**2023年2月8日星期三**

目前简单了解了MMDetection的使用方式和模型移动部署框架

模型移动部署框架主要有三种：PaddlePaddle、NCNN和TensorFlow。不知道我们的模型会属于哪种模式，但不必急于弄清，因为后面可能还会有变化。目前最好多学几种部署方案，触类旁通。

框架不会影响运行速度，我的理解是来说NCNN比较通用，其他两种在自己的体系里使用起来格外方便。

首先尝试的是NCNN，要用到Android Studio，以自己的mi 6手机作为移动端，尝试进行简单的开发，然后到部署模型。

**2023年2月9日星期四**

昨天用Android Studio把一个Hello World程序部署到了手机上，今天成功把NCNN的一个YOLOv5官方demo部署到了手机上，并成功运行。

**报错记录：**

**Android Studio no devices**

**Installation did not succeed. The application could not be installed: INSTALL\_FAILED\_USER\_RESTRICTED**

**原因分析：权限不足，Android Studio/计算机无法访问移动设备/手机或无法向手机发送APP。**

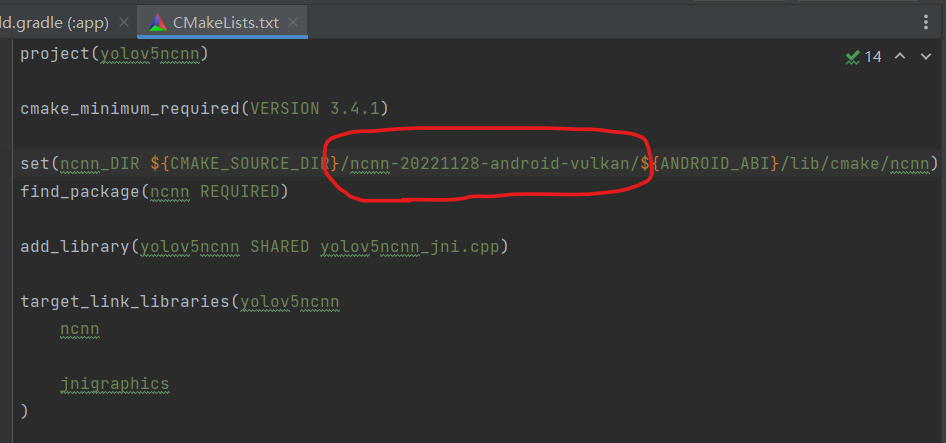
**解决方案：手机要打开开发者模式，打开“USB调试”及其下的“USB安装”。如果仍然报如上错误，可断开连接，确认开启上述按钮后重新连接。**

**By not providing "Findxxx-flight-algorithms.cmake" in CMAKE\_MODULE\_PATH this project has asked CMake to find a package configuration file provided by "xxx-flight-algorithms", but CMake did not find one.**

**原因分析：没有正确添加CmakeLists.txt要求的配置文件。**

**解决方案：将官网配套的 “ncnn-20221128-android-vulkan.zip”解压缩后整个文件夹放入 “D:\AndroidTool\AndroidProject\ncnn-android-yolov5-master\app\src\main\jni”路径下。**

**且CmakeLists.txt中的路径要更新为与实际文件夹名称一致。**

****

**No toolchains found in the NDK toolchains folder for ABI with prefix: arm-linux-androideabi**

**原因分析：最新版ndk（version=25.x.xxxxxxx）的toolchains文件夹中无arm-linux-androideabi文件**

**解决方案：两种方法**

1. **同时安装低版本的ndk（如version=21.x.xxxxxxx，约1GB），将低版本ndk中toolchains 文件夹下的arm-linux-androideabi等文件复制到25.x.xxxxxxx版本ndk的toolchains 文件夹中.**
2. **安装低版本的ndk，然后在build.gradle的android代码块内添加一句：“ndkVersion '22.1.7171670'”即可。**

一、CPU的浮点计算性能公式

我们常用双精度浮点运算

能力衡量一个处理器的科学计算

的能力，就是处理64bit小数点浮动数据的能力。

intel的最新cpu支持高级矢量指令集

AVX2、AVX512， 其中AVX2的处理器的单指令的长度是256bit，每颗intelCPU包含2个FMA，一个FMA一个时钟周期可以进行2次乘或者加的运算，那么这个处理器在1个核心1个时钟周期可以执行256bit\*2FMA\*2M/A/64=16次浮点运算，也称为16FLOPs，就是Floating Point Operations Per Second；

支持AVX512的处理器的单指令的长度是512Bit，每个intel核心假设包含2个FMA，一个FMA一个时钟周期

可以进行2次乘或者加的运算，那么这个处理器在1个核心1个时钟周期可以执行512bit\*2FMA\*2M/A/64=32次浮点运算，也称为32FLOPs。

就是说理论上后者的运算能力其实是前者的一倍，但是实际中不可能达到，因为进行更长的指令运算，流水线之间更加密集，但核心频率会降低；导致整个处理器的能力降低；

一个处理器的计算能力和核心的个数，核心的频率，核心单时钟周期的能力三个因素有关系。

GPU的浮点性能计算公式

GPU能做的CPU都能做，CPU能做的GPU却不一定能够做到，GPU一般一个时钟周期可以操作64bit的数据，1个核心实现1个FMA。

这个GPU的计算能力的单元是：64bit\*1FMA\*2M/A/64bit=2FLOPs/Cycle。

GPU的计算能力也是一样和核心个数，核心频率，核心单时钟周期能力三个因素有关，但是架不住GPU的核心的数量多

例如：对现在nvidia 的pascal架构超算卡--- Tesla P100，是1792核@1.328GHz,其理论的双精度浮点性能是：1792Core\*1.328GHZ\*2FLOPs/Cycle=4759.552GFLOPs=4.7TFLOPs。

