

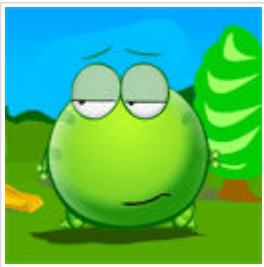
## 网络资源是无限的

目录视图

摘要视图

RSS 订阅

## 个人资料



fengbingchun



访问：3068639次

积分：31403

等级：BLOG &gt; 8

排名：第161名

原创：457篇 转载：141篇

译文：0篇 评论：1767条

异步赠书：Kotlin领衔10本好书 SDCC 2017之区块链技术实战线上峰会 程序员9月书讯 每周荐书：Java Web、Python极客编程（评论送书）

## 使用Caffe进行手写数字识别执行流程解析

2017-04-03 20:25

2811人阅读

评论(3)

分类：Caffe (41) OCR (9)

版权声明：本文为博主原创文章，未经博主允许不得转载。

之前在 <http://blog.csdn.net/fengbingchun/article/details/50987185> 中仿照Caffe中的examples实现对手写数字进行识别，这里详细介绍下其执行流程并精简了实现代码，使用Caffe对MNIST数据集进行train的文章可以参考 <http://blog.csdn.net/fengbingchun/article/details/68065338>：

1. 先注册所有层，执行layer\_factory.hpp中类LayerRegisterer的构造函数，类LayerRegistry的AddCreator和Registry静态函数；关于Caffe中Layer的注册可以参考：  
<http://blog.csdn.net/fengbingchun/article/details/54310956>

2. 指定执行mode是采用CPU还是GPU；

## 文章分类

[ActiveX](#) (18)[Android](#) (9)[Bar Code](#) (16)[CImg](#) (4)[Caffe](#) (41)[C#](#) (4)[CUDA](#) (29)[CMake](#) (4)[Code::Blocks](#) (3)[Contour Detection](#) (9)[CxImage](#) (6)[C/C++/C++11](#) (116)[Database/Dataset](#) (5)[Design Patterns](#) (25)[Deep Learning](#) (18)[Eclipse](#) (3)[Emgu CV](#) (1)[Eigen/OpenBLAS](#) (11)[FFmpeg](#) (1)[Feature Extraction](#) (1)[FreeType](#) (1)[Face](#) (10)[GPU](#) (3)[Git/SVN](#) (3)[GCC](#) (2)[GDAL](#) (5)[HTML](#) (3)[Image Recognition](#) (8)[Image Processing](#) (18)[Image Registration](#) (13)

3. 指定需要的.prototxt和.caffemodel文件：注意此处的.prototxt文件(lenet\_train\_test\_.prototxt)与train时.prototxt文件(lenet\_train\_test.prototxt)在内容上的差异。caffemodel文件即是train后最终生成的二进制文件lenet\_iter\_10000.caffemodel，里面存放着所有层的权值和偏置。lenet\_train\_test\_.prototxt文件内容如下：

[plain]

```
01. name: "LeNet" # net名
02. layer { # memory required: (784+1)*4=3140
03.     name: "data" # layer名字
04.     type: "MemoryData" # layer类型, Data enters Caffe through data layers,
read data directly from memory
05.     top: "data" # top名字, shape: 1 1 28 28 (784)
06.     top: "label" # top名字, shape: 1 (1) #感觉并无实质作用, 仅用于增加一个top blob, 不可
07.     memory_data_param { # 内存数据参数
08.         batch_size: 1 # 指定待识别图像一次的数量
09.         channels: 1 # 指定待识别图像的通道数
10.         height: 28 # 指定待识别图像的高度
11.         width: 28 # 指定待识别图像的宽度
12.     }
13.     transform_param { # 图像预处理参数
14.         scale: 0.00390625 # 对图像像素值进行scale操作, 范围[0, 1)
15.     }
16. }
17. layer { # memory required: 11520*4=46080
18.     name: "conv1" # layer名字
19.     type: "Convolution" # layer类型, 卷积层
20.     bottom: "data" # bottom名字
21.     top: "conv1" # top名字, shape: 1 20 24 24 (11520)
22.     param { # Specifies training parameters
23.         lr_mult: 1 # The multiplier on the global learning rate
24.     }
25.     param { # Specifies training parameters
26.         lr_mult: 2 # The multiplier on the global learning rate
27.     }
28.     convolution_param { # 卷积参数
29.         num_output: 20 # 输出特征图(feature map)数量
30.         kernel_size: 5 # 卷积核大小(卷积核其实就是权值)
31.         stride: 1 # 滑动步长
```

