系统使用说明

前言

1. 系统说明

version：1.0

*号为特别注意事项

使用时请先编译附带的 caffe

2. 系统使用方法

---

1. 在 retrieval 目录下新建目录 build, 并进入该目录

   ```
   mkdir build
   cd build
   ```

2. 在 build 目录下输入

   ```
   cmake ../
   make clean
   make retrieval && make
   ```

3. 在 bin/目录下生成有测试代码，在 lib/下为库

***依赖说明:**

1. 保证系统已经安装 openblas

2. 并且在/usr/local/lib 或者 /usr/lib 下能找到 openblas 的链接库

3. 若出现 IndexFlat* IndexIVFPQ 链接问题，请在 faiss/目录下重新编译，方式如下：

eg. make clean && make

4. 测试代码 main.cpp 与 testRead.cpp 仅供使用参考，实际运行时缺少数据文件(文件较大)，若需要请联系作者

5. 参数建议使用默认，按照 main.cpp 提供的示例进行使用，若需要修改需要了解 PQ 算法，或者联系作者咨询

6. faiss 代码有所修改，删去了不必要的部分

3. 人车属性提取使用说明

---

1. 车的属性提取代码和特征提取代码合并在一个文件中

即： ./cfeature/Feature.cpp

a) 使用 PictureAttrExtraction 函数即可从车辆图片中提取 color 和 type 属性 id

b) 使用示例见： ./test/testVehicleAttr.cpp

2. 人的属性提取代码在./personAttr/目录下

即： personAttr.cpp

a) 使用示例见: ./test/testPersonAttr.cpp

*b) 使用时注意修改./include/personAttr 文件中的 ROOT_DIR_ATTR 为 personAttrFile 目录的路径

4. 特征提取代码使用说明

---

1. 特征提取代码在./cfeature/目录下
2. 具体使用方法和车辆属性提取方法一致，仅使用函数不同

a) 具体使用方法见: ./test/testVehicleAttr.cpp

a) 人车特征提取均可以使用 PictureFeatureExtraction 函数

b) 注意,人和车的特征提取方法仅在 initNet 时不同，需要输入对应的 prototxt 文件和 model 文件

3. 基本 prototxt 和 model 文件均在./model/路径下

a) 附带有_person 文件名的均为人的特征提取所要使用的文件，_vehicle 同理为车所需

***b) 特别注意：**

使用时修改 prototxt 文件中的 source 路径，使得与程序内的 file_list 文件路径一致

5. 数据库使用说明

---

1. 本封装代码，仅支持 mysql

2. 在./include/featureSql.h 中，修改 FeatureSql()函数内的数据库的相关参数

3. 建立数据库的代码未给出：

a) 请使用 searchWithUdType(user_define_type)函数进行 mysql 的查询

b) 具体使用细节见 ./test/testSql.cpp

4. 建立数据库的列名(个数与所给属性对应)可任意，但值应该均为整数

a) 具体属性值和整数对应关系如下：

人： ./personAttrFile/att_index.txt

车： ./VehicleAttr/color_class.txt

./VehicleAttr/vehicle_type.txt

b) 可以使用 sql 封装代码中的 InitMapColor, InitMapType, InitPersonAttr 分别初始化车的颜色对应关系,车的车型对应关系，人的属性对应关系并且由 _id_map_color, _id_map_type, _id_person_attr 三个变量分别获取

eg. 1.初始化之后，string value = _id_person_attr["hair"][0];

即 hair 属性的 0 值为 value = “black”

2. string value = _id_map_color[0];

即车的 color 颜色属性 0 值对应 value=“棕”色

*3. 整体系统使用代码参考 FeatureWithAttrTest.cpp

作者:

email: slh@pku.edu.cn

一、 特征提取以及车辆属性提取

1. 相关代码文件： Feature.h, Feature.cpp
2. 名称空间： feature_index
   a) 使用方式： using namespace feature_index;
3. 主要函数说明：
   a) 初始化函数：

1. Caffe 网络文件初始化

函数原型：

```
caffe::Net<float> *InitNet(std::string proto_file, std::string proto_weight);
```

输入参数： proto_file: caffe 网络结构文件
Proto_weight: caffe 模型文件
输出参数： caffe::Net<float>*

2. 初始化 GPU 函数

```
int InitGpu(const char *CPU_MODE, int GPU_ID);
```

输入参数： CPU_MODE，可选值为，“GPU”, “CPU”
GPU_ID: GPU 模式下为 GPU 编号，CPU 模式下任意

b) 特征提取函数：

1. 提取特征

函数原型：

```
float* PictureFeatureExtraction(int count, caffe::Net<float>* _net, std::string blob_name)
```

输入参数：count: 输入的图片数量
_net: 初始化后的 Caffe 网络指针
blob_name: 提取的特征所在的层名称
输出参数：
由 TOTALBYTESIZE 设置的维数的 count 大小的浮点特征结果
即: float result[TOTALBYTESIZE * count];

2. 提取特征加属性值：

函数原型：

```
float* PictureAttrFeatureExtraction(int count, caffe::Net<float>*_net,
```

std::string feature_blob_name,
std::string Attr_color_name,
std::string Attr_type_name,
int* color_re, int* type_re);

输入参数: count, _net 参数意义与特征提取函数相同。
feature_blob_name： 所要提取的特征的层名称
Attr_color_name： 所要提取的颜色属性的层名称
Attr_type_name： 所要提取的车型属性的层名称
color_re, type_re：颜色数组和车型数组，为属性结果输出，需初始化

输出参数：
一定维数的 count 大小的图片特征结果，与特征提取结果相同

3. 仅提取属性值函数

函数原型：

void PictureAttrExtraction(int count, caffe::Net<float>*_net,
std::string Attr_color_name,
std::string Attr_type_name,
int* color_re, int* type_re);

