

Visual Studio libtorch使用教程

预备软件

- libtorch: cpu或gpu版本均可，根据本机环境选择，但要注意下载nightly build版，因stable版头文件引用有错误。
- visual studio: 本教程在vs2015与2019版本下均调试通过，根据本机环境选择。
- *cmake: 可选，当教程中其他方法均失败情况下尝试。

使用流程

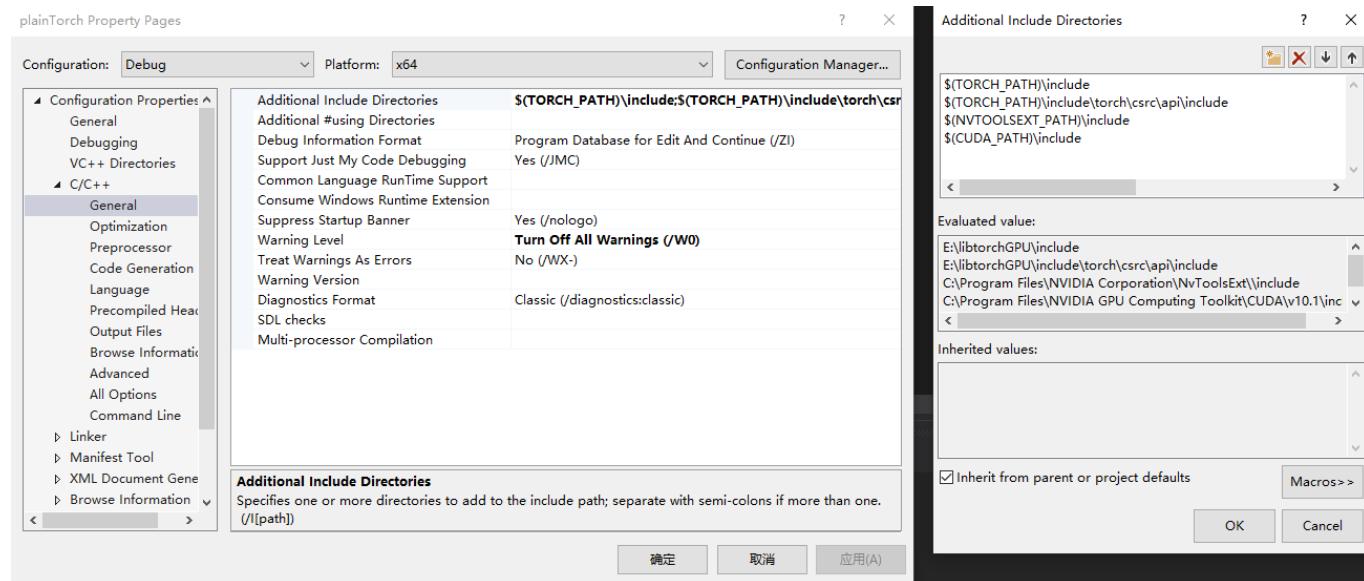
1. 下载对应版本libtorch并解压到对应目录，如在本机为E:\libtorchGPU
