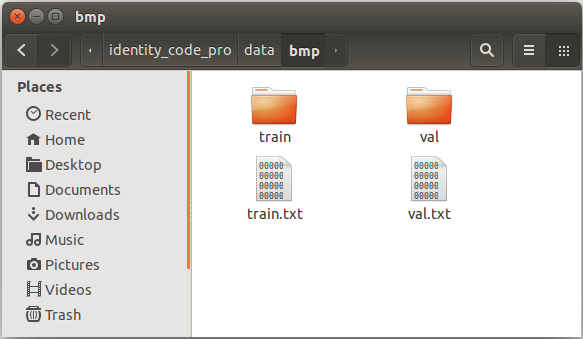
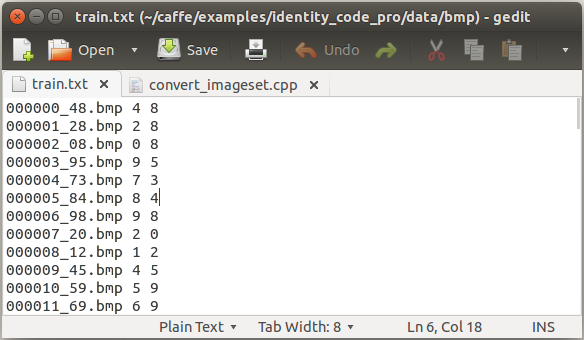
# Caffe多标签分类（二）

## 准备数据





## 修改convert\_imageset.cpp

因为图像列表有为0001.bmp 1 2这样的形式，注意代码中红色加粗的部分

//#ifdef MULTILABEL

#include <algorithm>

#include <fstream> // NOLINT(readability/streams)

#include <string>

#include <utility>

#include <vector>

#include "boost/scoped\_ptr.hpp"

#include "gflags/gflags.h"

#include "glog/logging.h"

#include "caffe/proto/caffe.pb.h"

#include "caffe/util/db.hpp"

#include "caffe/util/format.hpp"

#include "caffe/util/io.hpp"

#include "caffe/util/rng.hpp"

using namespace caffe; // NOLINT(build/namespaces)

using std::pair;

using boost::scoped\_ptr;

DEFINE\_bool(gray, false,

"When this option is on, treat images as grayscale ones");

DEFINE\_bool(shuffle, false,

"Randomly shuffle the order of images and their labels");

DEFINE\_string(backend, "lmdb",

"The backend {lmdb, leveldb} for storing the result");

DEFINE\_int32(resize\_width, 0, "Width images are resized to");

DEFINE\_int32(resize\_height, 0, "Height images are resized to");

DEFINE\_bool(check\_size, false,

"When this option is on, check that all the datum have the same size");

DEFINE\_bool(encoded, false,

"When this option is on, the encoded image will be save in datum");

DEFINE\_string(encode\_type, "",

"Optional: What type should we encode the image as ('png','jpg',...).");

int main(int argc, char\*\* argv) {

#ifdef USE\_OPENCV

::google::InitGoogleLogging(argv[0]);

// Print output to stderr (while still logging)

FLAGS\_alsologtostderr = 1;

#ifndef GFLAGS\_GFLAGS\_H\_

namespace gflags = google;

#endif

gflags::SetUsageMessage("Convert a set of images to the leveldb/lmdb\n"

"format used as input for Caffe.\n"

"Usage:\n"

" convert\_imageset [FLAGS] ROOTFOLDER/ LISTFILE DB\_NAME\n"

"The ImageNet dataset for the training demo is at\n"

" http://www.image-net.org/download-images\n");

gflags::ParseCommandLineFlags(&argc, &argv, true);

//001.jpg 1 2

**if (argc < 3) {**

gflags::ShowUsageWithFlagsRestrict(argv[0], "tools/convert\_imageset");

return 1;

}

const bool is\_color = !FLAGS\_gray;

const bool check\_size = FLAGS\_check\_size;

const bool encoded = FLAGS\_encoded;

const string encode\_type = FLAGS\_encode\_type;

std::ifstream infile(argv[2]);

std::vector<std::pair<std::string, std::vector<float> > > lines;

std::string filename;

std::string label\_count\_string = argv[5];

int label\_count = std::atoi(label\_count\_string.c\_str());

std::vector<float> label(label\_count);

while (infile >> filename)

{

for (int i = 0; i < label\_count;i++)

{

infile >> label[i];

}

lines.push\_back(std::make\_pair(filename, label));

}

if (FLAGS\_shuffle) {

// randomly shuffle data

LOG(INFO) << "Shuffling data";

shuffle(lines.begin(), lines.end());

