

引言

本文旨在介绍如何在Windows平台使用pytorch的c++ api部署pytorch的CNN模型，本文的部署的模型只有推理功能，这是由于torch::jit不支持部分层或者操作的反向传播。当然即使只是推理也足够许多项目运行了，部署使用的工具有[visual studio](#)，[opencv](#)，[libtorch](#)。

环境

本文环境如下：

win10平台

cuda10.2+cudnn7.6.5

双显卡Gtx 1080Ti

visual studio 2017 community version

opencv 4.5.0

libtorch 1.7

事实上，除了libtorch是版本有强制要求不低于pytorch版本外（主要是可能存在的api问题，否则也不必）和visual studio的版本要求外，其他如opencv并无版本要求，甚至如果只部署cpu的话，显卡都不是必须。

visual studio

visual studio版本最好在2015及以上，本文用2017版本。下载链接在[链接1](#)，具体安装过程可以参考[链接2](#)。打开链接1下载社区版本即可，安装时对于c++程序设计只需安装对应部分，勾选如下：



visual studio的安装并无太多需要赘述，按照教程操作就好。

opencv

截止成文时，opencv版本已到4.5.0。去[官网](#)下载你想要的版本即可，当然限定平台为Windows平台。opencv的版本如若读者没有特别需要均无影响，本文也试过不同版本，不影响部署。opencv安装也容易，虽然是exe文件，但是实际就是个压缩包，解压到你想要的目录最好。值得注意的是，好的编程习惯很重要，所有程序涉及路径以英文路径为佳，避免出错。

libtorch

libtorch使用所需要的环境和训练最好保持一致，其中cuda，显卡驱动以及libtorch版本配置一般不应低于训练环境。尤其是libtorch版本要求更为严格，否则部分pytorch的api无法在libtorch中使用。

本文中以libtorch1.7为例介绍，读者最新版1.7.1使用，亲测可用。下载时以release版本为佳，避免一些不必要的错误。

PyTorch Build	Stable (1.7.1)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
CUDA	9.2	10.1	10.2	11.0 None
Run this Command:	<p>Windows binaries do not support Java. Support is only available for Linux and MacOS. Download here for C++ (Release version):</p> <p>https://download.pytorch.org/libtorch/cu102/libtorch-win-shared-with-deps-1.7.1.zip</p> <p>Download here for C++ (Debug version):</p> <p>https://download.pytorch.org/libtorch/cu102/libtorch-win-shared-with-deps-debug-1.7.1.zip</p>			

下载后同样解压到读者想要路径，可以更改好解压名称如下以方便版本管理。

此电脑 > 新加卷 (D:) > AllentFiles > code > dependency

名称	修改日期	类型
libtorch17debug	2020/11/13 16:26	文件夹
libtorch17release	2020/11/13 17:16	文件夹
opencv-4.5.0-vc14_vc15	2020/11/13 16:41	文件夹

这样下来就已经准备好部署所需要的依赖项。

例子

生成.pt文件

接下来以ResNet34分类模型为例尝试部署分类模型。准备一张图片用以判断是否部署成功，本文用例句如下：



接下来先和官网类似生成torchscript模型，亦即本文中的pt文件。本文使用代码如下：

```

from torchvision.models import resnet34
import torch.nn.functional as F
import torch.nn as nn
import torch
import cv2

#读取一张图片，并转换成[1,3,224,224]的float张量并归一化
image = cv2.imread("flower.jpg")
image = cv2.resize(image,(224,224))
input_tensor = torch.tensor(image).permute(2,0,1).unsqueeze(0).float()/225.0

#定义并加载resnet34模型在imagenet预训练的权重
model = resnet34(pretrained=True)
model.eval()
#查看模型预测该付图的结果
output = model(input_tensor)
output = F.softmax(output,1)
print("模型预测结果为第{}类，置信度为{}".format(torch.argmax(output),output.max()))

#生成pt模型，按照官网来即可
model=model.to(torch.device("cpu"))
model.eval()
var=torch.ones((1,3,224,224))
traced_script_module = torch.jit.trace(model, var)
traced_script_module.save("resnet34.pt")

