# Caffe多标签分类

## 引言

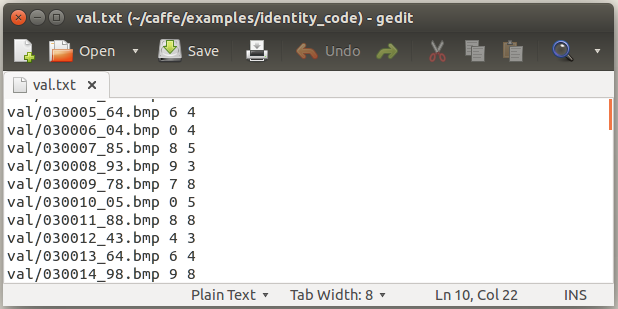
一般情况下，我们使用CNN进行分类，一幅图像只对应一个标签，例如，一幅狗的图像对应一个dog类，一幅猫的图像对应一个cat类，这在caffe中比较容易实现，一个普通的CNN网络就能达到此目的，但是有些情况下，我们需要知道，这是一只白色的狗还是黄色的狗，一只黑色的猫还是金色的猫，再例如，一张验证码，往往包含多个数字或字母，这就需要一幅图像输出多个标签。对于验证码的情况，一幅包含4个0~9的数字的验证码图签我，如果以单个标签输出，而有10×10×10×10=10000个类别，但是如果对输出4个连接层，让第一个全连接层学习第一个数字，第二个全连接层学习第二个数字，如此这样，只需要训练4个全连接层，每个全连接层负责10个类别的分类，这样就简单得多。

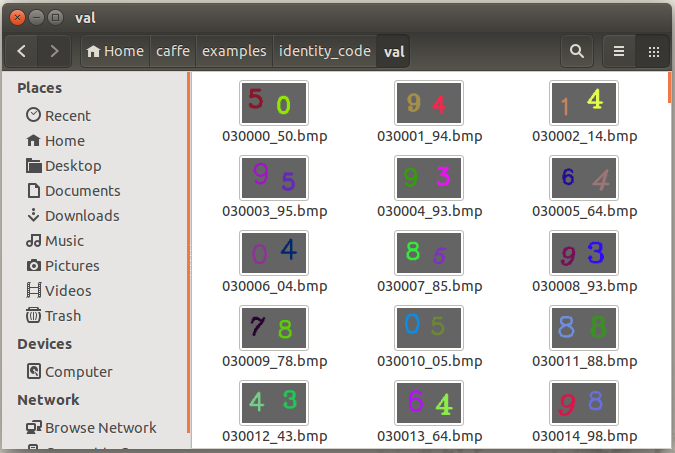
以下，我们对包含两个数字的验证码进行识别，说明多标签的分类。

## 生成图像数据

使用程序生成31000幅二维码图像，其中30000幅图像用于训练，1000幅图像用于测试。并生成label清单train.txt和val.txt



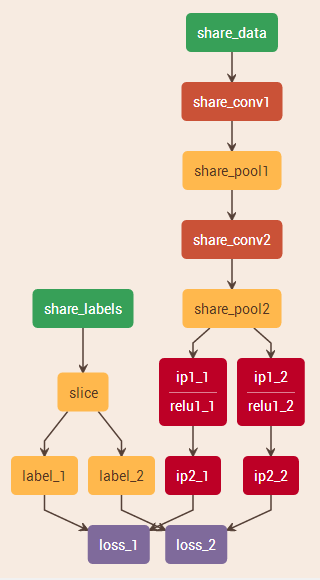




## 生成lmdb数据

在工程目录中，创建train\_lmdb和val\_lmdb文件夹，用于存放lmdb数据，根据实际情况修改5.2中的程序相应文件夹目录，将训练和测试图像及标签转换成LMDB数据。

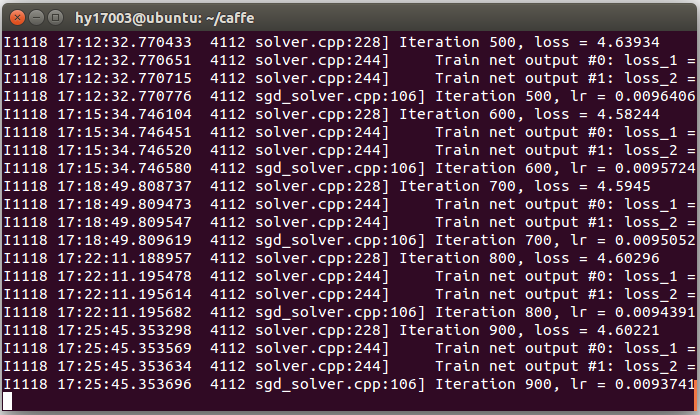
## 网络设计



分别将数据和标签以Data的形式作为输入，数据层经卷积池化后开始分支，输入到两个全连接层，而标签层输入到slice层，将两个标签分离，与全连接层一起输入到两个loss层。