[ImageMagick](#) (3)  
[Java](#) (5)  
[Linux](#) (26)  
[Log](#) (2)  
[Makefile](#) (2)  
[Mathematical Knowledge](#) (27)  
[Multi-thread](#) (8)  
[Matlab](#) (33)  
[MFC](#) (8)  
[MinGW](#) (3)  
[Mac](#) (1)  
[Neural Network](#) (16)  
[OCR](#) (10)  
[Office](#) (2)  
[OpenCL](#) (2)  
[OpenSSL](#) (7)  
[OpenCV](#) (93)  
[OpenGL](#) (2)  
[OpenGL ES](#) (3)  
[OpenMP/Intel TBB](#) (4)  
[Photoshop](#) (1)  
[Python](#) (7)  
[Qt](#) (1)  
[SIMD](#) (14)  
[Software Development](#) (5)  
[System architecture](#) (2)  
[Skia](#) (1)  
[Software Testing](#) (8)  
[Shell](#) (3)  
[Socket](#) (3)  
[Target Detection](#) (2)  
[Target Tracking](#) (2)

使用Caffe进行手写数字识别执行流程解析 - 网络资源是有限的 - CSDN博客

```

32.     weight_filler { # The filler for the weight
33.         type: "xavier" # 权值使用xavier滤波
34.     }
35.     bias_filler { # The filler for the bias
36.         type: "constant" # 偏置使用常量滤波
37.     }
38. }
39. }
40. layer { # memory required: 2880*4=11520
41.     name: "pool1" # layer名字
42.     type: "Pooling" # layer类型, Pooling层
43.     bottom: "conv1" # bottom名字
44.     top: "pool1" # top名字, shape: 1 20 12 12 (2880)
45.     pooling_param { # pooling parameter, pooling层参数
46.         pool: MAX # pooling方法: 最大值采样
47.         kernel_size: 2 # 滤波器大小
48.         stride: 2 # 滑动步长
49.     }
50. }
51. layer { # memory required: 3200*4=12800
52.     name: "conv2" # layer名字
53.     type: "Convolution" # layer类型, 卷积层
54.     bottom: "pool1" # bottom名字
55.     top: "conv2" # top名字, shape: 1 50 8 8 (3200)
56.     param { # Specifies training parameters
57.         lr_mult: 1 # The multiplier on the global learning rate
58.     }
59.     param { # Specifies training parameters
60.         lr_mult: 2 # The multiplier on the global learning rate
61.     }
62.     convolution_param { # 卷积参数
63.         num_output: 50 # 输出特征图(feature map)数量
64.         kernel_size: 5 # 卷积核大小(卷积核其实就是权值)
65.         stride: 1 # 滑动步长
66.         weight_filler { # The filler for the weight
67.             type: "xavier" # 权值使用xavier滤波
68.         }
69.         bias_filler { # The filler for the bias
70.             type: "constant" # 偏置使用常量滤波

```

[VC6 \(6\)](#)  
[VS2008 \(16\)](#)  
[VS2010 \(4\)](#)  
[VS2013 \(3\)](#)  
[vigna \(2\)](#)  
[VLC \(5\)](#)  
[VLFeat \(1\)](#)  
[wxWidgets \(1\)](#)  
[Watermark \(4\)](#)  
[Windows7/10 \(7\)](#)  
[Windows Core Programming \(9\)](#)  
[XML \(3\)](#)

