shicai / Caffe_Manual

Caffe使用教程 http://weibo.com/shicaiyang

13 commits | **1** branch | **0** releases | **1** contributor

Branch: **master** | **New pull request** | **Find file** | **Clone or download**

shicai Update ReadMe.md ... Latest commit `8e0c3a8` on 19 Nov 2015

ReadMe.md	Update ReadMe.md	2 years ago
get_features.cpp	Update get_features.cpp	2 years ago
sc_load.m	Create sc_load.m	2 years ago
test_caffe_manual.cpp	Update test_caffe_manual.cpp	2 years ago
test_caffe_manual_output.log	Create test_caffe_manual_output.log	2 years ago
train_mnist_99.57.log	caffe manual and test code	2 years ago
train_mnist_99.58.log	Update train_mnist_99.58.log	2 years ago

ReadMe.md

##Caffe使用教程 by Shicai Yang (@星空下的巫师) on 2015/08/06

初始化网络

```
#include "caffe/caffe.hpp"
#include <string>
#include <vector>
using namespace caffe;

char *proto = "H:\\Models\\Caffe\\deploy.prototxt"; /* 加载CaffeNet的配置 */
Phase phase = TEST; /* or TRAIN */
Caffe::set_mode(Caffe::CPU);
// Caffe::set_mode(Caffe::GPU);
// Caffe::SetDevice(0);

//! Note: 后文所有提到的net，都是这个net
boost::shared_ptr< Net<float> > net(new caffe::Net<float>(proto, phase));
```

加载已训练好的模型

```
char *model = "H:\\Models\\Caffe\\bvlc_reference_caffenet.caffemodel";
net->CopyTrainedLayersFrom(model);
```

读取模型中的每层的结构配置参数

```
char *model = "H:\\Models\\Caffe\\bvlc_reference_caffenet.caffemodel";
NetParameter param;
```

```
ReadNetParamsFromBinaryFileOrDie(model, &param);
int num_layers = param.layer_size();
for (int i = 0; i < num_layers; ++i)
{
    // 结构配置参数:name , type , kernel size , pad , stride等
    LOG(ERROR) << "Layer " << i << ":" << param.layer(i).name() << "\t" << param.layer(i).type();
    if (param.layer(i).type() == "Convolution")
    {
        ConvolutionParameter conv_param = param.layer(i).convolution_param();
        LOG(ERROR) << "\t\tkernel size: " << conv_param.kernel_size()
            << ", pad: " << conv_param.pad()
            << ", stride: " << conv_param.stride();
    }
}
```

读取图像均值

```
char *mean_file = "H:\\Models\\Caffe\\imagenet_mean.binaryproto";
Blob<float> image_mean;
BlobProto blob_proto;
const float *mean_ptr;
unsigned int num_pixel;

bool succeed = ReadProtoFromBinaryFile(mean_file, &blob_proto);
if (succeed)
{
    image_mean.FromProto(blob_proto);
    num_pixel = image_mean.count(); /* NCHW=1x3x256x256=196608 */
    mean_ptr = (const float *) image_mean.cpu_data();
}
```

根据指定数据，前向传播网络

```
//! Note: data_ptr指向已经处理好（去均值的，符合网络输入图像的长宽和Batch Size）的数据
void caffe_forward(boost::shared_ptr< Net<float> > & net, float *data_ptr)
{
    Blob<float>* input_blobs = net->input_blobs()[0];
    switch (Caffe::mode())
    {
    case Caffe::CPU:
        memcpy(input_blobs->mutable_cpu_data(), data_ptr,
            sizeof(float) * input_blobs->count());
        break;
    case Caffe::GPU:
        cudaMemcpy(input_blobs->mutable_gpu_data(), data_ptr,
            sizeof(float) * input_blobs->count(), cudaMemcpyHostToDevice);
        break;
    default:
        LOG(FATAL) << "Unknown Caffe mode.";
    }
    net->ForwardPrefilled();
}
```