```

输出结果为：

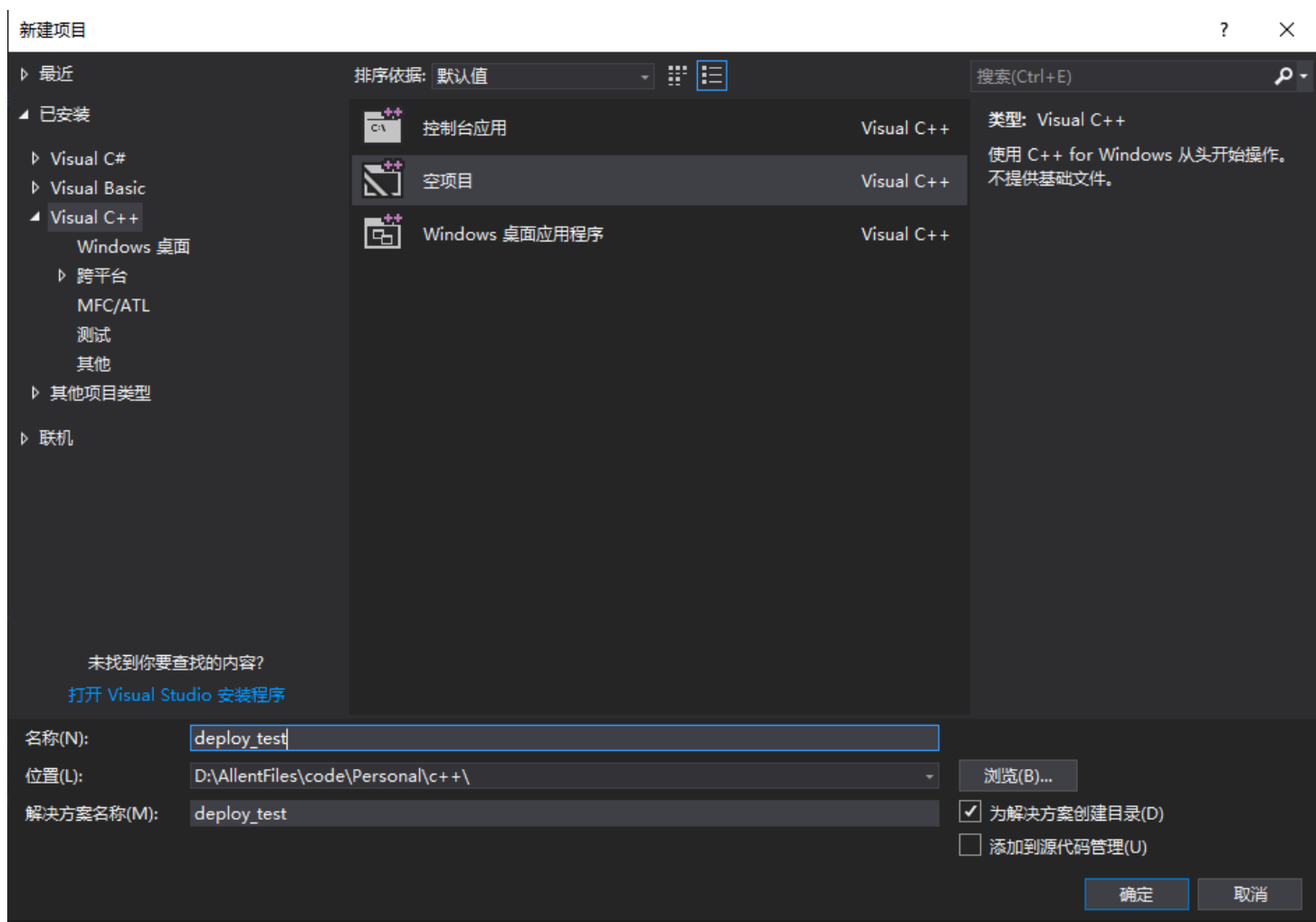
```
模型预测结果为第723类，置信度为0.5916505455970764
```

