参考资料

- Pytorch C++ API文档
- c++调用torch模型方法

Step by step

- Pytorch模型

```
import torch
import torch.nn.functional as F

# define network
class MyModule(torch.nn.Module):
    def __init__(self, N, M):
        super(MyModule, self).__init__()
        self.fc1 = torch.nn.Linear(N, M)
        self.fc1.weight.data.normal_(0, 0.1)  # initialization

    def forward(self, input):
        output = self.fc1(input)

        return F.relu(output)
```

- 将模型转换成Torch Script并保存在本地

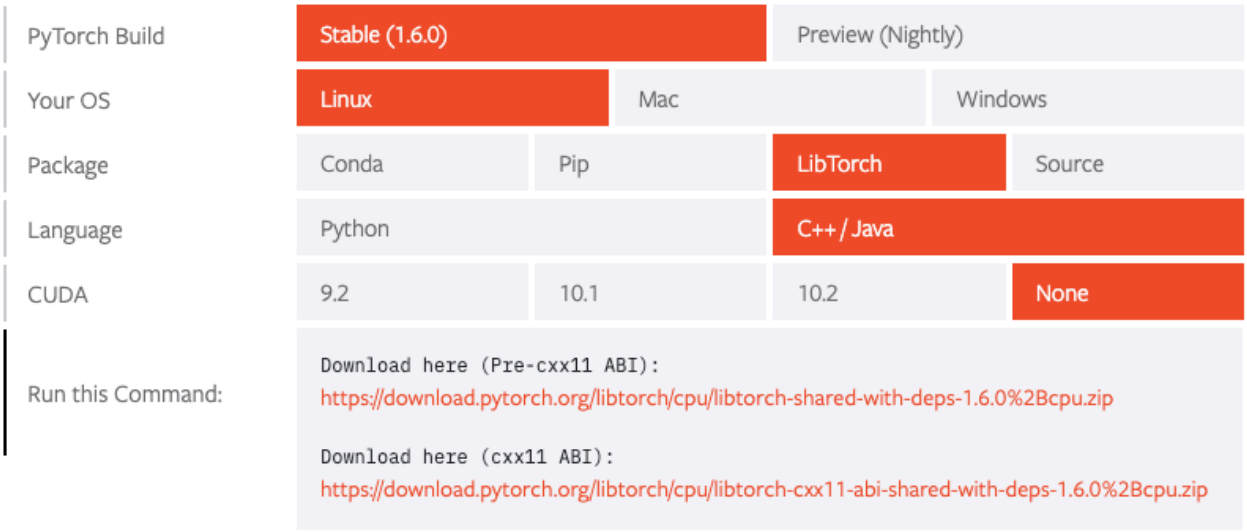
```
# instance
my_module = MyModule(10,20)

# save model
sm = torch.jit.script(my_module)
sm.save("my_module_model.pt")
```

- libtorch依赖库下载

官网下载链接，在下图配置基础上，根据自己需要选择系统类型。

下载完成后，建议解压到上述py文件所在路径。



- C++代码调用案例（example-app.cpp）

```
#include <torch/script.h> // One-stop header.

#include <iostream>
#include <memory>

int main(int argc, const char* argv[]) {
  if (argc != 2) {
    std::cerr << "usage: example-app <path-to-exported-script-module>\n";
    return -1;
  }

  torch::jit::script::Module module;
  try {
    // Deserialize the ScriptModule from a file using torch::jit::load().
    module = torch::jit::load(argv[1]);

    // Create a vector of inputs.
    std::vector<torch::jit::IValue> inputs;
    inputs.push_back(torch::ones({5, 10}));

    // Execute the model and turn its output into a tensor.
    at::Tensor output = module.forward(inputs).toTensor();
    std::cout << output << '\n';
  }
  catch (const c10::Error& e) {
    std::cerr << "error loading the model\n";
    return -1;
  }

  std::cout << "ok\n";
}
```

- CMakeList.txt文件编写

  注意需要将文件中{libtorch 所在路径}替换成上面下载解压后的的libtorch库所在路径（建议放到同目录）

```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
project(custom_ops)

set(Torch_DIR {libtorch 所在路径}/libtorch/share/cmake/Torch)
find_package(Torch REQUIRED)

add_executable(example-app example-app.cpp)
target_link_libraries(example-app "${TORCH_LIBRARIES}")
set_property(TARGET example-app PROPERTY CXX_STANDARD 14)
```

- 编译、链接打包程序

  选手可以通过“apt install cmake”命令安装cmake，后期我们会在下一个版本的docker镜像同步安装。

```
mkdir build
cd build
cmake -DCMAKE_PREFIX_PATH={libtorch 所在路径}/libtorch ..
cmake --build . --config Release
make
```

完成后在build路径下可以得到C++代码的可执行文件example-app。

- 运行代码

  控制台最后一行显示“ok”即可认为运行成功。

```
./example-app ../my_module_model.pt
```

- 生成决赛所需动态链接库

  决赛系统测评系统会在按照上述教程，在demo目录下放置libtorch文件夹，选手可直接在本地生成so动态链接库并上传即可，但需要对CMakeList.txt进行调整，编译、链接后，可在当前目录（build）看到.so文件动态链接库。

```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
project(custom_ops)

set(Torch_DIR {libtorch 所在路径}/libtorch/share/cmake/Torch)
find_package(Torch REQUIRED)

# 更改部分
SET(LIBRARY_OUTPUT_PATH "./" )
ADD_LIBRARY(solution SHARED solution.cxx ${Torch_DIR}/../../../lib
${CURRENT_INCLUDE})
```

```
#add_executable(example-app example-app.cpp)

target_link_libraries(example-app "${TORCH_LIBRARIES}")
set_property(TARGET example-app PROPERTY CXX_STANDARD 14)
```