例如：对现在nvidia 的Volta架构的超算卡---Tesla V100，是2560核@1.245GHz，其理论的双精度浮点性能是：2560Core\*1.245GHZ\*2FLOPs/Cycle=6374.4GFLOPs=6.3TFLOPs

现在ML繁荣的时代，对64bit长度的浮点运算需求不是那么的大，反而是32bit或者16bit、8bit INT、4bit INT的运算需求比较大。

因此nvidia 最新的tesla一直在强调单精度

甚至半精度，turing就是这样的。

intel为了加速这些计算，也在其处理器中实现了一些加速低精度运算的指令。

**2023年2月11日星期六**

研究两个问题：1.计算算法所需的计算量和硬件可支持的运算量；2.Android Studio的项目框架结构及如何修改。

美格规格书中写道：

内置Adreno™ GPU 635，支持OpenGL ES 3.2，Vulkan1.x，支持OpenCL 2.0。模组内置AI 处理器Dual HVX 和4K HMX，AI 算力超过10 Tops。

百度百科给出的Adreno GPU规格如下：



*Vulkan被称为下一代OpenGL，之前实验中的YOLOv5NCNN使用的就是Vulkan。*

总结：64GB存储 + 4GB低功耗内存 + 10 TFLOPs运算速度。*和前面的红字比较，应该是比较够用的。*

*TOPS是Tera Operations Per Second的缩写，1TOPS代表处理器每秒钟可进行一万亿次（10^12）操作*

**2023年2月22日星期三**

近期部署实战经验及遇到的问题

*同济子豪兄课程文件*

*https://github.com/open-mmlab/OpenMMLabCourse*

输入模型：

模型形式：.pth等

代码形式：.py，.java等

输出模型：

编程调用模型：onnx，tnn等

可调用模块（提供给其他课题组）：

app形式（调试用）

目前跑通的示例：

D:\AndroidTool\AndroidProject\ncnn-android-yolov5-master

java搭建网络，最终输出app

D:\PythonCodes\MMDeployDemo

python搭建网络，输出中间模型，最终输出模型

~~D:\PythonCodes\mmdeploy~~

linux可能不是必须的

部分库的安装还有问题

流程仍不清晰，关键是要能自己生成模型部署时必要的配置文件，以及掌握在移动端调用模型的方法

~~D:\AndroidTool\AndroidProject\ncnn-android-scrfd-master~~

和yolov5的示例差不多

D:\AndroidTool\AndroidProject\tnn-master

提供了丰富的文档和转化好的tnnmodel格式模型

文档更适合有一定经验的开发者

有的机型对NPU的支持存在问题

可自定义的部分较多

尚不能自己生成模型部署时必要的配置文件，以及在移动端调用模型

tnn的部署方案也许简单可行

**The IDE is running low on memory and this might affect performance. Please consider increasing available heap.**

**将 "IDE最大堆大小 "从1280 MB（默认）设置为2048 MB后，性能得到了改善。**

**文件 --> 设置 --> 外观和行为 --> 系统设置 --> 内存设置**

**Expected NDK STL shared object file at D:\AndroidTool\AndroidSDK\ndk\25.2.9519653\sources\cxx-stl\llvm-libc++\libs\arm64-v8a\libc++\_shared.so**

**D:\AndroidTool\AndroidSDK\ndk\25.2.9519653\sources\cxx-stl\llvm-libc++下的libs文件夹缺失。**

**官网下载页面被阻止**

**官网下载的ndk中也没有libs文件夹**

**使用不受支持的旧版本（r24中的libs文件夹）**

**No toolchains found in the NDK toolchains folder for ABI with prefix: arm-linux-androideabi**

**ndkVersion "22.1.7171670"**

//编译9min48s时，10000+warning

//BUILD SUCCESSFUL in 9m 47s，但是Build Output还在跑，30min时25k warning。

部署成功，图片输入暂时只能使用默认图片，视频输入调用本地摄像头，识别能力较弱。

**Ubuntu中**

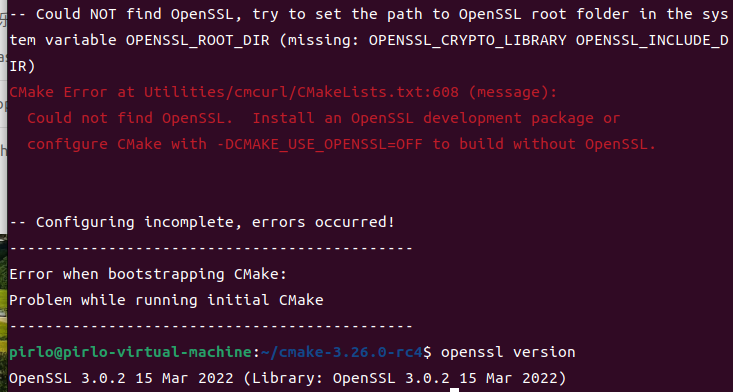
安装cmake <https://blog.csdn.net/qq_38964360/article/details/126119740>

**Cannot find appropriate Makefile processor on this system.**

**apt-get install build-essential**

缺少gcc、g++的解决方案类似

**Could NOT find OpenSSL, try to set the path to OpenSSL root folder in the system variable OPENSSL\_ROOT\_DIR (missing: OPENSSL\_CRYPTO\_LIBRARY OPENSSL\_INCLUDE\_DIR)**





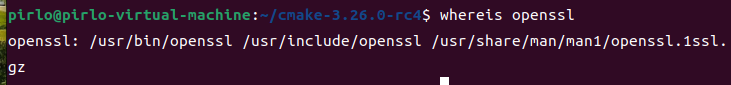
 添加环境变量无效

export OPENSSL\_ROOT\_DIR=/usr/bin/openssl

export OPENSSL\_LIBRARIES=/usr/share/man/man1/openssl.1ssl.gz

**重新安装OpenSSL：sudo apt-get update**

**sudo apt-get install libssl-dev**



再次尝试，成功：CMake has bootstrapped.

