7.1、在Caffe的移植中加入OpenMP

BVLC的Caffe原版代码中并没有考虑加入OpenMP并行化，其多线程并行化主要体现在其依赖库上。如，OpenBLAS、ATLAS等CPU运行的BLAS支持OpenMP并行化，GPU上使用NVIDIA的CUDA通用计算库可以支持GPU的多线程并行。故，我们如果要在Caffe本身的代码中加入OpenMP并行化，需要在编译脚本中加入对OpenMP的支持。

Caffe借助CMake寻找依赖包，生成编译脚本。我们需要在caffe-android-lib中的Caffe CMakelist.txt中加入对OpenMP的支持：

#add OpenMP

find\_package(OpenMP)

if (OPENMP\_FOUND)

message(“OPENMP FOUND”)

set (CMAKE\_C\_FLAGS "${CMAKE\_C\_FLAGS} ${OpenMP\_C\_FLAGS}")

set (CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} ${OpenMP\_CXX\_FLAGS}")

endif()

7.2、线程数指定、环境变量设定

为支持OpenMP的软件指定线程数主要有两种方法：

·显式：在每处OpenMP预编译宏中，显式地给出受这个宏控制的并行化代码应当具有的线程数量，如：#pragma omp parallel num\_threads(8)

·隐式：在代码中不指定并行化代码，运行之前设定环境变量OMP\_NUM\_THREADS来控制线程数。如，export OMP\_NUM\_THREADS=8

根据我们测试平台的ARM CPU特性，同时考虑到测试的便利性，我们采用后一种方式，通过设置环境变量来控制线程数为4。

由于Caffe最后被编译为动态链接库caffe.so供可执行文件caffe-time.bin运行时链接，我们还需要在环境变量LD\_LIBRARY\_PATH中加入动态链接库caffe.so的路径，以确保库能够被可执行文件找到。

综上，执行caffe-time.bin之前，我们通过adb shell运行如下脚本设定环境变量：

#!/system/bin/sh

export LD\_LIBRARY\_PATH=/data/local/tmp/caffe-bwbranch:$LD\_LIBRARY\_PATH

export OMP\_NUM\_THREADS=4