模型保存与C++调用

Tensorflow

林楚铭

cmlin17@fudan.edu.cn

简介

- 保存训练好的模型
- 编译Tensorflow C++ library
- 使用Tensorflow C++ API运行模型

模型打包

- 将Model声明好，导入训练好的ckpt，保存sess，导出模型
- 注意：模型内的所有操作必须均是由tf提供，例如不能含有tf.py_func

```
saver1.restore(sess, 'tfmodel/checkpoint.ckpt-9000')
saver2.restore(sess, 'model/baseline.ckpt-28000')

builder = tf.saved_model.builder.SavedModelBuilder('pack')
builder.add_meta_graph_and_variables(sess, ['vgg16'])
builder.save()
```

编译C++ Library

- 参考：https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/cmake
- 准备好源码以及各种依赖工具（vs2015,python3.5,swig,cmake)
- cmake .. -A x64 -DCMAKE_BUILD_TYPE=Release ^
- -DSWIG_EXECUTABLE=E:/tflib/swigwin/swig.exe ^
- -DPYTHON_EXECUTABLE=E:/tflib/Anaconda3/python.exe ^
- -DPYTHON_LIBRARIES=E:/tflib/Anaconda3/libs/python35.lib ^
- -Dtensorflow_ENABLE_GPU=ON ^
- -DCUDNN_HOME="C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0" ^
- -Dtensorflow_BUILD_PYTHON_BINDINGS=OFF ^
- -Dtensorflow_ENABLE_GRPC_SUPPORT=OFF ^
- -Dtensorflow_BUILD_SHARED_LIB=ON
- 一些引用路径请根据实际情况修改

编译C++ Library

- 添加MSBuild指令
- “C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat"
- 编译代码
- MSBuild /p:Configuration=Release ^
- /p:Platform=x64 ^
- /p:PreferredToolArchitecture=x64 ALL_BUILD.vcxproj ^
- /filelogger
- 大约三个小时后就编译好了，会得到tensorflow.dll以及tensorflow.lib

VS C++ 工程依赖设置

- 首先将工程平台更换为Release x64
- include目录：
- E:\tflib\tensorflow\tensorflow\contrib\cmake\build
- E:\tflib\tensorflow\tensorflow\contrib\cmake\build\external\eigen_archive
- E:\tflib\tensorflow\tensorflow\contrib\cmake\build\protobuf\src\protobuf\src
- E:\tflib\tensorflow\third_party\eigen3
- E:\tflib\tensorflow
- lib目录：
- E:\tflib\tensorflow\tensorflow\contrib\cmake\build\Release
- 链接器输入添加
- tensorflow.lib

C++运行模型实例

```cpp
#include <vector>
#include <eigen/Dense>
#include <string>
#include "tensorflow/core/public/session.h"
#include "tensorflow/cc/ops/standard_ops.h"
#include "tensorflow/cc/saved_model/loader.h"
#include <opencv2/opencv.hpp>

class classifier {
    tensorflow::SavedModelBundle bundle;
    const int _isize = 224;
    const std::string model_dir = "model";
    cv::Mat probs;
    cv::Mat feats;
public:
    classifier();

    static classifier& getInstance() {
        static classifier instance;
        return instance;
    }

    void run(std::vector<cv::Mat>&);

    cv::Mat getProbs();

    cv::Mat getFeatures();

};
```

```cpp
classifier::classifier() {
    if (!bundle.session) {
        SessionOptions sess_opt;
        RunOptions run_opt;
        sess_opt.config.mutable_gpu_options()->set_allow_growth(true);
        TF_CHECK_OK(LoadSavedModel(sess_opt, run_opt, model_dir, { "vgg16" }, &bundle));
    }
}
```

```cpp
Tensor image_batch(DT_FLOAT, TensorShape({ N, _isize, _isize, 3 }));
auto input = image_batch.tensor<float, 4>();
for (int i = 0; i < N; i++) {
    Mat tmp;
    resize(inputs[i], tmp, Size(_isize, _isize), 0, 0, INTER_LINEAR);
    for (int r = 0; r < _isize; r++) {
        for (int c = 0; c < _isize; c++) {
            for (int d = 0; d < 3; d++) {
                input(i, r, c, d) = tmp.at<Vec3b>(r, c)[d];
            }
        }
    }
}
vector<Tensor> outputs;
TF_CHECK_OK(bundle.session->Run({ { "images:0", image_batch }, {"reid_images:0", image_batch} },
{ "prob:0", "resnet_model/Reshape:0" }, {}, &outputs));
```