**虚拟机连不上网（突发）**

<https://blog.csdn.net/m0_46100784/article/details/126833558> 方法五DHCP

sudo make install

**虚拟机共享文件夹失效（突发）**

虚拟机-重新安装Vmware Tools，若该选项变灰，虚拟机-设置，将CD和软盘设为使用物理驱动器。

运行.pl文件时，全部选择yes

没有解决

**ERROR: Could not install packages due to an OSError…**

**install 后面加上 –user**

**vmware tool安装时出现段错误segment fault**

版本不匹配，可以尝试使用Vmware自带的linux.iso，如果依旧报错，通过apt-get方式安装。

2023年3月3日星期五

在ubuntu中安装了anaconda和pycharm

按照官网的docs配置环境

安装ncnn时，需要安装Vulkan utils，却怎么也找不到。换清华源无效。

……

跳过这一步，git MMDeploy进行模型转换。

2023年3月5日星期日

MMDeploy的部署还是需要ncnn啊……

其中被卡住的一步是：

cmake -DCMAKE\_TOOLCHAIN\_FILE=${NDK\_PATH}/build/cmake/android.toolchain.cmake -DANDROID\_ABI="${ANDROID\_ABI}" -DANDROID\_PLATFORM=android-30 -DNCNN\_VULKAN=ON -DNCNN\_DISABLE\_EXCEPTION=OFF -DNCNN\_DISABLE\_RTTI=OFF ..

**把${}中的内容替换成自己的路径就可以通过。**

**比如:**

**cmake -DCMAKE\_TOOLCHAIN\_FILE=/opt/android-ndk-r23c/build/cmake/android.toolchain.cmake -DANDROID\_ABI="arm64-v8a" -DANDROID\_PLATFORM=android-30 -DNCNN\_VULKAN=ON -DNCNN\_DISABLE\_EXCEPTION=OFF -DNCNN\_DISABLE\_RTTI=OFF ..**

奇怪的是，之前我用echo检查环境变量时发现设置的没错，但是无法成功cmake。

Android.md cmake时如果报错，找不到toolchain、ncnn、opencv之类的，可以把${}格式的部分替换成自己的路径，我编辑为android.txt放在android.md同一目录下：

|  |
| --- |
| cmake .. \  -DMMDEPLOY\_BUILD\_SDK=ON \  -DMMDEPLOY\_BUILD\_EXAMPLES=ON \  -DMMDEPLOY\_BUILD\_SDK\_JAVA\_API=ON \  -DOpenCV\_DIR=/home/pirlo/OpenCV-android-sdk/sdk/native/jni/abi-arm64-v8a \  -Dncnn\_DIR=/home/pirlo/ncnn/build\_arm64-v8a/install/lib/cmake/ncnn \  -DMMDEPLOY\_TARGET\_BACKENDS=ncnn \  -DMMDEPLOY\_SHARED\_LIBS=OFF \  -DCMAKE\_TOOLCHAIN\_FILE=/opt/android-ndk-r23c/build/cmake/android.toolchain.cmake \  -DANDROID\_ABI=${ANDROID\_ABI} \  -DANDROID\_PLATFORM=android-30 \  -DANDROID\_CPP\_FEATURES="rtti exceptions" |

然后还可能报两个错误，一个跟spdlog有关，它在工程中是一个特殊的文件夹（分支？），因此将工程以压缩包形式下载时，得到的spdlog文件夹是空的，需要自己单独下载。

还有一个是缺少JAVA，之前装环境时没注意到，以为非必选。过程比较繁琐，第三步可能会报错。

按照文档安装JAVA并不麻烦，做前两大步就行。

这样android.md的内容就都跑通了。

2023年3月8日星期三

现在还有两个主要问题，一个是cmake ncnn sdk时，找不到ncnnConfig.cmake所在的路径。（已经设置了绝对路径并检查）

把模型文件传给虚拟机。

重启后能找到ncnn了，其实和Android.md中设置的NCNN\_DIR是一样的。但又出现找不到vulkan的提示。Linux-x86\_64.md中有提到安装Vulkan吗？我没安装过吗？之前也报这个错误吗？

**Vulkan的安装在ncnn安装的wiki里，因为我提前安装了ncnn，因此没有这个步骤。由于没有cuda，Pytorch那一步也跳过了。**

重试Android.md发现最后一步install出现incompatible linux eabi错误，之前有吗？

安装Vulkan报错，没有数字签名、由于没有公钥,无法验证下列签名、无法安全地用该源进行更新，所以默认禁用该源。换源换出问题了？把忘记什么时候添加的sources文件最下面两行注释掉就解决问题了。

<https://blog.csdn.net/luolinll1212/article/details/113261022>

无法定位软件包 vulkan-utils

<https://blog.csdn.net/weixin_62988760/article/details/125697967>

下载后Build，非常卡，多次死机，pycharm闪退…………

至于模型转换的问题，第一项参数config文件其实在项目里也已经给出，需要自己选择。由于磁盘空间不足，我删除了onnxruntime和tnn-master，这里应该只能选择ncnn系列的config文件了。执行deploy.python报错：

Only py/yml/yaml/json type are supported now!