编写网络清单文件和solve文件及训练脚本。

## 训练



## 程序清单

### 生成二维码图像

#include "stdafx.h"

#include <opencv2\opencv.hpp>

#include <fstream>

using namespace std;

using namespace cv;

RNG rng;

int font[7] = { 0, 2, 3, 4, 6, 7, 16 };

double scale[5] = { 0.85, 0.95, 1.0, 1.05, 1.15 };

void printNum(Mat im, int n, int x, int y)

{

int r = rng.uniform(0, 255);

int g = rng.uniform(0, 255);

int b = rng.uniform(0, 255);

while (r == 100 || g == 100 || b == 100)

{

r = rng.uniform(0, 255);

g = rng.uniform(0, 255);

b = rng.uniform(0, 255);

}

int fidx = rng.uniform(0, 7);

int sidx = rng.uniform(0, 5);

char buf[10];

sprintf\_s(buf, "%d", n);

string text(buf);

putText(im, text, Point(x+1, y), font[fidx], scale[sidx], Scalar(b, g, r), 2);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

ofstream ocout;

ocout.open("F:\\image\_dataset\\numbers\\train.txt");

for (int i = 0; i < 31000; i++)

{

Mat im(Size(80, 50), CV\_8UC3, Scalar(100, 100, 100));

int n1 = rng.uniform(0, 10);

int n2 = rng.uniform(0, 10);

int dx = rng.uniform(-5, 5);

int dy = rng.uniform(-5, 5);

printNum(im,n1,10+dx,35+dy);

dx = rng.uniform(-5, 5);

dy = rng.uniform(-5, 5);

printNum(im, n2, 45 + dx, 35 + dy);

char buf[100];

sprintf\_s(buf, "F:\\image\_dataset\\numbers\\train\\%06d\_%d%d.bmp",i,n1,n2);

string filename(buf);

imwrite(filename, im);

ocout << filename << " " << n1 <<" "<< n2 << endl;

}

ocout.close();

return 0;

}

### 生成多标签LMDB数据

# coding=utf-8

import os

import lmdb

from PIL import Image

import numpy as np

import sys

import caffe

caffe\_root = '/home/caffe'

image\_path = '/home/hy17003/caffe/examples/identity\_code/train/'

data\_lmdb = '/home/hy17003/caffe/examples/identity\_code/train\_lmdb/train\_data\_lmdb'

label\_lmdb = '/home/hy17003/caffe/examples/identity\_code/train\_lmdb/train\_label\_lmdb'

sample\_list\_txt = '/home/hy17003/caffe/examples/identity\_code/train.txt'

sys.path.insert(0, caffe\_root + '/python')

####################data(images)############################

# get imageFileList

file\_list = os.listdir(image\_path)

#your data lmdb path

in\_db=lmdb.open(data\_lmdb,map\_size=int(1e12))

with in\_db.begin(write=True) as in\_txn:

for in\_idx,in\_ in enumerate(file\_list):

im\_file=image\_path + in\_

im=Image.open(im\_file)

im = im.resize((50,100),Image.BILINEAR)

im=np.array(im)

im=im[:,:,::-1]

im=im.transpose((2,0,1))

im\_dat=caffe.io.array\_to\_datum(im)

in\_txn.put('{:0>10d}'.format(in\_idx),im\_dat.SerializeToString())

print 'data train: {} [{}/{}]'.format(in\_, in\_idx+1, len(file\_list))

del im\_file, im, im\_dat

in\_db.close()

print 'data(images) are done!'

file\_list = os.listdir(image\_path)

######data of label################

#txt with labels eg. (0001.jpg 2 5)

file\_input=open(sample\_list\_txt,'r')

label1\_list=[]

label2\_list=[]

for line in file\_input:

content=line.strip()

content=content.split(' ')

label1\_list.append(int(content[1]))

label2\_list.append(int(content[2]))

del content

file\_input.close()

#your labels lmdb path

in\_db=lmdb.open(label\_lmdb,map\_size=int(1e12))

with in\_db.begin(write=True) as in\_txn:

for in\_idx,in\_ in enumerate(file\_list):

target\_label=np.zeros((2,1,1)) # 2种lable

target\_label[0,0,0]=label1\_list[in\_idx]

target\_label[1,0,0]=label2\_list[in\_idx]

label\_data=caffe.io.array\_to\_datum(target\_label)

in\_txn.put('{:0>10d}'.format(in\_idx),label\_data.SerializeToString())

print 'label train: {} [{}/{}]'.format(in\_, in\_idx+1, len(file\_list))

del target\_label, label\_data

in\_db.close()

print 'labels are done!'