### Free Codes

[pudn](#)  
[freecode](#)  
[Peter's Functions](#)  
[CodeProject](#)  
[SourceCodeOnline](#)  
[Computer Vision Source Code](#)  
[Codesoso](#)  
[Digital Watermarking](#)  
[SourceForge](#)  
[HackChina](#)  
[oschina](#)  
[libsvm](#)  
[joys99](#)  
[CodeForge](#)  
[cvchina](#)  
[tesseract-ocr](#)  
[sift](#)

```

71.     }
72.   }
73. }
74. layer { # memory required: 800*4=3200
75.   name: "pool2" # layer名字
76.   type: "Pooling" # layer类型, Pooling层
77.   bottom: "conv2" # bottom名字
78.   top: "pool2" # top名字, shape: 1 50 4 4 (800)
79.   pooling_param { # pooling parameter, pooling层参数
80.     pool: MAX # pooling方法: 最大值采样
81.     kernel_size: 2 # 滤波器大小
82.     stride: 2 # 滑动步长
83.   }
84. }
85. layer { # memory required: 500*4=2000
86.   name: "ip1" # layer名字
87.   type: "InnerProduct" # layer类型, 全连接层
88.   bottom: "pool2" # bottom名字
89.   top: "ip1" # top名字, shape: 1 500 (500)
90.   param { # Specifies training parameters
91.     lr_mult: 1 # The multiplier on the global learning rate
92.   }
93.   param { # Specifies training parameters
94.     lr_mult: 2 # The multiplier on the global learning rate
95.   }
96.   inner_product_param { # 全连接层参数
97.     num_output: 500 # 输出特征图(feature map)数量
98.     weight_filler { # The filler for the weight
99.       type: "xavier" # 权值使用xavier滤波
100.    }
101.    bias_filler { # The filler for the bias
102.      type: "constant" # 偏置使用常量滤波
103.    }
104.  }
105. }
106. # ReLU: Given an input value x, The ReLU layer computes the output as x if x > 0 and
107. # negative_slope * x if x <= 0. When the negative slope parameter is not set,
108. # it is equivalent to the standard ReLU function of taking max(x, 0).
109. # It also supports in-place computation, meaning that the bottom and

```

TIRG  
imgSeek  
OpenSURF

#### Friendly Link

OpenCL  
Python  
poesia-filter  
TortoiseSVN  
imgSeek  
Notepad  
Beyond Compare  
CMake  
VIGRA  
CodeGuru  
vchome  
aforgenet  
CVLAB  
Doxygen  
Coursera  
OpenMP

#### Technical Forum

Matlab China  
OpenCV China  
The CImg Library  
Open Computer Vision Library  
CxxImage  
ImageMagick  
ImageMagick China

```

110. # the top blob could be the same to preserve memory consumption
111. layer { # memory required: 500*4=2000
112.     name: "relu1" # layer名字
113.     type: "ReLU" # layer类型
114.     bottom: "ip1" # bottom名字
115.     top: "ip1" # top名字 (in-place), shape: 1 500 (500)
116. }
117. layer { # memory required: 10*4=40
118.     name: "ip2" # layer名字
119.     type: "InnerProduct" # layer类型,全连接层
120.     bottom: "ip1" # bottom名字
121.     top: "ip2" # top名字, shape: 1 10 (10)
122.     param { # Specifies training parameters
123.         lr_mult: 1 # The multiplier on the global learning rate
124.     }
125.     param { # Specifies training parameters
126.         lr_mult: 2 # The multiplier on the global learning rate
127.     }
128.     inner_product_param {
129.         num_output: 10 # 输出特征图(feature map)数量
130.         weight_filler { # The filler for the weight
131.             type: "xavier" # 权值使用xavier滤波
132.         }
133.         bias_filler { # The filler for the bias
134.             type: "constant" # 偏置使用常量滤波
135.         }
136.     }
137. }
138. layer { # memory required: 10*4=40
139.     name: "prob" # layer名字
140.     type: "Softmax" # layer类型
141.     bottom: "ip2" # bottom名字
142.     top: "prob" # top名字, shape: 1 10 (10)
143. }
144. # 占用总内存大小为: 3140+46080+11520+12800+3200+2000+2000+40+40=80820