代码运行结束即可生成.pt文件。

Visual Studio项目配置

新建Visual Studio工程项目。

打开Visual Studio 2017，点击文件->新建->项目，新建空白的c++项目如下：



本文新建项目名称为`deploy_test`，新建空白项目后右键源文件，点击添加新建项，生成`main.cpp`。至此，Visual Studio项目的准备工作已做好，接下来时配置项目环境。

编译环境配置

在项目的管理器中设置项目的编译为Release，平台选择x64。如图：



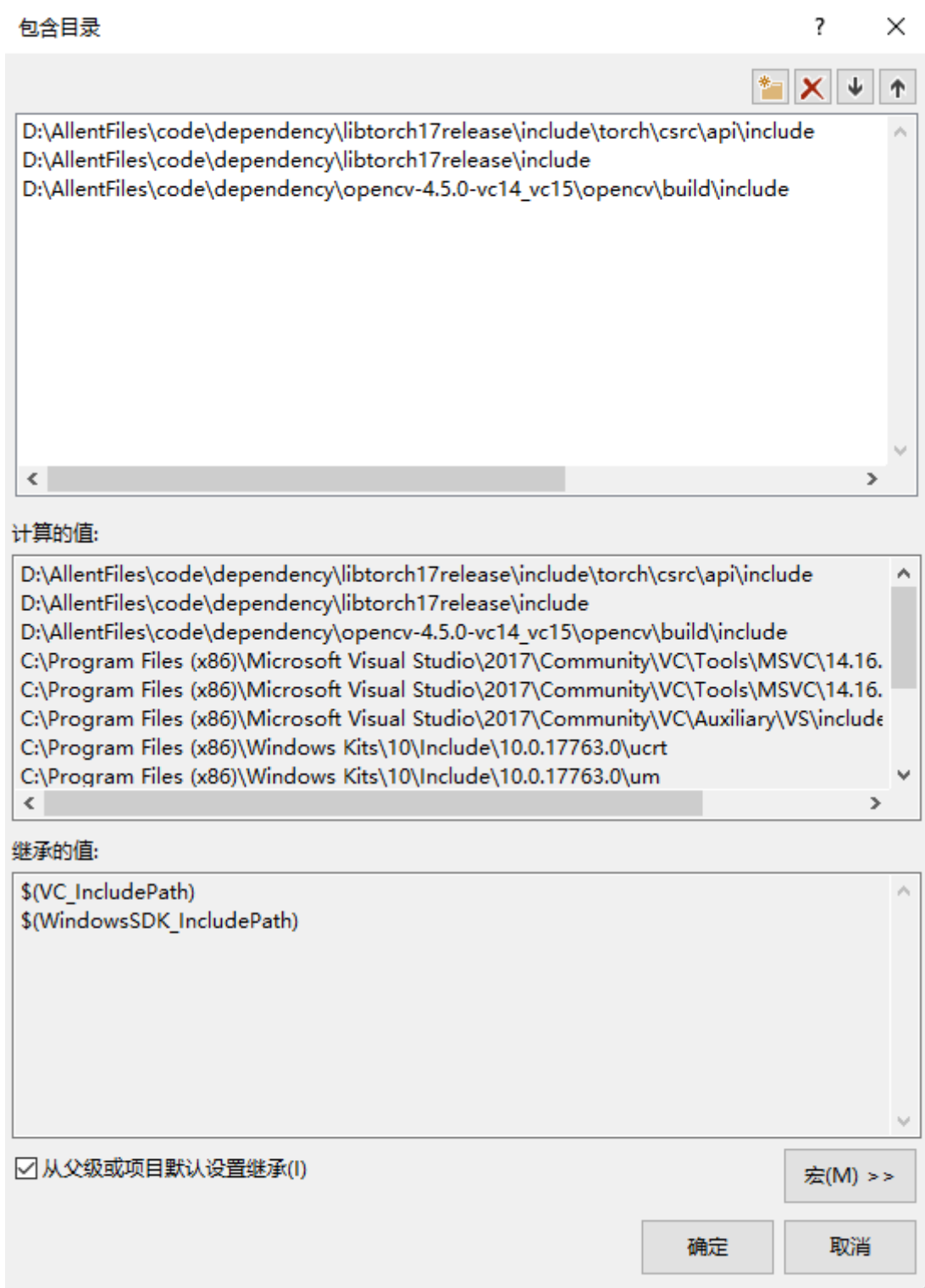
配置项目属性

include

右键项目`deploy_test`，选择属性进入属性页的配置属性。选择VC++目录，需要添加包含目录和库目录。包含目录配置路径为

your path to libtorch\include\torch\csrc\api\include
your path to libtorch\include
your path to opencv\build\include