据说是因为未添加config文件参数。百度到的结果很少。我尝试了detection/classification\_ncnn/onnxruntime\_dynamic/static.py，均是这个结果。

2023年3月9日星期四

三个问题一个也没解决，QAQ

1. 模型转换：依然卡在config那一步，我试着自己编写了文件，但它的问题应该是找不到文件。
2. Android.md最后的install：is incompatible with armelf\_linux\_eabi

网上的解决方法不多，而且明显不适合。最终发现cmake时替换${}不彻底，有一项不影响cmake但会导致install失败。

磁盘又爆满了。

1. Linux-x86\_64.md还是找不到ncnn/onnx的converter，我试着退回ncnn的编译，发现也是找不到。不过这个文件可能也没有必要跑通。

2023年3月10日星期五

实验表明，程序不是找不到配置文件，而是认为配置文件不是合法文件。



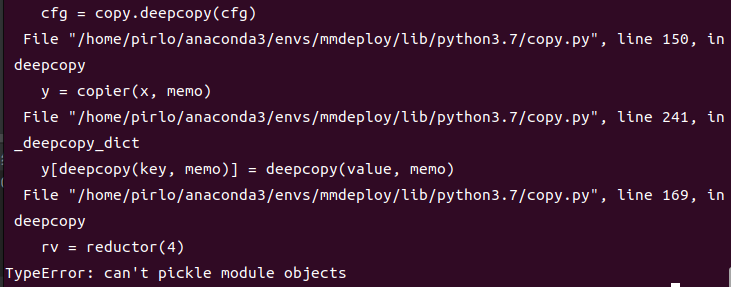
问题似乎浮出了水面，其实错的不是deploy\_config\_path，而是model\_config\_path。我清晰地记得一开始我是写对的，然后很快鬼使神差地把.py和.pth的路径交换了一下。当然换回来之后还是跑不通。



于是我新建了文件夹，把学长的mmdetection下的coco\_detection.py粘贴过去。



……



据说是变量格式问题，但这种情况也不好去改代码。当然，这些错误traceback的第一项都是：



估计两个配置文件总有一个出问题。我看的视频里deploy\_cfg不修改直接用，于是我把我补充的代码都注释掉，还换其他的配置文件测试，结果还是一样的。



用教程示例试了一下也不成功，需要装mmdet，装完后报错和我们的r18错误不同。看来还是得在Linux下编译一个onnxruntime来测试。

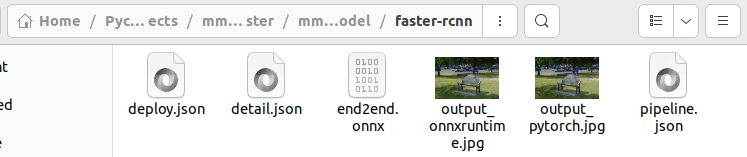
虚拟机扩容到75G。

2023年3月11日星期六

重新安装了onnxruntime，不挂梯子下载不下来。Linux-x86\_64.md cpu+onnxruntime编译顺利通过，cmake注意两点。

1. c++版本设置为11.
2. PycharmProjects/mmdeploy-master/third\_party/pybind11中的文件在分支中，需要单独下载。

成功生成了onnxruntime下的faster-rcnn的onnx模型。参数写在Android.txt末尾。



2023年3月13日星期一

Linux-x86\_64.md找不到Protobuf，上次也是这个问题，当时觉得不需要跑通就没有记录。

**Protobuf not found, onnx model convert tool won't be built**

**“protobuf是一种轻便高效的结构化数据存储格式”，需要单独安装，安装后即可通过编译。下载链接：**

<https://github.com/protocolbuffers/protobuf/releases/>

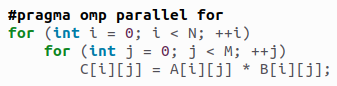
步骤参考：<https://blog.csdn.net/u014689845/article/details/86692079>

到“若protobuf目录下已经有了configure文件，就不用运行./autogen.sh”这一步，我们protobuf 22.2目录下既没有configure文件，也没有autogen.sh。但是我们的问题已经解决了。

**g++: error: unrecognized command-line option ‘-static-openmp’**

这个错误有点麻烦，直接搜出来前面都是c++版本低的，搜后面的关键词出来的是说ndk版本低。但是我们用的是23版本，就算不是最新也很高了。

关于openmp这个东西，似乎是不需要自己安装的。我编写了一段代码来测试它是否存在：



这段代码可以正常编译和运行，因此openmp应该是存在的。尝试更换g++-7，需要换源，报错，未果。

重新尝试了一下模型转换，ncnn依然不成功，报错：

**TypeError: can't pickle module objects**

Onnxruntime成功，虽然有些UserWarning，它们是说mmcv要升级改名的，暂时不用管。

**GPG 错误：http://mirrors.aliyun.com/ubuntu bionic InRelease: 由于没有公钥，无法验证下列签名： NO\_PUBKEY 3B4FE6ACC0B21F32**

**E: 仓库 “http://mirrors.aliyun.com/ubuntu bionic InRelease” 没有数字签名。**

**N: 无法安全地用该源进行更新，所以默认禁用该源。**

**N: 参见 apt-secure(8) 手册以了解仓库创建和用户配置方面的细节。**

**sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 3B4FE6ACC0B21F32**

**有一些软件包无法被安装。如果您用的是 unstable 发行版，这也许是因为系统无法达到您要求的状态造成的。该版本中可能会有一些您需要的软件包尚未被创建或是它们已被从新到(Incoming)目录移出。下列信息可能会对解决问题有所帮助：**