```

lenet\_train\_test\_.prototxt可视化结果(<http://ethereon.github.io/netscope/quickstart.html>)如下图:



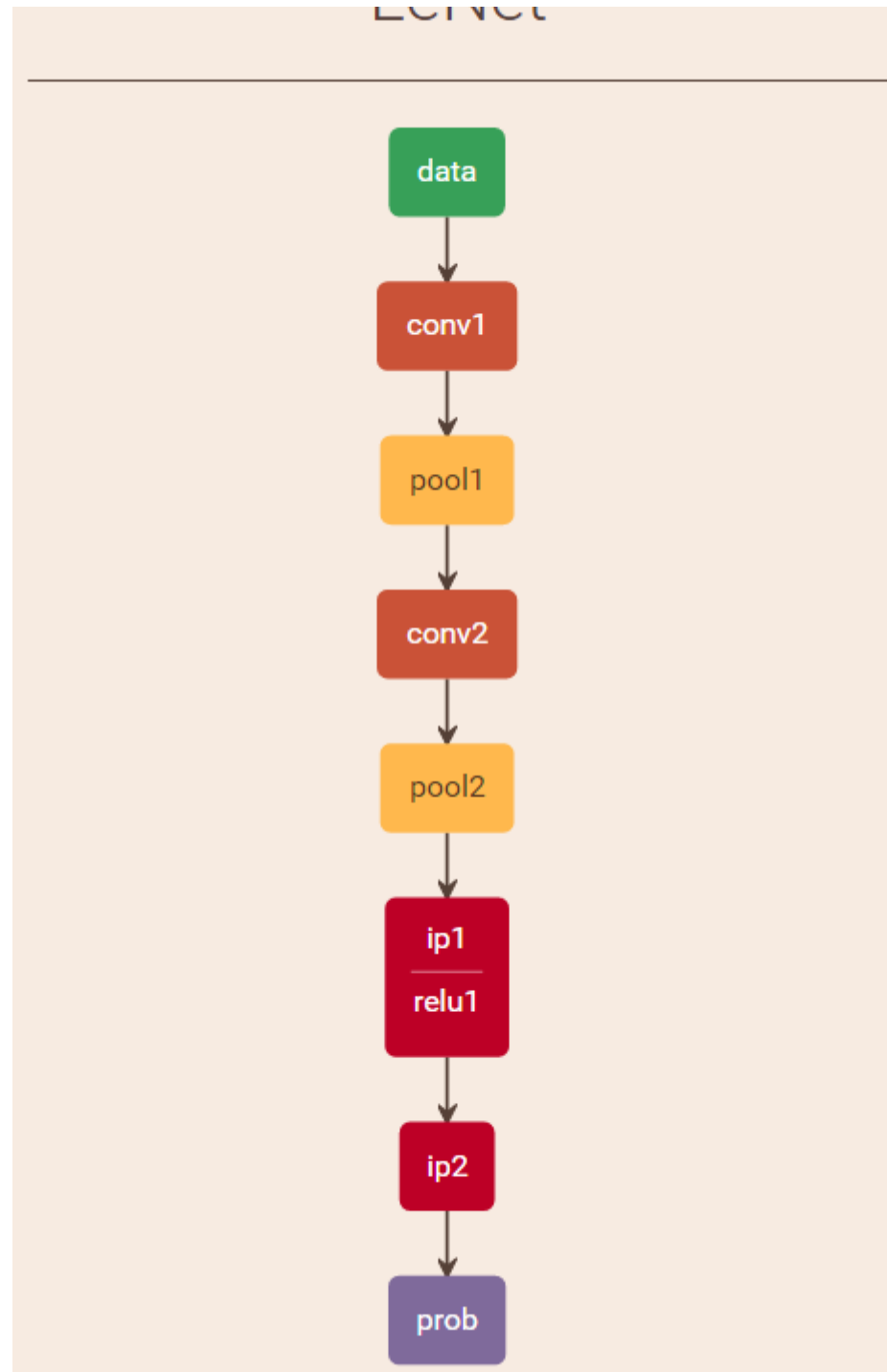
[OpenCV\\_China](#)  
[Subversion China](#)