}

LOG(INFO) << "A total of " << lines.size() << " images.";

if (encode\_type.size() && !encoded)

LOG(INFO) << "encode\_type specified, assuming encoded=true.";

int resize\_height = std::max<int>(0, FLAGS\_resize\_height);

int resize\_width = std::max<int>(0, FLAGS\_resize\_width);

// Create new DB

scoped\_ptr<db::DB> db\_image(db::GetDB(FLAGS\_backend));

scoped\_ptr<db::DB> db\_label(db::GetDB(FLAGS\_backend));

db\_image->Open(argv[3], db::NEW);

db\_label->Open(argv[4], db::NEW);

scoped\_ptr<db::Transaction> txn\_image(db\_image->NewTransaction());

scoped\_ptr<db::Transaction> txn\_label(db\_label->NewTransaction());

// Storing to db

std::string root\_folder(argv[1]);

Datum datum\_label;

Datum datum\_image;

int count = 0;

int data\_size\_label = 0;

int data\_size\_image = 0;

bool data\_size\_initialized = false;

for (int line\_id = 0; line\_id < lines.size(); ++line\_id) {

bool status;

std::string enc = encode\_type;

if (encoded && !enc.size()) {

// Guess the encoding type from the file name

string fn = lines[line\_id].first;

size\_t p = fn.rfind('.');

if (p == fn.npos)

LOG(WARNING) << "Failed to guess the encoding of '" << fn << "'";

enc = fn.substr(p);

std::transform(enc.begin(), enc.end(), enc.begin(), ::tolower);

}

status = ReadImageToDatum(root\_folder + lines[line\_id].first,

lines[line\_id].second[0], resize\_height, resize\_width, is\_color,

enc, &datum\_image);

if (status == false) continue;

datum\_label.set\_height(1);

datum\_label.set\_width(1);

datum\_label.set\_channels(label\_count);

//int count\_tmp = datum\_label.float\_data\_size();

for (int index\_label = 0; index\_label < lines[line\_id].second.size(); index\_label++)

{

float tmp\_float\_value = lines[line\_id].second[index\_label];

datum\_label.add\_float\_data(tmp\_float\_value);

}

if (check\_size) {

if (!data\_size\_initialized) {

data\_size\_label = datum\_label.channels() \* datum\_label.height() \* datum\_label.width();

data\_size\_image = datum\_image.channels() \* datum\_image.height() \* datum\_image.width();

data\_size\_initialized = true;

}

else {

const std::string& data\_label = datum\_label.data();

CHECK\_EQ(data\_label.size(), data\_size\_label) << "Incorrect data field size "

<< data\_label.size();

const std::string& data\_image = data\_image.data();

CHECK\_EQ(data\_image.size(), data\_size\_image) << "Incorrect data field size "

<< data\_image.size();

}

}

// sequential

string key\_str\_image = caffe::format\_int(line\_id, 8) + "\_" + lines[line\_id].first;

string key\_str\_label = caffe::format\_int(line\_id, 8) + "label\_" + lines[line\_id].first;

// Put in db

string out\_label;

string out\_image;

CHECK(datum\_label.SerializeToString(&out\_label));

CHECK(datum\_image.SerializeToString(&out\_image));

datum\_label.clear\_float\_data();

txn\_label->Put(key\_str\_label, out\_label);

txn\_image->Put(key\_str\_image, out\_image);

if (++count % 1000 == 0) {

// Commit db

txn\_image->Commit();

txn\_image.reset(db\_image->NewTransaction());

txn\_label->Commit();

txn\_label.reset(db\_label->NewTransaction());

LOG(INFO) << "Processed " << count << " files.";

}

}

// write the last batch

if (count % 1000 != 0) {

txn\_label->Commit();

txn\_image->Commit();

LOG(INFO) << "Processed " << count << " files.";

}

#else

LOG(FATAL) << "This tool requires OpenCV; compile with USE\_OPENCV.";

#endif // USE\_OPENCV

return 0;

}

//#endif

## 创建convert\_data\_lmdb.sh脚本文件

注意这里有两个标签，红色加粗部分设置成2

#!/usr/bin/env sh

# Create the image to lmdb inputs

TOOLS=/home/hy17003/caffe/build/tools

#图像文件的存放位置

TRAIN\_DATA\_ROOT=/home/hy17003/caffe/examples/identity\_code\_pro/data/bmp/train/

