**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。**

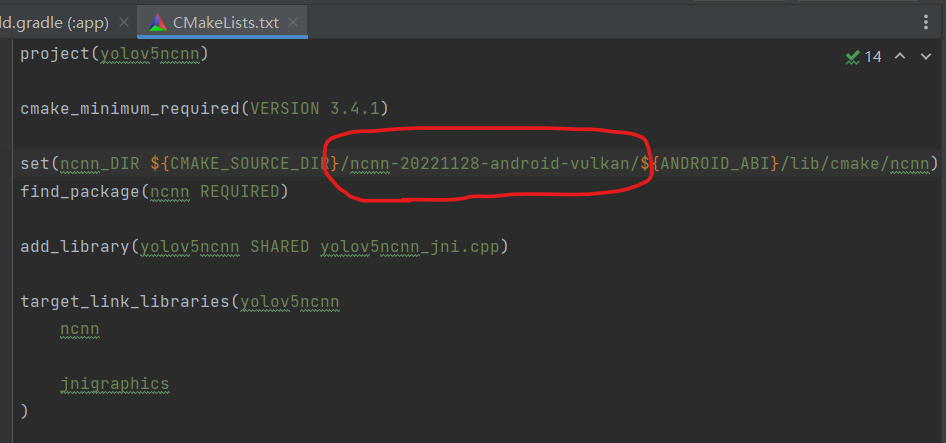
**解决方案：手机要打开开发者模式，打开“USB调试”及其下的“USB安装”。如果仍然报如上错误，可断开连接，确认开启上述按钮后重新连接。**

**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**

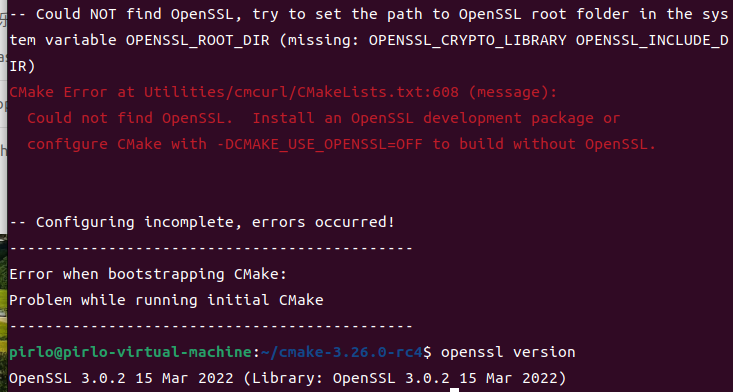
安装cmake <https://blog.csdn.net/qq_38964360/article/details/126119740>

**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)**





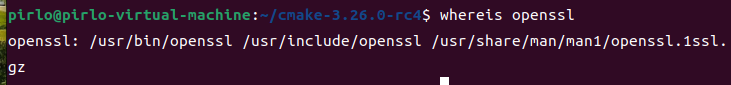
 添加环境变量无效

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QAQ。

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？之前也抱着个错误吗？

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。

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。



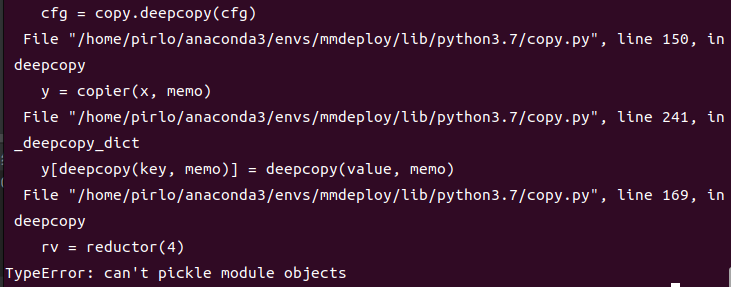
问题似乎浮出了水面，其实错的不是deploy\_config\_path，而是model\_config\_path。我清晰地记得一开始我是写对的，然后很快鬼使神差地把.py和.pth的路径交换了一下。当然换回来之后还是跑不通。



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。

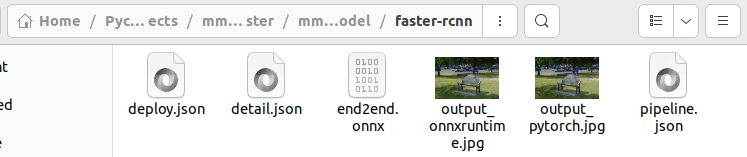
虚拟机扩容到75G。

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/>

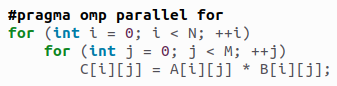
步骤参考：<https://blog.csdn.net/u014689845/article/details/86692079>

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。

**常用命令**

环境变量

~/.bashrc

磁盘清理

Anaconda清理