### Technical Blog

[邹宇华](#)  
[深之JohnChen](#)  
[HUNNISH](#)  
[周伟明](#)  
[superdont](#)  
[carson2005](#)  
[OpenHero](#)  
[Netman\(Linux\)](#)  
[wqvbjhc](#)  
[yang\\_xian521](#)  
[gnuhipc](#)  
[gnuhipc](#)  
[千里8848](#)  
[CVART](#)  
[tornadomeet](#)  
[gotosuc](#)  
[onezeros](#)  
[hellogv](#)  
[abcjennifer](#)  
[crzy\\_sparrow](#)

### 评论排行

[tiny-cnn开源库的使用\(MI](#) (121)  
[Windows7 32位机上 , O](#) (120)  
[Ubuntu 14.04 64位机上7](#) (89)



人脸识别引擎SeetaFace	(67)
tesseract-ocr3.02字符识	(64)
卷积神经网络(CNN)的简	(56)
Windows7上使用VS2013	(49)
图像配准算法	(44)
tesseract-ocr	(42)
Windows 7 64位机上Op	(36)

## 最新评论

人脸识别引擎SeetaFaceEngine  
fengbingchun:

@yudiemiaomiao:我也看FaceInfo这个结构体,但是在源码里我没有找到对roll、p...

人脸识别引擎SeetaFaceEngine  
IceMiao433: @fengbingchun:作者给出的源码输出结构体是这样的: typedef struct Fa...

人脸识别引擎SeetaFaceEngine  
fengbingchun:

@yudiemiaomiao:这三个值在source code中应该都没有实现

人脸识别引擎SeetaFaceEngine  
IceMiao433: 博主,你好!请问是否有输出角度信息呢? roll = %f, pitch = %f, yaw = %f...

使用Caffe进行手写数字识别执行  
yuyi1005: 终于看到一篇写得又整洁又清楚的了

Caffe源码中各种依赖库的作用及  
fengbingchun: @linyong89:没安装过matio,不过看到matio中有个visual\_studio目录,里...

Caffe源码中各种依赖库的作用及  
linyong89: @fengbingchun:对 主要是想在移动端实现深度学习的论文 博主有没有matio的wi...

train时lenet\_train\_test.prototxt与识别时用到的lenet\_train\_test\_.prototxt差异:

(1)、数据层:训练时用Data,是以Imdb数据存储方式载入网络的,而识别时用MemoryData方式直接从内存载入网络;

(2)、Accuracy层:仅训练时用到,用以计算test集的准确率;

(3)、输出层Softmax/SoftmaxWithLoss层:训练时用SoftmaxWithLoss,输出loss值,识别时用Softmax输出10类数字的概率值。

4. 创建Net对象并初始化,有两种方法:一个是通过传入string类型(.prototxt文件)参数创建,一个通过传入NetParameter参数;

5. 调用Net的CopyTrainedLayersFrom函数加载在train时生成的二进制文件.caffemodel即lenet\_iter\_10000.caffemodel,有两种方法,一个是通过传入string类型(.caffemodel文件)参数,一个是通过传入NetParameter参数;

6. 获取Net相关参数在后面识别时需要用到:

(1)、通过调用Net的blob\_by\_name函数获得待识别图像所要求的通道数、宽、高;

(2)、通过调用Net的输出\_blobs函数获得输出blob的数目及大小,注:这里输出2个blob,第一个是label, count为1,第二个是prob, count为10,即表示数字识别结果的概率值。

7. 开始进行手写数字识别:

(1)、通过opencv的imread函数读入图像;

(2)、根据从Net中获得的需要输入图像的要求对图像进行颜色空间转换和缩放;

(3)、因为MNIST train时,图像为前景为白色,背景为黑色,而现在输入图像为前景为黑色,背景为白色,因此需要对图像进行取反操作;



OpenCV代码提取：resize函数的  
fengbingchun:

@wuzhiyang95\_xiamen:fb\_c  
不依赖任何第三方库，你可以单独使用它，它中大部分...