2. 配置环境变量TORCH_LIBRARY为libtorch解压目录E:\libtorchGPU
3. 打开visual studio创建新项目，添加源代码

```
#include <iostream>
#include <torch/torch.h>

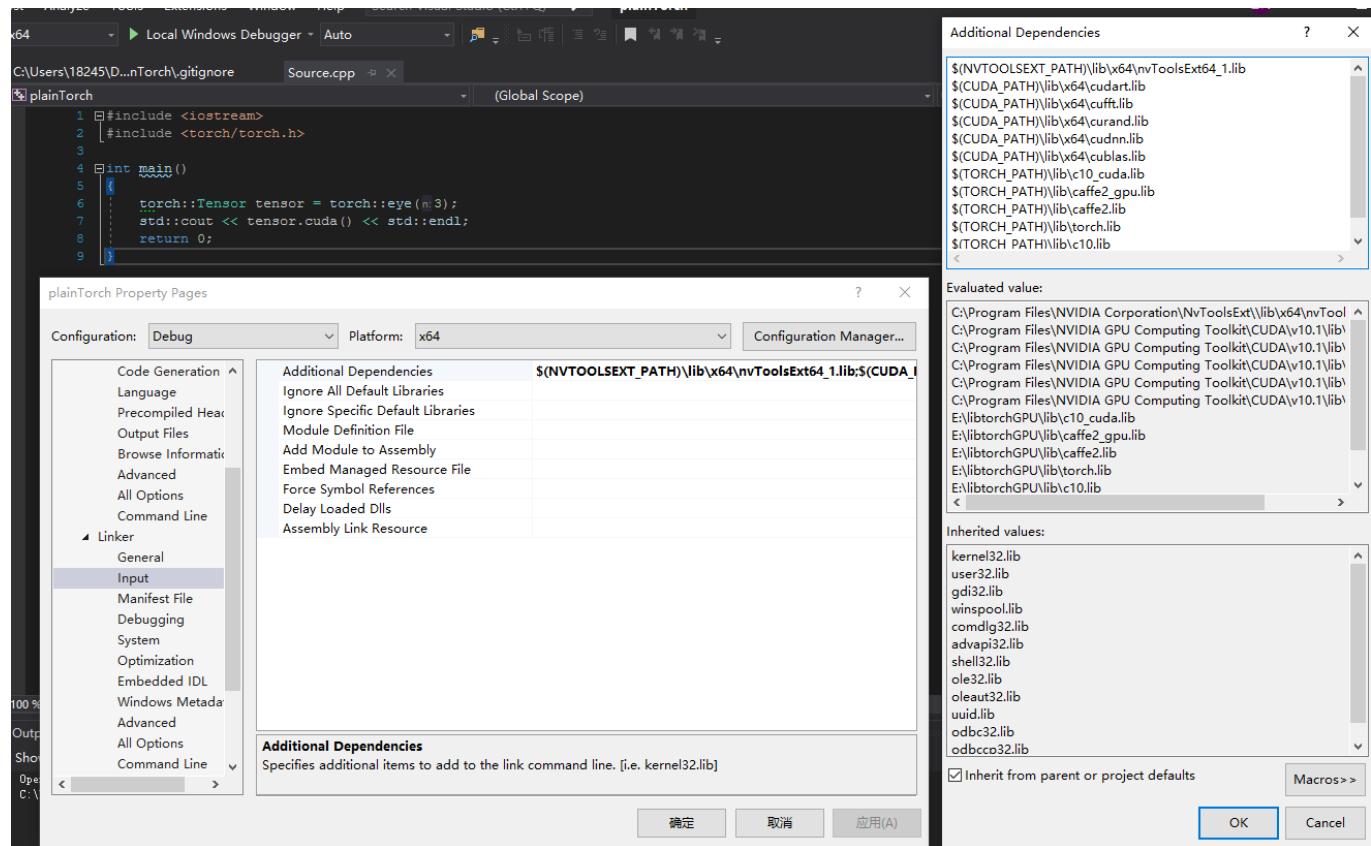
using namespace std;

int main()
{
    torch::Tensor tensor = torch::eye(3);
    std::cout << tensor.cuda() << std::endl;
    return 0;
}
```