**下列软件包有未满足的依赖关系： gcc-7 : 依赖: libgcc-7-dev (= 7.5.0-3ubuntu1~18.04) 但无法安装它**

**libstdc++-7-dev : 依赖: libgcc-7-dev (= 7.5.0-3ubuntu1~18.04) 但无法安装它**

**E: 无法修正错误，因为您要求某些软件包保持现状，就是它们破坏了软件包间的依赖关系。**

sudo apt install libgcc-7-dev:i386

这将导致g++ 11的依赖被卸载，而且问题仍未解决，可能需要安装更多包。文档里只是要求g++版本不低于7，因此11应该是可以的，还是别改了。

经过上述尝试后，现各项cmake均不能成功，pytorch无法重新下载，屡次尝试无果，可能要重新装环境了。

2023年3月14日星期二

从3.8导出的虚拟机继续，先跑Android.md，然后Linux。遇到了一些新问题，补充到前文中了。Android.md跑通。

Linux要编译ncnn和onnxruntime，还是需要vulkan，且wiki和网上的方法都不能完全跑通，重启后静置一下午，暂时不再报错。但ncnn在install时出现unrecognized command-line option ‘-static-openmp’错误，感觉这两天白忙活了。

编译算子cmake不能通过，model convert最小化安装时闪退，后面的cmake不通过。

**No such file or directory #include <Python.h>**

这个乍一看很惊人，原因是python-dev包没有安装。

**sudo apt-get install python3-dev**

缺少mmdeploy就pip install之，按照<https://mmdeploy.readthedocs.io/zh_CN/latest/get_started.html>

然后从mmdetection中找到缺少的配置文件在项目中补全就行了，我是在config下新建文件夹把文件直接拷贝过去的，按照报错来就行。这样onnxruntime的流程就还是跑不通。

回到了原来的虚拟电脑，经过简单操作之后大部分步骤都跑通了，问题还是unrecognized command-line option ‘-static-openmp’。

可能出问题的部分有g++、ncnn和mmdeploy。不过CMAKE\_CXX\_COMPILER选g++-11、g++-7还是/usr/bin/g++都跑不通，暂时排除。



整理下目前的进度吧：

|  |  |  |
| --- | --- | --- |
| Android ncnn SDK | 成功生成 | 暂未测试 |
| ONNXRuntime SDK | 成功生成 | 测试成功 |
| ncnn SDK | 生成失败 ‘-static-openmp’ |  |

2023年3月15日星期三

针对目前g++11无法识别openmp编译选项的问题，不同角度进行了搜索，但是都没有符合我们情况的线索。我认为嫌疑最大的还是g++11，因为文档里使用的是g++7嘛，openmp也证明了没有问题。

我切到后来创建的虚拟电脑，安装gcc7，g++7，这次没有自动卸载任何包。按照网站的流程走，直到跑通样例。然后跑咱们的模型。



我把这个文件放在/home/pirlo/PycharmProjects/pythonProject/mmdetection/configs/a\_yu，仍然出现该错误。root/tiny这样的路径也确实没有，原来写在student.py里，那我们修改文件里的路径。

基于g++7和python3.8重构了一个实验环境。中午跟学长充分交流了一下，用mmdetection（dev\_yu）替换了项目中的mmdetection，同步调整了deploy.py参数并改用.sh方式调用。在.sh文件中写好export防止报错。如此跑通了样例和教师模型（onnxruntime）。学生模型报错：

**2023-03-15 15:26:41,298 - mmdeploy - ERROR - `mmdeploy.apis.pytorch2onnx.torch2onnx` with Call id: 0 failed. exit**

下面重新安装ncnn。此时我们的gcc/g++的7/11是共存的，gcc-11-aarch64-linux-gnu也是最新版。于是我安装了gcc-8-aarch64-linux-gnu（因为最低就到8），然后再次cmake。在ncnn/build下仍然失败，ncnn/python下可以成功。接下来的最小化pip也可以进行了。



编译命令是：cmake -DCMAKE\_CXX\_COMPILER=/usr/bin/gcc-11 -DCMAKE\_BUILD\_TYPE=Release -DNCNN\_VULKAN=ON -DNCNN\_BUILD\_EXAMPLES=ON -DNCNN\_PYTHON=ON .. 较安装gcc-8-aarch64-linux-gnu之前没有改变。按照mmdeploy这边的要求加上了最后一项。

接下来ncnn+ndk、cpu+ncnn SDK的cmake，注意编译器要选g++-7，因为一些函数名改变了，新的g++11不认识。Dncnn\_DIR中的ncnn要保持小写。

现在cmake的部分全部完成，接下来就是使用deploy.py进行模型转换。

|  |  |  |  |
| --- | --- | --- | --- |
|  | Faster-cnn样例 | Teacher | Student |
| ONNX | √ | √ | AttributeError |
| ncnn | AssertionError | PermissionError | AttributeError |
| ONNX2ncnn | PermissionError | PermissionError | \ |

ncnn配置文件：mmdeploy/configs/mmdet/detection/single-stage\_ncnn\_dynamic.py

其中Teacher-ncnn-PermissionError发生在deploy.py 232行，转换到后端时。此时onnx文件已经生成。使用onnx2ncnn时，同样出现该错误。