OpenCV代码提取：resize函数的  
chengwandou: 博主您好，这个  
是否能摆脱对opencv库的依赖？  
我看了您github上的项目，提到  
要opencv3....

利用OpenCV求取图像的重心

fengbingchun:

@weixin\_40235951:使用C++接  
口，可以参考  
<http://answers.opencv...>

## 阅读排行

神经网络简介

(41511)

tesseract-ocr3.02字符识

(40819)

卷积神经网络(CNN)基础

(36509)

OpenCV中resize函数五

(35100)

举例说明使用MATLAB C

(33090)

有效的rtsp流媒体测试地:

(31607)

利用cvMinAreaRect2求耳

(30215)

网页中插入VLC播放器播

(29861)

MNIST数据库介绍及转接

(28048)

Windows 7 64位机上搭建

(25227)

## 文章存档

2017年09月 (15)

(4)、将图像数据传入Net，有两种方法：一种是通过MemoryDataLayer类的Reset函数，一种是通过MemoryDataLayer类的AddMatVector函数传入Mat参数；

(5)、调用Net的ForwardPrefilled函数进行前向计算；

(6)、输出识别结果，注，前向计算完返回的Blob有两个，第二个Blob中的数据才是最终的识别结果的概率值，其中最大值的索引即是识别结果。

8. 通过lenet\_train\_test\_.prototxt文件分析各层的权值、偏置和神经元数量，共9层：

(1)、data数据层：无权值和偏置，神经元数量为 $1 \times 1 \times 28 \times 28 + 1 = 785$ ；

(2)、conv1卷积层：卷积窗大小为 $5 \times 5$ ，输出特征图数量为20，卷积窗种类为20，输出特征图大小 $24 \times 24$ ，训练参数(权值+偏置)为 $20 \times 1 \times 5 \times 5 + 20 = 520$ ，神经元数量为 $1 \times 20 \times 24 \times 24 = 11520$ ；

(3)、pool1降采样层：滤波窗大小为 $2 \times 2$ ，输出特征图数量为20，滤波窗种类为20，输出特征图大小 $12 \times 12$ ，可训练参数(权值+偏置)为 $1 \times 20 + 20 = 40$ ，神经元数量为 $1 \times 20 \times 12 \times 12 = 2880$ ；

(4)、conv2卷积层：卷积窗大小为 $5 \times 5$ ，输出特征图数量为50，卷积窗种类为 $50 \times 20$ ，输出特征图大小 $10 \times 10$ ，可训练参数(权值+偏置)为 $50 \times 20 \times 5 \times 5 + 50 = 25050$ ，神经元数量为 $1 \times 50 \times 8 \times 8 = 3200$ ；

(5)、pool2降采样层：滤波窗大小为 $2 \times 2$ ，输出特征图数量为50，滤波窗种类为50，输出特征图大小为 $4 \times 4$ ，可训练参数(权值+偏置)为 $1 \times 50 + 50 = 100$ ，神经元数量为 $1 \times 50 \times 4 \times 4 = 800$ ；

(6)、ip1全连接层：滤波窗大小为 $1 \times 1$ ，输出特征图数量为500，滤波窗种类为 $500 \times 800$ ，输出特征图大小为 $1 \times 1$ ，可训练参数(权值+偏置)为 $500 \times 800 \times 1 \times 1 + 500 = 400500$ ，神经元数量为 $1 \times 500 \times 1 \times 1 = 500$ ；