根据Feature层的名字获取其在网络中的Index

```
//! Note: Net的Blob是指，每个层的输出数据，即Feature Maps
// char *query_blob_name = "conv1";
unsigned int get_blob_index(boost::shared_ptr< Net<float> > & net, char *query_blob_name)
{
    std::string str_query(query_blob_name);
    vector< string > const & blob_names = net->blob_names();
    for( unsigned int i = 0; i != blob_names.size(); ++i )
    {
        if( str_query == blob_names[i] )
        {
            return i;
        }
```

```
    }
    LOG(FATAL) << "Unknown blob name: " << str_query;
}
```

读取网络指定Feature层数据

```
//! Note: 根据CaffeNet的deploy.prototxt文件，该Net共有15个Blob，从data一直到prob
char *query_blob_name = "conv1"; /* data, conv1, pool1, norm1, fc6, prob, etc */
unsigned int blob_id = get_blob_index(net, query_blob_name);

boost::shared_ptr<Blob<float> > blob = net->blobs()[blob_id];
unsigned int num_data = blob->count(); /* NCHW=10x96x55x55 */
const float *blob_ptr = (const float *) blob->cpu_data();
```

根据文件列表，获取特征，并存为二进制文件

详见**get_features.cpp**文件：

> 主要包括三个步骤

- 生成文件列表，格式与训练用的类似，每行一个图像 包括文件全路径、空格、标签（没有的话，可以置0）
- 根据train_val或者deploy的prototxt，改写生成feat.prototxt 主要是将输入层改为image_data层，最后加上prob和argmax（为了输出概率和Top1/5预测标签）
- 根据指定参数，运行程序后会生成若干个二进制文件，可以用MATLAB读取数据，进行分析

根据Layer的名字获取其在网络中的Index

```
//! Note: Layer包括神经网络所有层，比如，CaffeNet共有23层
// char *query_layer_name = "conv1";
unsigned int get_layer_index(boost::shared_ptr< Net<float> > & net, char *query_layer_name)
{
```

```
    std::string str_query(query_layer_name);
    vector< string > const & layer_names = net->layer_names();
    for( unsigned int i = 0; i != layer_names.size(); ++i )
    {
        if( str_query == layer_names[i] )
        {
            return i;
        }
    }
    LOG(FATAL) << "Unknown layer name: " << str_query;
}
```

读取指定Layer的权重数据

```
//! Note: 不同于Net的Blob是Feature Maps，Layer的Blob是指Conv和FC等层的Weight和Bias
char *query_layer_name = "conv1";
const float *weight_ptr, *bias_ptr;
unsigned int layer_id = get_layer_index(net, query_layer_name);
boost::shared_ptr<Layer<float> > layer = net->layers()[layer_id];
std::vector<boost::shared_ptr<Blob<float>  >> blobs = layer->blobs();
if (blobs.size() > 0)
{
    weight_ptr = (const float *) blobs[0]->cpu_data();
    bias_ptr = (const float *) blobs[1]->cpu_data();
}

//! Note: 训练模式下，读取指定Layer的梯度数据，与此相似，唯一的区别是将cpu_data改为cpu_diff
```

修改某层的Weight数据

```
const float* data_ptr;          /* 指向待写入数据的指针， 源数据指针*/
float* weight_ptr = NULL;       /* 指向网络中某层权重的指针，目标数据指针*/
unsigned int data_size;         /* 待写入的数据量 */
```

```
char *layer_name = "conv1";     /* 需要修改的Layer名字 */

unsigned int layer_id = get_layer_index(net, query_layer_name);
boost::shared_ptr<Blob<float> > blob = net->layers()[layer_id]->blobs()[0];

CHECK(data_size == blob->count());
switch (Caffe::mode())
{
case Caffe::CPU:
    weight_ptr = blob->mutable_cpu_data();
    break;
case Caffe::GPU:
    weight_ptr = blob->mutable_gpu_data();
    break;
default:
    LOG(FATAL) << "Unknown Caffe mode";
}
caffe_copy(blob->count(), data_ptr, weight_ptr);

//! Note: 训练模式下，手动修改指定Layer的梯度数据，与此相似
// mutable_cpu_data改为mutable_cpu_diff，mutable_gpu_data改为mutable_gpu_diff
```

保存新的模型

```
char* weights_file = "bvlc_reference_caffenet_new.caffemodel";
NetParameter net_param;
net->ToProto(&net_param, false);
WriteProtoToBinaryFile(net_param, weights_file);
```