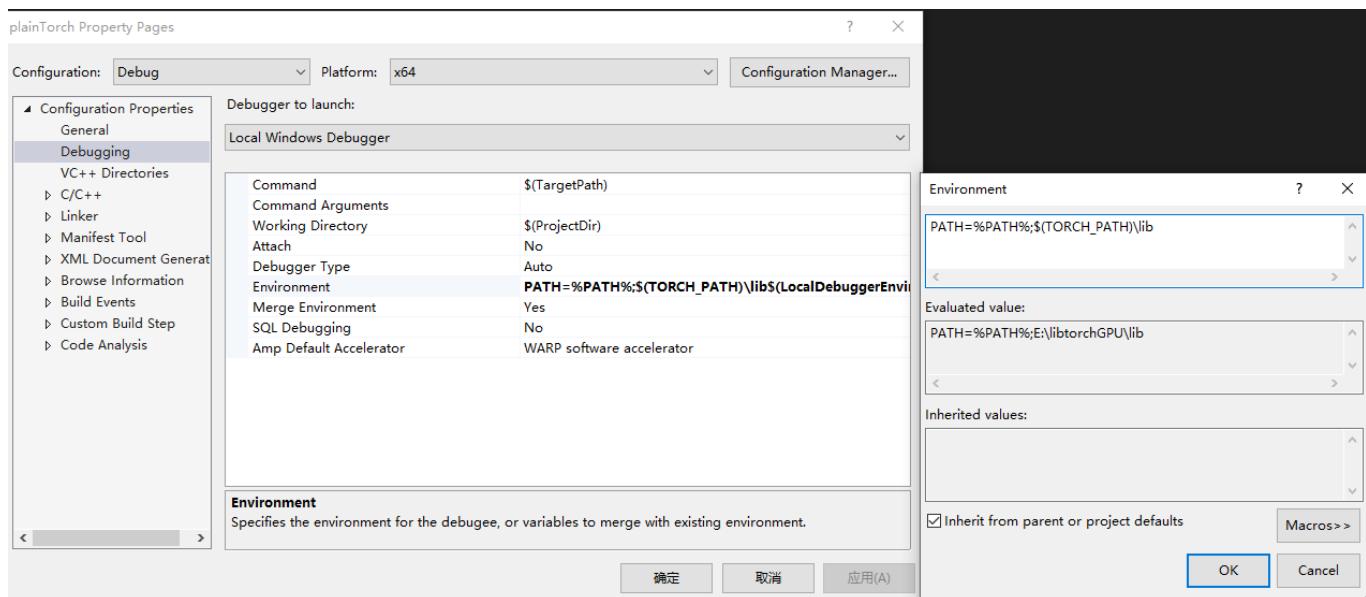
3. 右键项目属性，总共有三项需要配置：



```
$(TORCH_PATH)\include;$(TORCH_PATH)\include\torch\csrc\api\include;$(NVTOOLSEXT_PATH)\include;$(CUDA_PATH)\include;%AdditionalIncludeDirectories)
```



```
$(NVTOOLSEXT_PATH)\lib\x64\nvToolsExt64_1.lib
$(CUDA_PATH)\lib\x64\cudart.lib
$(CUDA_PATH)\lib\x64\cufft.lib
$(CUDA_PATH)\lib\x64\curand.lib
$(CUDA_PATH)\lib\x64\cudnn.lib
$(CUDA_PATH)\lib\x64\cublas.lib
$(TORCH_PATH)\lib\c10_cuda.lib
$(TORCH_PATH)\lib\caffe2_gpu.lib
$(TORCH_PATH)\lib\caffe2.lib
$(TORCH_PATH)\lib\torch.lib
$(TORCH_PATH)\lib\c10.lib
```