有时出现visualize onnxruntime model failed，就把onnxruntime的环境变量重新设置一下。

Student的转换是重点，目前报错出现在这一句：

torch2ir(ir\_type)(

args.img,

args.work\_dir,

ir\_save\_file,

deploy\_cfg\_path,

model\_cfg\_path,

checkpoint\_path,

device=args.device)

其中各项参数都是我们希望的，但最终报错：**AttributeError: 'NoneType' object has no attribute 'forward'**

**2023-03-15 20:27:30,554 - mmdeploy - ERROR - `mmdeploy.apis.pytorch2onnx.torch2onnx` with Call id: 0 failed. exit.**

2023年3月17日星期五

目前来看，ncnn和student两部分可能都有问题，ncnn我装了两个（手动+脚本），上述错误是脚本安装的ncnn报出的。Traceback往深了找都是mmcv或mmdeploy软件包里的代码，不容易观察，我换用torch2onnx.py进行转换，报错在torch2onnx()函数，与之前一致。最后还是得看源码。

我用输出中间变量的方式寻找了出错的过程，从昨天最后给出的错误来看，是输入的torch nn model没有forward参数。我检查了中间结果并与教师网络转换的中间结果对比，输入的checkpoint路径都是一致且正确的，教师模型在运行到相同语句时顺利通过。那么问题可能出自模型本身，或者在模型的载入过程。

torch\_model = task\_processor.init\_pytorch\_model(model\_checkpoint)

这里路径model\_checkpoint是正确的且与转换教师模型时格式一致，但得到的torch\_model是NoneType的。init\_pytorch\_model()是一个抽象方法，无法通过\_\_subclasses\_\_()来查询它的子类。

仅仅用epoch\_17.pth替换epoch\_12.pth，错误类型不变。

………………………………………………………………………………………………………

前面提到，教师模型基于ncnn转为onnx模型时，会在转换成功后报错Permission denied。该错误发生在backend部分。使用脚本重装ncnn。

这次安装完整，backend仍然报错，AssertionError: onnx2ncnn failed。打印中间结果则无疾而终。

**2023年3月20日星期一**

之前的线索断掉了，但是可以确定读取checkpoint的环节出现了问题，但是读取函数内没有报错，只是返回了一个空值。

1. 修改模型地址为‘1’，mmcv/runner/checkpoint.py内报错FileNotFoundError；
2. 改动教师模型epoch\_17/archive/version中的内容，不影响正常转换；
3. 删除epoch\_17/archive/data中的一个文件，torch/serialization.py报错RuntimeError，找不到文件。
4. 使用pickle查看epoch\_12/archive/data.pkl，报错：

**\_pickle.UnpicklingError: A load persistent id instruction was encountered,but no persistent\_load function was specified.**

网上说是保存和加载模型时的pytorch版本不一致导致的，不过另一种说法可能更适用于我们的情况：

*“直接解释是因为生成pickle文件的过程中使用了persistent\_load，但是读取过程中没有提供。本例中的具体原因是，读取的pickle文件并非pickle格式，只是用pickle这个词表示这是一个规范压缩的数据文件，改用合适的read\_\*()函数就可以了。”*

在检查pickle模块可以正常使用后，我换用mmcv.load读取模型：

**raise TypeError(f'Unsupported format: {file\_format}')**

**TypeError: Unsupported format: None**

这样就跟我们前面的问题对应上了。网上的解释是输出模型的格式不对，查代码发现mmcv的io.py认为该文件不是合法的句柄（不是pkl）。

删除epoch\_12/archive/data中的一个文件，torch/serialization.py报错RuntimeError，找不到文件。说明data.pkl文件被读取了并发挥了一定作用。

继续研究onnx2ncnn失败的问题。

不定时会出现这种错误，不管它重新运行程序就行：**mmdeploy - ERROR - `mmdeploy.apis.pytorch2onnx.torch2onnx` with Call id: 0 failed. exit.**

更新ncnn后，错误一般是：

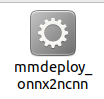
**assert ret\_code == 0, 'onnx2ncnn failed'**

**AssertionError: onnx2ncnn failed**

报错的代码位于mmdeploy/backend/ncnn/onnx2ncnn.py：

onnx2ncnn\_path = get\_onnx2ncnn\_path()  
ret\_code = call([onnx2ncnn\_path, onnx\_path, save\_param, save\_bin])  
**assert** ret\_code == 0, **'onnx2ncnn failed'**

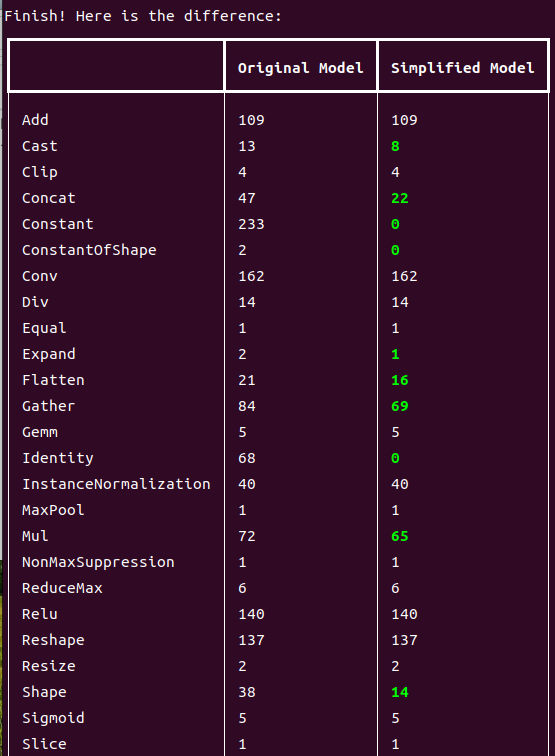
上述代码中，第一句正确找到了转换程序，各参数都正确。因此问题应该出在其执行过程，或者是生成的中间模型end2end.onnx不能使用：