输入参数: count, _net 参数意义与特征提取函数相同。
Attr_color_name： 所要提取的颜色属性的层名称
Attr_type_name： 所要提取的车型属性的层名称
color_re, type_re：颜色数组和车型数组，为属性结果输出，需初始化

输出参数：无

4. 使用说明：

a) 相关函数封装在 Feature 类内，主要使用步骤如下：

i. 在名称空间下初始化 Feature 类

```
Feature index = Feature();
```

ii. 将写有图片路径的文件列表放在(或输出到)模型文件 prototxt 中 data 层的 source 参数所设置的位置，如下所示

```
name: "GoogleNet"
layer {
    name: "data"
    type: "ImageData"
    top: "data"
    top: "label"
    image_data_param {
        source:"/DirToRetrieval/retrieval/model/file_list"
```

```
            batch_size: 1
            new_height: 224
            new_width: 224
        }
    }
```

iii. 初始化 caffe 网络，初始化 GPU（CPU）

```
caffe::Net<float>* net = index.InitNet("person.prototxt",
                                       "person.caffemodel");
index.InitGpu("GPU",1); //初始化 GPU，编号为 1
/// index.InitGpu("CPU", -1); //初始化 CPU
```

iv. 调用特征提取函数

```
/// 选用相应的函数完成所需功能，此处选用了提取属性的函数
index.PictureAttrExtraction(count, net, "color/classifier",
                                        "model_loss1/classifier",
                                        color_re, type_re);
```

v. color_re, type_re 即结果。

1. 提取属性前需要初始化：

   a) int* color_re = new int[count];

   b) int* type_re = new int[count];

二、 检索

1. 相关代码文件： Retrieval.h, Retrieval.cpp

2. 名称空间： retrieval

   b) 使用方式： using namespace retrieval;

3. 主要函数说明：

   a) 类初始化函数：

      1. FeatureIndex 类初始化

函数原型：

FeatureIndex(long long NumOfData);

或 FeatureIndex(int dimension, int nlist, int groups, int nbits);

输入参数： NumOfData: 索引的图片数量

dimension： 数据维度，nlist，粗糙聚类中心数，

groups: 高维特征分组数

nbits: PQ 量化编码码长

输出参数： 无

b) 索引操作

1. 增删索引数据

函数原型：

```
void AddItemToFeature(float* data);
void DeleteItemFromFeature(int id);

// 下面两个函数为批处理

void AddItemList(int numOfdata, float* data);
void DeleteItemList(int beginId, int numOfdata);
```

输入参数： data: 输入的图片特征数据，

id: 所要删除的图片 id，为图片的索引编号

numOfdata: 批处理所需的图片的数量

beginId: 批处理时所需的起始图片索引编号 id

输出参数：无

2. 索引数据串行化：

函数原型：

```
void WriteIndexToFile(const char* saveFileName);
void ReadIndexFromFile(const char* saveFileName);
```

输入参数： saveFileName：保存的文件名/读取的文件名

输出参数： 无，读入索引数据会自动初始化索引变量

3. 检索函数

函数原型：

```
void RetievalIndex(int numOfquery, float* nquery, int Ktop, long* index,
                                                    float* Distance);
```

输入参数: numOfquery: 输入query的数量

nquery： 与输入数量相等的图片特征数据

Ktop： 返回前 K 相似的结果

index, Distance: 输出索引编号与距离结果，该参数需要初始化

输出参数：无

4. 使用说明：

a) 相关函数封装在 FeatureIndex 类内，主要使用参见

./src/main/testRetrieval.cpp // 检索示例，存储示例

./src/main/testRead.cpp // 索引读取示例

三、 人的属性提取

1. 相关代码文件： personAttr.h, personAttr.cpp
2. 名称空间： attrOfPerson
   a） 使用方式： using namespace attrOfPerson;
3. 主要函数说明
   a) 初始化函数
      1. PersonAttr 类初始化函数
      函数原型:
      PersonAttr(std::string thr);
      输入参数: thr: 计算参数文件名
      固定值为：./personAttrFile/thr.txt
   b) 获取属性结果函数
      1. 获取属性函数
      函数原型：
      float* get_att(float *fea, int dic_size, int att_dic_size, int fea_size,
      int att_size, int img_num,
      char *database);
      输入参数： fea： 输入图片的特征数组
      fea_size： 输入图片的维数
      img_num: 输入图片的数量
      其他值默认设置见 testPersonAttr.cpp
      输出参数：
      二分类结果，数组大小由 att_size 指定，该函数需要结合 peta_att_union 函数一起使用。具体见代码示例。
4. 使用说明
   具体见代码

./src/test/testPersonAttr.cpp

四、 数据库操作

1. 相关代码文件： featureSql.h, featureSql.cpp

2. 名称空间： FeatureSQL

   a） 使用方法： using namespace FeatureSQL;

3. 主要函数说明

   a) 初始化函数

   1. FeatureSql 类初始化函数

   函数原型:

   inline FeatureSql();

   输入参数和输出参数：无，使用时修改函数内连接数据库的相关参数

   b) 数据库检索函数

   1. 自定义内容检索

   函数原型：

```
int* searchWithUdType(std::string table, std::vector<std::string> typeName,
                                          std::vector<std::string> relation,
                                          std::vector<int>id, int& row_count);
```

   输入参数:

   table: 所查表的表名
   typeName, 自定义的表列名(即存储的属性项)
   relation, 多个属性约束间的逻辑关系，and, or
   id, 属性值对应的 Id 值
   row_count，返回结果的数量

   输出参数：

   符合属性条件的图片数据的索引编号数组

4. 使用说明

   具体见代码

   ./src/test/testSql.cpp