VAL\_DATA\_ROOT=/home/hy17003/caffe/examples/identity\_code\_pro/data/bmp/val/

IMAGE\_LIST\_ROOT=/home/hy17003/caffe/examples/identity\_code\_pro/data/bmp/

#LMDB文件的存放位置

ROOT\_LMDB=/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb

# Set RESIZE=true to resize the images to 50x100. Leave as false if images have

# already been resized using another tool.

#是否剪切为相同的大小

RESIZE=true

if $RESIZE; then

RESIZE\_HEIGHT=50

RESIZE\_WIDTH=100

else

RESIZE\_HEIGHT=0

RESIZE\_WIDTH=0

fi

if [ ! -d "$TRAIN\_DATA\_ROOT" ]; then

echo "Error: TRAIN\_DATA\_ROOT is not a path to a directory: $TRAIN\_DATA\_ROOT"

echo "Set the TRAIN\_DATA\_ROOT variable in create\_imagenet.sh to the path" \

"where the ImageNet training data is stored."

exit 1

fi

if [ ! -d "$VAL\_DATA\_ROOT" ]; then

echo "Error: VAL\_DATA\_ROOT is not a path to a directory: $VAL\_DATA\_ROOT"

echo "Set the VAL\_DATA\_ROOT variable in create\_imagenet.sh to the path" \

"where the ImageNet validation data is stored."

exit 1

fi

echo "Creating train lmdb..."

GLOG\_logtostderr=1 $TOOLS/convert\_imageset \

--resize\_height=$RESIZE\_HEIGHT \

--resize\_width=$RESIZE\_WIDTH \

--shuffle \

$TRAIN\_DATA\_ROOT \

$IMAGE\_LIST\_ROOT/train.txt \

$ROOT\_LMDB/train\_image \

$ROOT\_LMDB/train\_label \

**2**

echo "Creating val lmdb..."

GLOG\_logtostderr=1 $TOOLS/convert\_imageset \

--resize\_height=$RESIZE\_HEIGHT \

--resize\_width=$RESIZE\_WIDTH \

--shuffle \

$VAL\_DATA\_ROOT \

$IMAGE\_LIST\_ROOT/val.txt \

$ROOT\_LMDB/val\_image \

$ROOT\_LMDB/val\_label \

**2**

$TOOLS/compute\_image\_mean $ROOT\_LMDB/train\_image \

$ROOT\_LMDB/mean.binaryproto

echo "Done."

## 重新编译caffe

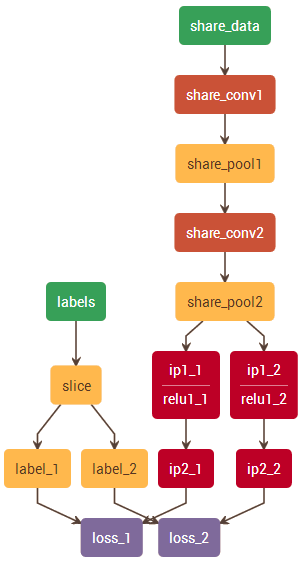
删除build目录，新建build目录，

cmake ..

make all

sudo make install

## 网络文件



name: "lenet\_pro"

#数据输入层

layer {

top: "share\_data"

name: "share\_data"

type: "Data"

data\_param {

source: "/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb/train\_image"

backend:LMDB

batch\_size: 64

}

transform\_param {

mean\_file: "/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb/mean.binaryproto"

mirror: false

}

include: {

phase: TRAIN

}

}

layer {

top: "share\_data"

name: "share\_data"

type: "Data"

data\_param {

source: "/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb/val\_image"

backend:LMDB

batch\_size: 64

}

transform\_param {

mean\_file: "/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb/mean.binaryproto"

mirror: false

}

include: {

phase: TEST

}

}

#标签输入层

layer {

name: "labels"

type: "Data"

top: "labels"

data\_param {

source: "/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb/train\_label"

backend:LMDB

batch\_size: 64

}

include {

phase: TRAIN

}

}

layer {

name: "labels"

type: "Data"

top: "labels"

data\_param {

source: "/home/hy17003/caffe/examples/identity\_code\_pro/data/lmdb/val\_label"

backend:LMDB

batch\_size: 64

}

include {

phase: TEST

}

}

layer {

name: "slice"

type: "Slice"

bottom: "labels"

top: "label\_1"

top: "label\_2"

slice\_param {

slice\_dim: 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"

}

## solver文件

# The train/test net protocol buffer definition

net: "/home/hy17003/caffe/examples/identity\_code\_pro/net/lenet\_pro.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: "/home/hy17003/caffe/examples/identity\_code\_pro/result/multi\_label"

solver\_mode: CPU

## 训练脚本文件

#!/usr/bin/env sh

set -e

/home/hy17003/caffe/build/tools/caffe train –solver = /home/hy17003/caffe/examples/identity\_code\_pro/net/solver.prototxt $@

## 训练