(7)、relu1层：in-placeip1；

(8)、ip2全连接层：滤波窗大小为 $1 \times 1$ ，输出特征图数量为10，滤波窗种类为 $10 \times 500$ ，输出特征图大小为 $1 \times 1$ ，可训练参数(权值+偏置)为 $10 \times 500 \times 1 \times 1 + 10 = 5010$ ，神经元数量为 $1 \times 10 \times 1 \times 1 = 10$ ；

(9)、prob输出层：神经元数量为 $1 \times 10 \times 1 \times 1 + 1 = 11$ 。



2017年08月 (9)

2017年07月 (14)

2017年06月 (21)

2017年05月 (16)

展开

精简后的手写数字识别测试代码如下：

t\_predict()

e::Caffe::set\_mode(caffe::Caffe::CPU);

t std::string param\_file{ "E:/GitCode/Caffe\_Test/test\_data/model/mnist/lenet\_train\_test\_.prototxt"

t std::string trained\_filename{ "E:/GitCode/Caffe\_Test/test\_data/model/mnist/lenet\_iter\_10000.caff"

t std::string image\_path{ "E:/GitCode/Caffe\_Test/test\_data/images/" };

i两种方法可以实例化net

. 通过传入参数类型为std::string

e::Net&lt;float&gt; caffe\_net(param\_file, caffe::TEST);

e\_net.CopyTrainedLayersFrom(trained\_filename);

. 通过传入参数类型为caffe::NetParameter

ffe::NetParameter net\_param1, net\_param2;

ffe::ReadNetParamsFromTextFileOrDie(param\_file, &amp;net\_param1);

t\_param1.mutable\_state()-&gt;set\_phase(caffe::TEST);

ffe::Net&lt;float&gt; caffe\_net(net\_param1);

ffe::ReadNetParamsFromBinaryFileOrDie(trained\_filename, &amp;net\_param2);

ffe\_net.CopyTrainedLayersFrom(net\_param2);

num\_inputs = caffe\_net.input\_blobs().size(); // 0 ??

t boost::shared\_ptr&lt;caffe::Blob&lt;float&gt;&gt; blob\_by\_name = caffe\_net.blob\_by\_name("data");

image\_channel = blob\_by\_name-&gt;channels();

image\_height = blob\_by\_name-&gt;height();

image\_width = blob\_by\_name-&gt;width();

num\_outputs = caffe\_net.num\_outputs();

t std::vector&lt;caffe::Blob&lt;float&gt;\*&gt; output\_blobs = caffe\_net.output\_blobs();

require\_blob\_index{ -1 };

t int digit\_category\_num{ 10 };

(int i = 0; i &lt; output\_blobs.size(); ++i) {

if (output\_blobs[i]-&gt;count() == digit\_category\_num)

require\_blob\_index = i;

require\_blob\_index == -1) {

```
fprintf(stderr, "ouput blob don't match\n");
return -1;

:vector<int> target{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
:vector<int> result;

(auto num : target) {
    std::string str = std::to_string(num);
    str += ".png";
    str = image_path + str;

    cv::Mat mat = cv::imread(str.c_str(), 1);
    if (!mat.data) {
        fprintf(stderr, "load image error: %s\n", str.c_str());
        return -1;
    }

    if (image_channel == 1)
        cv::cvtColor(mat, mat, CV_BGR2GRAY);
    else if (image_channel == 4)
        cv::cvtColor(mat, mat, CV_BGR2BGRa);

    cv::resize(mat, mat, cv::Size(image_width, image_height));
    cv::bitwise_not(mat, mat);

    // 将图像数据载入Net网络, 有2种方法
    boost::shared_ptr<caffe::MemoryDataLayer<float>> memory_data_layer =
        boost::static_pointer_cast<caffe::MemoryDataLayer<float>>
        et.layer_by_name("data"));
    // 1. 通过MemoryDataLayer类的Reset函数
    mat.convertTo(mat, CV_32FC1, 0.00390625);
    float dummy_label[1] {0};
    memory_data_layer->Reset((float*)(mat.data), dummy_label, 1);

    // 2. 通过MemoryDataLayer类的AddMatVector函数
    //std::vector<cv::Mat> patches{mat}; // set the patch for testing
    //std::vector<int> labels{patches.size()};
```

```
//memory_data_layer-
ector(patches, labels); // push vector<Mat> to data layer

float loss{ 0.0 };
const std::vector<caffe::Blob<float>*>& results = caffe_net.ForwardPrefilled(&loss); // Net forward
const float* output = results[require_blob_index]->cpu_data();

float tmp{ -1 };
int pos{ -1 };

fprintf(stderr, "actual digit is: %d\n", target[num]);
for (int j = 0; j < 10; j++) {
    printf("Probability to be Number %d is: %.3f\n", j, output[j]);
    if (tmp < output[j]) {
        pos = j;
        tmp = output[j];
    }
}

result.push_back(pos);

(auto i = 0; i < 10; i++)
fprintf(stderr, "actual digit is: %d, result digit is: %d\n", target[i], result[i]);

ntf(stderr, "predict finish\n");
rn 0;
```