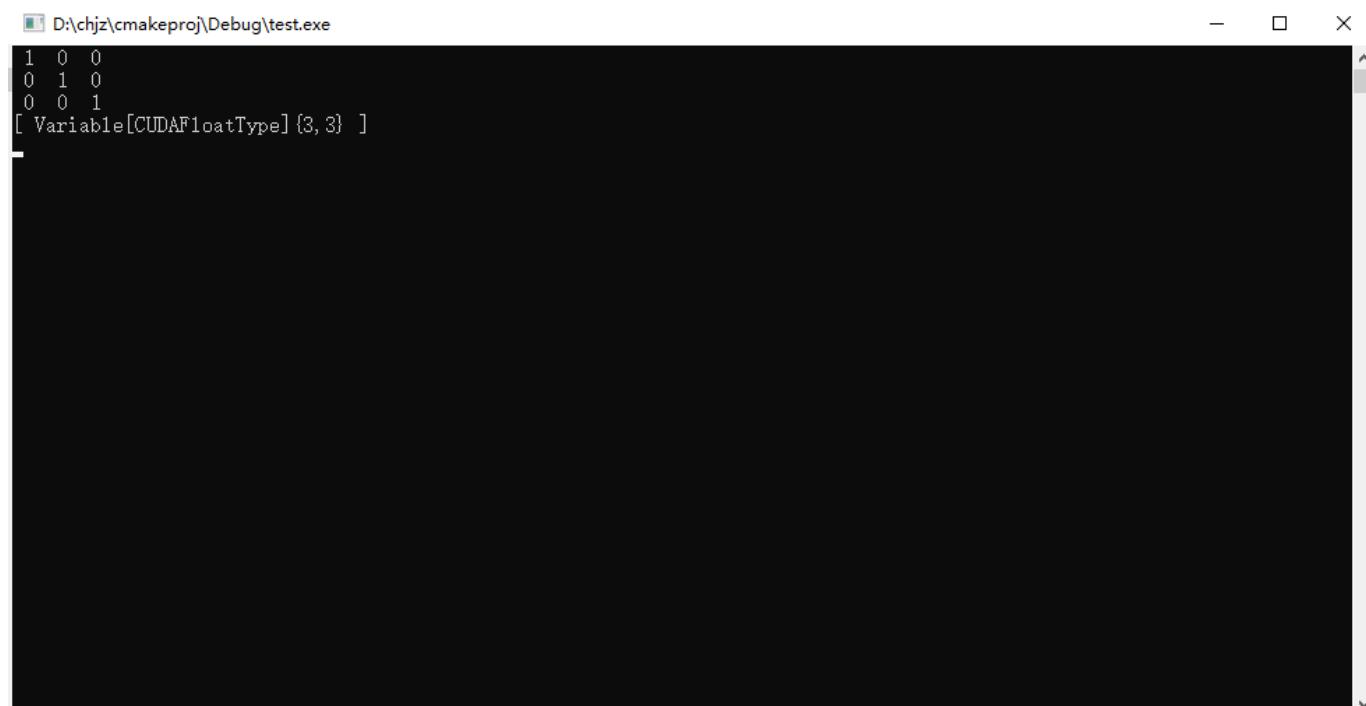
PATH=%PATH%;\$(TORCH_PATH)\lib

配置1设置引用目录，配置2设置引用静态库，配置3设置d11引用路径，若使用的是cpu版本可以将配置中有关cuda与nvtoolsext的部分全部删除。

4. 如果使用的vs版本为2017及以上还需要配置C/C++ -> Conformance Mode为No。

*在已有项目里添加libtorch同样需要以上三步。

运行成功！



在上述方法失效的情况下从cmake重新创建工程

以上方法依赖于libtorch现有的项目结构，故可能不适用于未来版本。在这种时候需要从cmake项目开始重新生成sln项目。

检查cmake版本与功能

输入命令`cmake -h`，检查generator选项，样例输出如下图

```

--help-module-list [<f>]      = List modules with help available and exit.
--help-modules [<f>]          = Print cmake-modules manual and exit.
--help-policy <cmp> [<f>]     = Print help for one policy and exit.
--help-policy-list [<f>]       = List policies with help available and exit.
--help-policies [<f>]         = Print cmake-policies manual and exit.
--help-property <prop> [<f>]   = Print help for one property and exit.
--help-property-list [<f>]    = List properties with help available and
                               exit.
--help-properties [<f>]        = Print cmake-properties manual and exit.
--help-variable var [<f>]      = Print help for one variable and exit.
--help-variable-list [<f>]    = List variables with help available and exit.
--help-variables [<f>]        = Print cmake-variables manual and exit.

Generators

The following generators are available on this platform (* marks default):
* Visual Studio 16 2019      = Generates Visual Studio 2019 project files.
                             Use -A option to specify architecture.
Visual Studio 15 2017 [arch] = Generates Visual Studio 2017 project files.
                             Optional [arch] can be "Win64" or "ARM".
Visual Studio 14 2015 [arch] = Generates Visual Studio 2015 project files.
                             Optional [arch] can be "Win64" or "ARM".
Visual Studio 12 2013 [arch] = Generates Visual Studio 2013 project files.
                             Optional [arch] can be "Win64" or "ARM".
Visual Studio 11 2012 [arch] = Generates Visual Studio 2012 project files.
                             Optional [arch] can be "Win64" or "ARM".
Visual Studio 10 2010 [arch] = Generates Visual Studio 2010 project files.
                             Optional [arch] can be "Win64" or "IA64".
Visual Studio 9 2008 [arch]  = Generates Visual Studio 2008 project files.
                             Optional [arch] can be "Win64" or "IA64".
Borland Makefiles           = Generates Borland makefiles.
NMake Makefiles             = Generates NMake makefiles.
NMake Makefiles JOM          = Generates JOM makefiles.
Green Hills MULTI           = Generates Green Hills MULTI files
                             (experimental, work-in-progress).
MSYS Makefiles              = Generates MSYS makefiles.
MinGW Makefiles             = Generates a make file for use with
                             mingw32-make.
Unix Makefiles              = Generates standard UNIX makefiles.

```

手动编写CMakeLists.txt文件，示例如下

```

cmake_minimum_required (VERSION 3.8)

set(CMAKE_CXX_STANDARD 11)

find_package(Torch REQUIRED)

add_executable (CMakeProject "CMakeProject.cpp" "CMakeProject.h")

target_link_libraries(CMakeProject ${TORCH_LIBRARIES})