本文的配置结果如下：

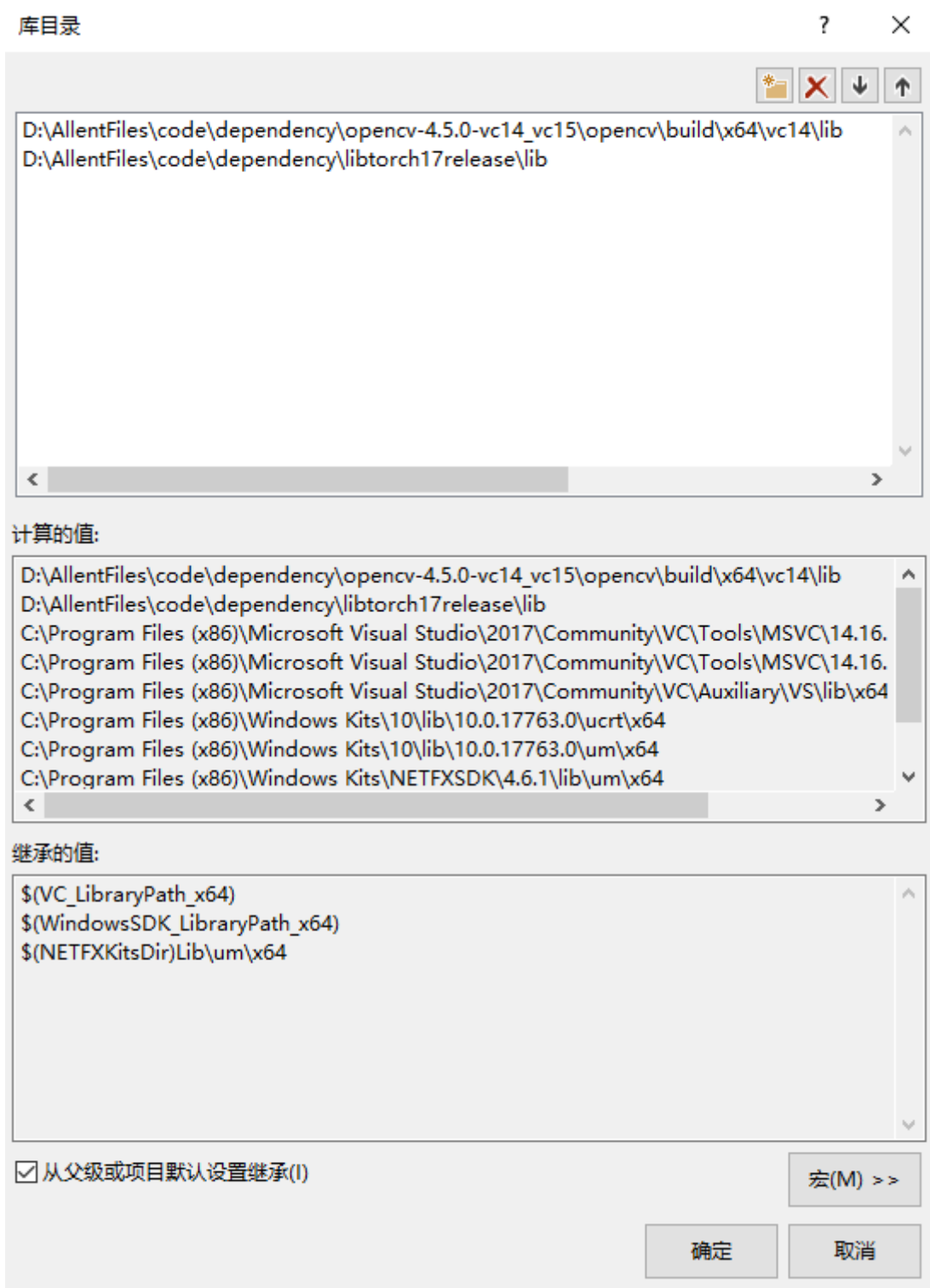


lib

库目录的配置路径为：

your path to libtorch\lib
your path to opencv\build\x64\vc14\lib

其中，vc版本与vs版本对应关系以及opencv与vc版本间的关系见[链接](#)。VS2017对应版本为vc15，opencv的\build\x64文件夹下同时有vc14和vc15，本文选择了vc14也是可用的（VS高版本向下兼容）。库目录具体配置如下图：