测试结果如下：

```
C:\Windows\system32\cmd.exe
Probability to be Number 5 is: 0.000
Probability to be Number 6 is: 0.000
Probability to be Number 7 is: 0.033
Probability to be Number 8 is: 0.010
Probability to be Number 9 is: 0.010
W0401 10:28:23.659054 2760 memory_data_layer.cpp:88] MemoryData does not transform array data on Reset()
actual digit is: 9
Probability to be Number 0 is: 0.000
Probability to be Number 1 is: 0.704
Probability to be Number 2 is: 0.011
Probability to be Number 3 is: 0.189
Probability to be Number 4 is: 0.002
Probability to be Number 5 is: 0.003
Probability to be Number 6 is: 0.000
Probability to be Number 7 is: 0.048
Probability to be Number 8 is: 0.014
Probability to be Number 9 is: 0.029
actual digit is: 0, result digit is: 0
actual digit is: 1, result digit is: 1
actual digit is: 2, result digit is: 2
actual digit is: 3, result digit is: 3
actual digit is: 4, result digit is: 4
actual digit is: 5, result digit is: 5
actual digit is: 6, result digit is: 5
actual digit is: 7, result digit is: 7
actual digit is: 8, result digit is: 2
actual digit is: 9, result digit is: 1
predict finish
test success
请按任意键继续. . .
```

GitHub : [https://github.com/fengbingchun/Caffe\\_Test](https://github.com/fengbingchun/Caffe_Test)

顶

0

踩

0

[上一篇](#) Caffe中对MNIST执行train操作执行流程解析[下一篇](#) Caffe中master与windows分支差异对比及通过命令提示符编译Caffe源码操作步骤

## 相关文章推荐

- Caffe学习笔记(OCR字符识别)
- Caffe中Layer注册机制
- 【免费】深入理解Docker内部原理及网络配置--王...
- Android入门实战
- Caffe学习1-图像识别与数据可视化
- 最新鲜最详细的VS2013下配置BOOST库（版本
- SDCC 2017之区块链技术实战线上峰会--蔡栋
- 5天搞定深度学习框架Caffe
- Caffe训练源码基本流程
- Windows+VS2013下Boost1.59编译运行
- php零基础到项目实战
- VS2013调用caffe新建自己的工程详细过程
- asp+sql删除记录的一般步骤
- VS2013新建项目配置64位
- C语言及程序设计入门指导
- 如何快糙好猛地在Windows下编译CAFFE并使用其..

## 查看评论

3楼 [yuyi1005](#) 3天前 02:32发表

终于看到一篇写得又整洁又清楚的了

2楼 [陈丹Celine](#) 2017-04-05 21:54发表

长见识了

1楼 [雪吖头](#) 2017-04-05 17:19发表

不错的分享，学习啦。

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

网站客服

杂志客服

微博客服

[webmaster@csdn.net](mailto:webmaster@csdn.net)

400-660-0108

| 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