```

使用cmake自带的generator生成vs项目，以vs2015为例

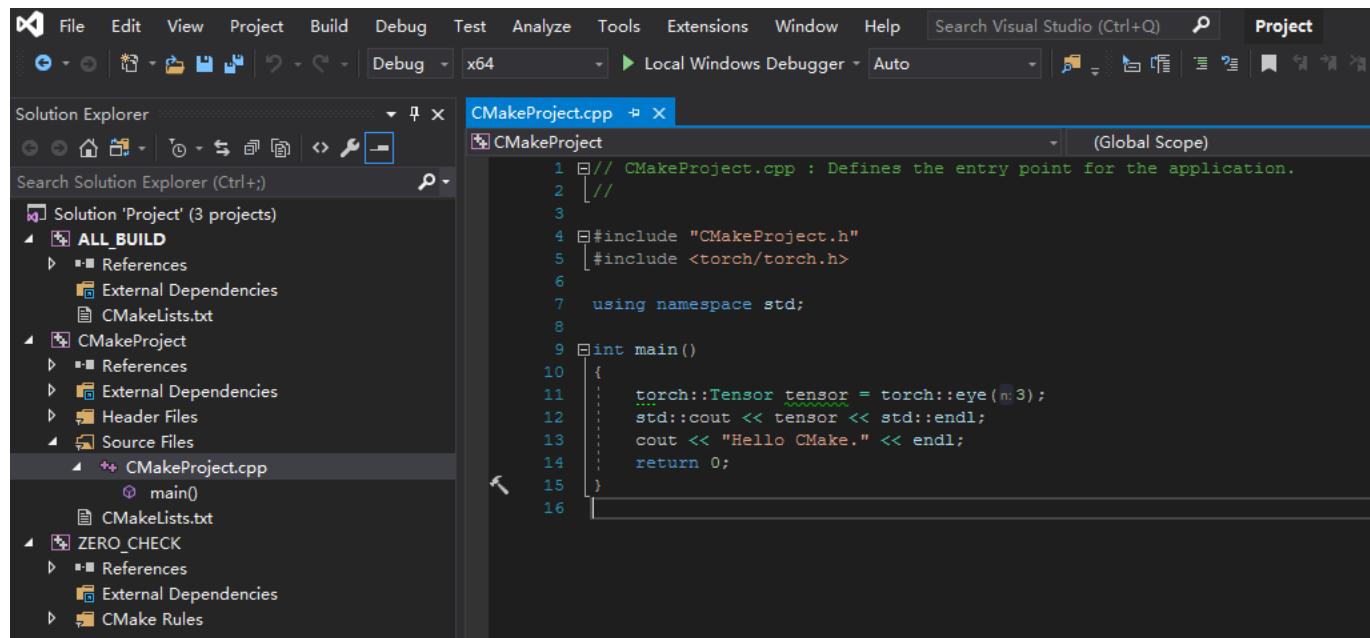
```
cmake -DCMAKE_PREFIX_PATH=C:\Users\chenjz\Downloads\libtorchGPU -  
DCMAKE_BUILD_TYPE=Debug -G"Visual Studio 14 2015 Win 64" .
```

若上述操作无误项目文件夹应该如下图, `Project.sln`为生成项目。

| 名称 | 修改日期 | 类型 | 大小 |
|------------------------------|----------------|----------------------|-------|
| CMakeFiles | 2019/4/5 20:46 | 文件夹 | |
| ALL_BUILD.vcxproj | 2019/4/5 20:46 | VC++ Project | 45 KB |
| ALL_BUILD.vcxproj.filters | 2019/4/5 20:46 | VC++ Project Fil... | 1 KB |
| cmake_install.cmake | 2019/4/5 20:46 | CMAKE 文件 | 2 KB |
| CMakeCache.txt | 2019/4/5 20:46 | 文本文档 | 15 KB |
| CMakeLists.txt | 2019/4/4 20:59 | 文本文档 | 1 KB |
| CMakeProject.cpp | 2019/4/5 9:28 | C++ 源文件 | 1 KB |
| CMakeProject.h | 2019/4/4 19:42 | C Header 源文件 | 1 KB |
| CMakeProject.vcxproj | 2019/4/5 20:46 | VC++ Project | 57 KB |
| CMakeProject.vcxproj.filters | 2019/4/5 20:46 | VC++ Project Fil... | 1 KB |
| Project.sln | 2019/4/5 20:46 | Visual Studio Sol... | 4 KB |
| ZERO_CHECK.vcxproj | 2019/4/5 20:46 | VC++ Project | 44 KB |
| ZERO_CHECK.vcxproj.filters | 2019/4/5 20:46 | VC++ Project Fil... | 1 KB |

项目说明

`Project.sln`共包含三个项目: ALL_BUILD, CMakeProject, ZERO_CHECK, 其中CMakeProject是我们想要的。



CMakeProject下的External Dependencies里包含了所有与libtorch有关的头文件, 配置好dll路径后若无错误就可以直接编译运行了。

在HM中调用代码

我在github上提供了一个HM16.20的cmake项目，大家可以在对应位置修改CMakeLists.txt来在指定项目调用libtorch。

- 在C++中加载pytorch模型教程https://pytorch.org/tutorials/advanced/cpp_export.html
- HM cmake工程https://github.com/chjz1024/HM_vc2015