;
   else
      ts = w;
   end

   SIZE = 500;
   if h>w
      im = imresize(im, [SIZE,w*SIZE/ts]);
      gt = imresize(gt, [SIZE,w*SIZE/ts]);
   else
      im = imresize(im, [h*SIZE/ts,SIZE]);
      gt = imresize(gt, [h*SIZE/ts,SIZE]);
   end
   % Update Size
   if ndims(im) < 3
      [h,w] = size(im);
      k=1;
   else
      [h,w,k] = size(im);
   end
   [~,~,k_gt] = size(gt);
   if h>w % h>w, padding width (bg:0 fg:1)
      pad_ = SIZE-w;
      left_pad = floor(pad_/2);
      right_pad = ceil(pad_/2);
```

```
        scale_im = [255*ones(h,left_pad,k), im, 255*ones(h,right_pad,k)];
        scale_gt = [255*zeros(h,left_pad,k_gt), gt, 255*zeros(h,right_pad,k_gt)];
    else % h<w, padding height
        pad_ = SIZE-h;
        top_pad = floor(pad_/2);
        bot_pad = ceil(pad_/2);
        scale_im = [255*ones(top_pad,w,k); im; 255*ones(bot_pad,w,k)];
        scale_gt = [255*zeros(top_pad,w,k_gt); gt; 255*zeros(bot_pad,w,k_gt)];
    end
    % Other Scales
    ts = [0.6,1,1.4];
    for s = 1:numel(ts)
        im = imresize(scale_im,ts(s));
        gt = imresize(scale_gt,ts(s));
        % padding/crop to SIZExSIZE
        if size(im,1) < SIZE  % Padding
            right_pad = SIZE - size(im,2);
            bot_pad = SIZE - size(im,1);
            im = [im, 255*ones(size(im,1),right_pad,k)];
            im = [im; 255*ones(bot_pad,size(im,2),k)];
            gt = [gt, 255*zeros(size(gt,1),right_pad,k_gt)];
            gt = [gt; 255*zeros(bot_pad,size(gt,2),k_gt)];
        else  % Crop
            im = im(1:SIZE,1:SIZE,:);
            gt = gt(1:SIZE,1:SIZE,:);
        end
        gt(gt>=128) = 255;
        gt(gt<128)  = 0;
        out_im = fullfile(out_dir,...
            [ims(i).name(1:end-4),'_',num2str(ts(s)),'.jpg']);
        out_gt = fullfile(out_dir,...
            [ims(i).name(1:end-4),'_',num2str(ts(s)),'.bmp']);
        imwrite(im,out_im);
        imwrite(gt,out_gt);
    end
end
end
```

下面是mirror图片的matlab脚本：

```
function image_mirror(d)
% mirror all images under @dir directory
% and store the mirrored images under @dir

files=dir(fullfile(d,'*.jpg'));
n=length(files);
```

```
for i=1:n
   src_file=fullfile(d,files(i).name);
   out_file=fullfile(d,[files(i).name(1:end-4),'_mr.jpg']);
   if ~exist(out_file,'file') && ~(strcmp(src_file(end-5:end-4),'mr'))
      im = imread(src_file);
      out=im(:,end:-1:1,:);
      % imshow(out);
      imwrite(out, out_file);
      fprintf('%s\n',fullfile(d,files(i).name));
   end
end
```

3. 关于输入图片尺寸的问题

对于非分割的问题，比如识别，我觉得在200x200 pixles左右就足够了（看大家都这么用的，这个自己没测试过）。但对于分割的问题，图片的size对分割结果影响还是很大的，用全卷积网络的测试结果：输入图片的size从500x500降低到300x300，IoU果断直接降了3个点，太恐怖了！！但size太大，卷积时间长，所以这就是一个精度和时间的折中问题了。

级别6：加速吧，GPU编程

呃，这一级还没练到，但迟早是要做的，说了“大数据是燃料，GPU是引擎”的，怎么能不懂引擎呢……

关于语义分割的一些其它工作

1. CRF：CRF在图像分割中是最常见的refine后处理手段。在CNN中目标是做成end-2-end的CRF，实习这段时间也做过不少这部分的工作，Oxford有篇CRF-RNN的paper，将denseCRF重新解释成RNN来进行end-2-end的Training
2. 结合grabcut交互式分割，或者SLIC超像素分割等方法进行边缘精细化的处理
3. 不闲扯淡了……

注：由于“实习上班+实验室+论文+刷leetcode+私事”占用时间的关系，好不容易抽出一个上午+一个晚上整理了一下，暂时想到这么多，算列个提纲吧，文章中不少具体细节有机会再补充。

艾斯1213
关注 - 13
粉丝 - 5
+加关注

« 上一篇：Linux服务器上监控网络带宽的18个常用命令
» 下一篇：【每个人都是梵高】A Neural Algorithm of Artistic Style

posted on 2016-07-22 12:54 艾斯1213 阅读(274) 评论(0) 编辑 收藏

最新IT新闻:
· 盖茨切换到Android暗示Surface Phone可能无见光之日
· 丰田马自达成立合资新公司，日系新能源会迎来第二春吗？
· 紧跟摩拜、ofo，哈罗单车联手支付宝参战"免押金"模式
· 特斯拉出手：造世界最大锂离子电池 蓄满能不得了
· 北美首个三星AI实验室落户蒙特利尔大学
» 更多新闻...

最新知识库文章:
· 实用VPC虚拟私有云设计原则
· 如何阅读计算机科学类的书
· Google 及其云智慧
· 做到这一点，你也可以成为优秀的程序员
· 写给立志做码农的大学生
» 更多知识库文章...