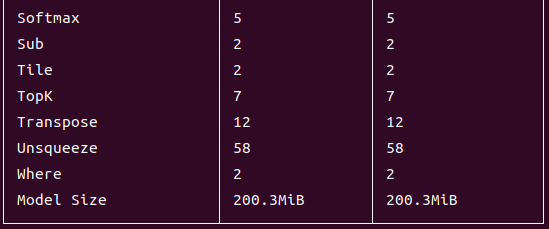


我用教师模型在onnxruntime框架下的转换的onnx模型进行推理，能跑通但是结果不正常，不知道是模型的问题还是我的打开方式不对。

|  |  |  |
| --- | --- | --- |
|  | Teacher | Student |
| ONNX | √ ？ | 'NoneType' object |
| ncnn | onnx2ncnn failed | / |

使用ncnn内的onnx2ncnn工具尝试对pytorch的简单Resnet18模型进行转换、简化（不简化会报错**Identity not supported yet!**数量是16个，原模型简化后Identity的数量16=>0）和最终转换，成功。使用教师模型的onnx模型，可以简化，不能进行下一步转换。

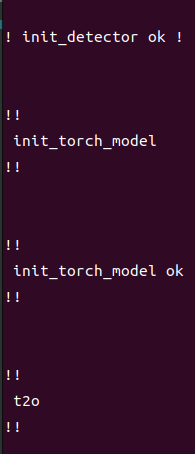
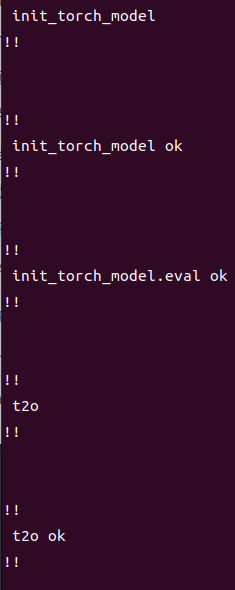




报的错误是**Unknown data type 0**，共有21个，最后一行报错**Segmentation fault (core dumped)**。

2023年3月23日星期四

找到了模型导入时的代码，在mmdeploy/apis/inference下。我在模型导入后被传递的各个阶段调用model.forward，发现并不是一直会出现NoneType错误：

学生 教师

原因是某次传递时，返回的不是model而是model.eval()。后者pytorch模型的一种模式，作用是不启用Batch Normalization和Dropout。也许学生网络没有这种模式，而教师网络有，因此可以通过。

将该程序中的返回值model.eval()改为model，则学生模型可以转换为onnx格式，目前进度和教师模型一致了。

2023年3月24日星期五

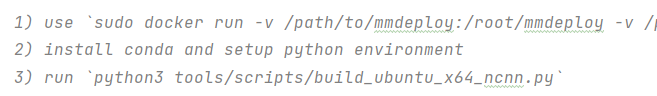
继续研究onnx2ncnn出错的原因，测试样例模型时发现转换工具mmdeploy\_onnx2ncnn找不到库libprotobuf.so.31。它们应该是在build\_ubuntu\_x64\_ncnn.py里一起编译安装的。因为无法查看转换工具是如何查找依赖库的，我只能从一键编译的python文件里确定它们应该的相对位置。或者由于此时的onnx文件已经相对独立，我们可以使用第三方的工具进行转换。

把mmdeploy-dep下的pbinstall挪到mmdeploy下就可以转换样例模型了，但是教师模型和学生模型仍然报错。

尝试用另一个虚拟电脑重装环境，花了很多时间。突然想到，为什么不把mmdeploy-dep下的其他文件夹也复制到mmdeploy里去呢？可惜没什么用。不过样例模型的情况提醒我们，确实可能是ncnn环境安装的问题。

2023年3月27日星期一

mmdeploy配置ncnn的代码的main函数最前面有这么一段：



我安装了miniconda，学着使用docker试了一遍，感觉没有什么不同，运行python程序时还死机了。Miniconda的存在使得conda命令都不好用了（环境变量被覆盖了）。

重试之前的虚拟电脑，发现又不好使了，这次挪pbinstall也没用了。通过export可以解决找不到库的问题，但ret==0依旧。

2023年3月28日星期二

用ncnn的转换工具转换学生模型，无论是否简化都是那个结果。21个位置数据类型也很难跟哪个参数对应起来（经过与样例模型的对比，错误应该与模型简化参数表中的参数都无关）。样例模型转化时，也会出现许多位置数据类型错误，但可以完成转化，因此问题应该还是出在段错误上。

**libonnxruntime.so.1.8.1: cannot open shared object file**

**sudo ldconfig /home/pirlo/PycharmProjects/pythonProject/onnxruntime-linux-x64-1.8.1/lib**

暂时解决了onnxruntime推理的问题，跑出了一张似曾相识的看不懂的图片。我把Windows测试教师模型的代码上传了（学生模型的onnx文件之前也上传了）。

2023年3月30日星期四

今天突然想到可能是我的推理代码有问题，毕竟mmopenlab的推理代码没跑通，onnxruntime的推理代码可能不适配。我用转化程序重新生成中间模型时发现同时生成了两幅测试图片，于是我利用这种方式测试我们的数据图片，结果返回了正常的图片。看来我们的模型转化没有问题，只是必须使用mmopenlab的东西来跑。