link

最后添加链接器，点击链接器->输入->附加依赖项，编辑添加库目录中libtorch库目录下所有的.lib文件名。此外，为使用opencv还需添加opencv的.lib文件。本文添加如下：

```
opencv_world450.lib
asmjit.lib
c10.lib
c10d.lib
c10_cuda.lib
caffe2_detectron_ops_gpu.lib
caffe2_module_test_dynamic.lib
caffe2_nvrtc.lib
clog.lib
cpuinfo.lib
dnnl.lib
fbgemm.lib
gloo.lib
gloo_cuda.lib
libprotobuf-lite.lib
libprotobuf.lib
libprotoc.lib
mkldnn.lib
torch.lib
torch_cpu.lib
torch_cuda.lib
```

dll

动态链接库需要放入指定位置，有三种做法：

- 将opencv中的\build\x64\vc14\bin文件夹路径放入环境变量path中，libtorch中的lib文件夹同样如此；
- 拷贝opencv和libtorch中的dll文件到项目的执行目录中；
- 将opencv和libtorch中的dll路径配置到VS项目中；

从省事角度和工程需要角度，一般都是直接拷贝dll到执行目录中。读者可以在项目编译后执行时报错缺啥拷贝啥。

至此，VS项目的配置内容已经完成，下面开始c++代码。

cpp代码

```

#include<opencv2/opencv.hpp>
#include <torch/torch.h>
#include <torch/script.h>

int main()
{
    //定义使用cuda
    auto device = torch::Device(torch::kCUDA,0);
    //读取图片
    auto image = cv::imread("your path to\\flower.jpg");
    //缩放至指定大小
    cv::resize(image, image, cv::Size(224, 224));
    //转成张量
    auto input_tensor = torch::from_blob(image.data, { image.rows, image.cols, 3 }, torch::k
    //加载模型
    auto model = torch::jit::load("your path to\\resnet34.pt");
    model.to(device);
    model.eval();
    //前向传播
    auto output = model.forward({input_tensor.to(device)}).toTensor();
    output = torch::softmax(output, 1);
    std::cout << "模型预测结果为第" << torch::argmax(output) << "类，置信度为" << output.max()
    return 0;
}

```

编译执行，代码的输出结果为：

```

模型预测结果为第723
[ CUDALongType{} ]类，置信度为0.591652
[ CUDAFloatType{} ]

```

可以发现，torchscript文件推理的结果还是和python的略有不同，不过也已经时小数点后第6位了，一般不会影响最后结果判定。

一些报错

错误1：无法使用GPU

目前最新的libtorch依据是1.7+cuda10.2，我也有使用，但是目前发布的版本编译的并不完美。如果官方仍然没有更新的话，以该版本运行的程序可以在CPU中正常使用，但是将模型移至GPU时会出错。相比于正常添加lib文件名，1.7版本的需要在链接器里多添加一句：

```
/INCLUDE:?warp_size@cuda@at@@YAHXZ
```

其他版本如有遇到类似问题可以同样方式解决，该错误似乎是windows平台导致，此解决方法参考了[此链接](#)。

错误2：编译时报错“std”：不明确的符号

调整属性页->属性配置->c/c++->语言->符合模式设置为否即可编译成功。

错误3：缺少dll

如果编译成功，执行报错由于找不到xxxx.dll，无法继续执行代码。则解决方式有三：

- 添加对应dll的目录到系统环境变量的path中
- 在VS项目中配置dll路径
- 直接将该dll复制粘贴到项目的执行路径下。

如果项目不多，建议直接复制粘贴。