### 网络清单

name: "multi\_label\_demo"

#数据层

layer {

name: "share\_data"

type: "Data"

top: "share\_data"

include {

phase: TRAIN

}

transform\_param {

scale: 0.00390625

}

data\_param {

source: "examples/identity\_code/train\_lmdb/train\_data\_lmdb"

batch\_size: 64

backend: LMDB

}

}

layer {

name: "share\_data"

type: "Data"

top: "share\_data"

include {

phase: TEST

}

transform\_param {

scale: 0.00390625

}

data\_param {

source: "examples/identity\_code/val\_lmdb/val\_data\_lmdb"

batch\_size: 64

backend: LMDB

}

}

#标签层

layer {

name: "share\_labels"

type: "Data"

top: "share\_labels"

include {

phase: TRAIN

}

data\_param {

source: "examples/identity\_code/train\_lmdb/train\_label\_lmdb"

batch\_size: 64

backend: LMDB

}

}

layer {

name: "share\_labels"

type: "Data"

top: "share\_labels"

include {

phase: TEST

}

data\_param {

source: "examples/identity\_code/val\_lmdb/val\_label\_lmdb"

batch\_size: 64

backend: LMDB

}

}

layer {

name: "slice"

type: "Slice"

bottom: "share\_labels"

top: "label\_1"

top: "label\_2"

slice\_param {

axis: 1

slice\_point: 1

}

}

#共用的卷积池化层

layer {

name: "share\_conv1"

type: "Convolution"

bottom: "share\_data"

top: "share\_conv1"

param {

lr\_mult: 1

}

param {

lr\_mult: 2

}

convolution\_param {

num\_output: 20

kernel\_size: 5

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

layer {

name: "share\_pool1"

type: "Pooling"

bottom: "share\_conv1"

top: "share\_pool1"

pooling\_param {

pool: MAX

kernel\_size: 2

stride: 2

}

}

layer {

name: "share\_conv2"

type: "Convolution"

bottom: "share\_pool1"

top: "share\_conv2"

param {

lr\_mult: 1

}

param {

lr\_mult: 2

}

convolution\_param {

num\_output: 50

kernel\_size: 5

stride: 1

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

layer {

name: "share\_pool2"

type: "Pooling"

bottom: "share\_conv2"

top: "share\_pool2"

pooling\_param {

pool: MAX

kernel\_size: 2

stride: 2

}

}

#第一个全连接层分支

layer {

name: "ip1\_1"

type: "InnerProduct"

bottom: "share\_pool2"

top: "ip1\_1"

param {

lr\_mult: 1

}

param {

lr\_mult: 2

}

inner\_product\_param {

num\_output: 500

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

layer {

name: "relu1\_1"

type: "ReLU"

bottom: "ip1\_1"

top: "ip1\_1"

}

layer {

name: "ip2\_1"

type: "InnerProduct"

bottom: "ip1\_1"

top: "ip2\_1"

param {

lr\_mult: 1

}

param {

lr\_mult: 2

}

inner\_product\_param {

num\_output: 10

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

layer {

name: "accuracy\_1"

type: "Accuracy"

bottom: "ip2\_1"

bottom: "label\_1"

top: "accuracy\_1"

include {

phase: TEST

}

}

layer {

name: "loss\_1"

type: "SoftmaxWithLoss"

bottom: "ip2\_1"

bottom: "label\_1"

top: "loss\_1"

}

#第二个全连接层分支

layer {

name: "ip1\_2"

type: "InnerProduct"

bottom: "share\_pool2"

top: "ip1\_2"

param {

lr\_mult: 1

}

param {

lr\_mult: 2

}

inner\_product\_param {

num\_output: 500

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

layer {

name: "relu1\_2"

type: "ReLU"

bottom: "ip1\_2"

top: "ip1\_2"

}

layer {

name: "ip2\_2"

type: "InnerProduct"

bottom: "ip1\_2"

top: "ip2\_2"

param {

lr\_mult: 1

}

param {

lr\_mult: 2

}

inner\_product\_param {

num\_output: 10

weight\_filler {

type: "xavier"

}

bias\_filler {

type: "constant"

}

}

}

layer {

name: "accuracy\_2"

type: "Accuracy"

bottom: "ip2\_2"

bottom: "label\_2"

top: "accuracy\_2"

include {

phase: TEST

}

}

layer {

name: "loss\_2"

type: "SoftmaxWithLoss"

bottom: "ip2\_2"

bottom: "label\_2"

top: "loss\_2"

}

## solve文件

# The train/test net protocol buffer definition

net: "examples/identity\_code/train\_val.prototxt"

test\_iter: 100

test\_interval: 1000

base\_lr: 0.01

momentum: 0.9

weight\_decay: 0.0005

lr\_policy: "inv"

gamma: 0.0001

power: 0.75

display: 100

max\_iter: 20000

snapshot: 5000

snapshot\_prefix: "examples/identity\_code/multi\_label"

solver\_mode: CPU