现在任务有三：

1. onnx2ncnn，目前还在debug
2. ncnn模型封装。包括封装成模块（主要）和封装成app。后者虽然有demo工程参考，但是要根据模型写jni.cpp可能还是比较困难。
3. 程序部署。我本想先拿之前编译好的demo工程试试的，但是板子的屏幕和电阻都不太好用，我不敢轻举妄动，暂时搁置了。

接下来主要解决任务1。我考虑过是否是cpu/cuda的问题，但是网上也有cpu部署成功的言论。我之前用ncnn的工具成功转化了样例模型，但教师和学生模型还是不行。

我发现直接调用mmdeploy里的转换工具也会报段错误。使用“--no-large-tensor”命令，模型简化过程无warning，可惜段错误依旧。用该工具转换测试用的resnet18，也能完成转换。看来问题还是和模型有关。

Segment fault（core dumped）一般是因为野指针或数组越界造成的，但是可执行程序怎么调试呢？我记得openmmlab的社区开放麦里似乎提到过类似的东西，但那是在写代码的层面，明天再啃啃。

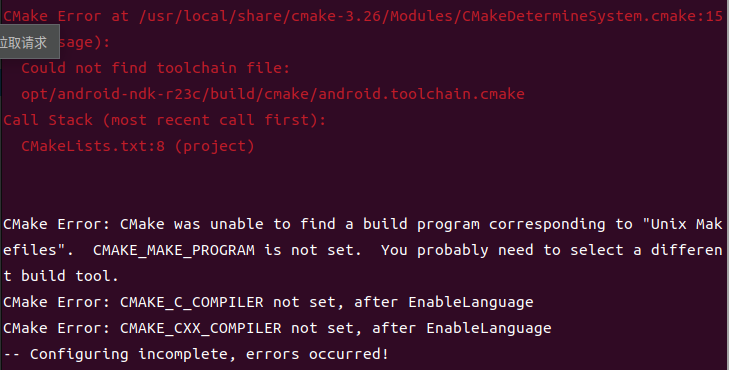
2023年3月31日星期五

通过查看系统日志进一步定位问题为ip 0 访问非法内存，error4 中断系统调用。mmdeploy和ncnn的报错都是这样的。我尝试通过反汇编定位错误，但是没有找到，准确地说我没法通过偏移地址找出出错的汇编语句。

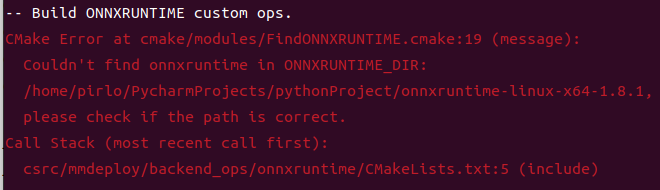
2023年4月6日星期四

本周从deploy.py中提取了调用模型的最小化代码，并在ubuntu中安装了android studio。

进行android-onnx-SDK的cmake时报错：



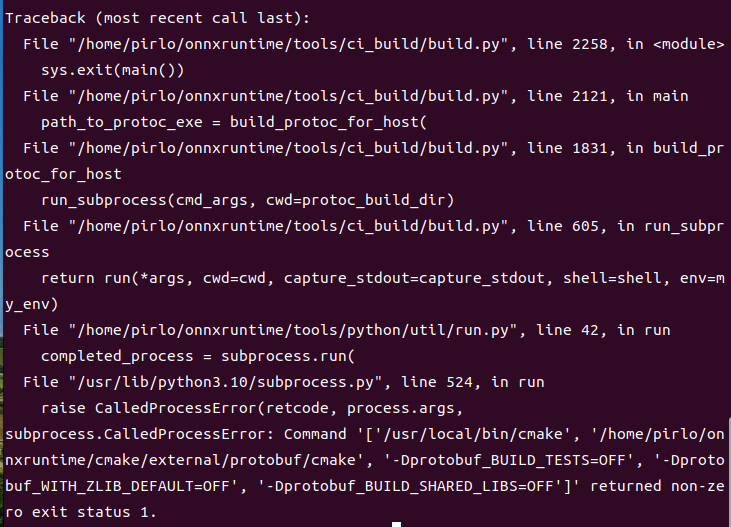
原来这部分cmake要在mmdeploy目录下进行。



这就很奇怪了，因为onnxruntime-linux的cmake也有这个路径，是没问题的。于是我参照ncnn-android的cmake，认为可能是需要onnxruntime内部先编译一下。Onnxruntime-android的官方文档<https://onnxruntime.ai/docs/build/android.html>做得很不好。AndroidStudio和cmdline-tool下载sdk的默认路径居然还是不一样的。更无语的是后面莫名其妙多了个build.sh文件，它在我们下载的android目录和onnxruntime目录下都是没有的。原来是在onnxruntime最新的工程里。然后build.sh也跑不通。

只能从bin目录下打开命令行工具也是个问题，解决方法也很离奇：<https://blog.csdn.net/xuxiobo5/article/details/115469446>

我退掉了之前安装的sdk和ndk，按照文档用命令行安装。Build结果如下：



Window侧编译也遇到很多问题，除了和ubuntu侧相同的问题外，还报出大量的failed to connect to github.com port 443等。

**常用命令**

环境变量

~/.bashrc

磁盘清理

Anaconda清理

sudo apt autoremove