基于 AM57x 的机器学习案例

Revision History

Draft Date	Revision No.	Description
2018/07/27	V1.0	1. 初始版本。

目　　录

1 YOLO Darknet 移植与应用开发 3
1.1 编译 darknet-master 工程 3
1.2 下载 YOLO 训练模型文件 6
1.3 开发板运行程序 6
2 Caffe 框架移植 9
2.1 Ubuntu 移植环境搭建 9
2.1.1 安装 g++编译器 9
2.1.2 安装 cmake 工具 10
2.2 交叉编译 Caffe 依赖库 14
2.2.1 配置交叉编译工具 14
2.2.2 编译 Openblas 依赖库 15
2.2.3 编译 snnapy 依赖库 17
2.2.4 编译 leveldb 依赖库 20
2.2.5 编译 lmdb 依赖库 22
2.2.6 编译 gflag 依赖库 25
2.2.7 编译 glog 依赖库 27
2.3 移植 Caffe 框架到开发板 30
2.3.1 编译 hdf5 依赖库 31
2.3.2 移植 caffe-master 到开发板 33
2.3.3 Caffe 框架测试 39
3 基于 Caffe 框架的人脸检测案例 42
3.1 编译 face_detection 43
3.2 人脸检测案例 45
更多帮助 47

1 YOLO Darknet 移植与应用开发

表 1

开发板型号	是否支持本例程
TL5728-EasyEVM	支持
TL5728-IDK	支持
TL5728F-EVM	支持

YOLO 是基于深度学习方法的端到端实时目标检测系统，Darknet 是 YOLO 的实现，但 Darknet 不仅包含 YOLO 的实现，还包括其它内容。

本实验主要介绍将 YOLO Darknet 机器学习框架移植到 AM5728 平台的方法，并且演示一个图片信息识别的 darknet-maste 案例。

为便于测试，我司提供经验证的测试文件位于光盘“Demo\app\Yolo_Darknet\bin”目录下。bin 文件夹含有：所需下载的 yolo.weights 文件，编译生成的 darknet 可执行文件，测试所需 cfg 文件夹（配置文件）以及 data 文件夹（图像数据）。以下提供 darknet-maste 工程的具体编译方法。

1.1 编译 darknet-master 工程

打开 Ubuntu，执行如下指令新建“home/tronlong/AM57xx/Yolo_Darknet”工作目录。

Host# mkdir -p /home/tronlong/AM57xx/Yolo_Darknet

```
tronlong@tronlong-virtual-machine:~$ mkdir -p /home/tronlong/AM57xx/Yolo_Darknet
tronlong@tronlong-virtual-machine:~$
```

图 1

将光盘“Demo\app\Yolo_Darknet\src\darknet-master.zip”工程源码压缩文件拷贝到“home/tronlong/AM57xx/Yolo_Darknet”目录，并将其解压到当前目录。

Host# unzip darknet-master.zip

```
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet$ pwd
/home/tronlong/AM57xx/Yolo_Darknet
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet$ ls
darknet-master.zip
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet$ unzip darknet-master.zip
```

图 2

```
  inflating: darknet-master/src/tree.h
  inflating: darknet-master/src/upsample_layer.c
  inflating: darknet-master/src/upsample_layer.h
  inflating: darknet-master/src/utils.c
  inflating: darknet-master/src/utils.h
  inflating: darknet-master/src/yolo_layer.c
  inflating: darknet-master/src/yolo_layer.h
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet$ ls
darknet-master  darknet-master.zip
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet$
```

图 3

进入 darknet-master 源码安装目录，修改 Makefile 文件，如下图所示：

Host# cd darknet-master/

Host# vi Makefile

```
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet$ cd darknet-master/
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet/darknet-master$ ls
cfg       include       LICENSE.gen   LICENSE.mit  python     src
data      LICENSE       LICENSE.gpl   LICENSE.v1   README.md
examples  LICENSE.fuck  LICENSE.meta  Makefile     scripts
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet/darknet-master$ vi Makefile
```

图 4

修改内容如下：

- CC=arm-linux-gnueabihf-gcc
- AR=arm-linux-gnueabihf-ar

```
16 VPATH=./src/:./examples
17 SLIB=libdarknet.so
18 ALIB=libdarknet.a
19 EXEC=darknet
20 OBJDIR=./obj/
21
22 CC=arm-linux-gnueabihf-gcc
23 NVCC=nvcc
24 AR=arm-linux-gnueabihf-ar
25 ARFLAGS=rcs
26 OPTS=-Ofast
27 LDFLAGS= -lm -pthread
```

图 5

修改完 Makefile 文件保存，执行如下命令，使用对应平台的 Linux Processor-SDK 加载环境变量。Linux Processor-SDK 路径请根据实际情况修改。

Host# source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup

```
tronlong@tronlong-virtual-machine:~/AM57xx/Yolo_Darknet/darknet-master$ source /
home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environ
ment-setup
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master>
```

图 6

执行 make 指令，编译 darknet-master 工程，编译完成后在当前目录下生成 darknet 可执行文件和其他文件。

Host# make

```
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master> make
mkdir -p obj
mkdir -p results
arm-linux-gnueabihf-gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-
pragmas -Wfatal-errors -fPIC -Ofast -c ./src/gemm.c -o obj/gemm.o
arm-linux-gnueabihf-gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-
pragmas -Wfatal-errors -fPIC -Ofast -c ./src/utils.c -o obj/utils.o
./src/utils.c: In function 'fgetl':
./src/utils.c:349:24: warning: format '%ld' expects argument of type 'long int',
 but argument 2 has type 'size_t {aka unsigned int}' [-Wformat=]
                printf("%ld\n", size);
```

图 7

```
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master> make
make: Nothing to be done for `all'.
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master> ls
cfg       include        LICENSE.fuck  LICENSE.mit  python     src
darknet   libdarknet.a   LICENSE.gen   LICENSE.v1   README.md
data      libdarknet.so  LICENSE.gpl   Makefile     results
examples  LICENSE        LICENSE.meta  obj          scripts
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master>
```

图 8

1.2 下载 YOLO 训练模型文件

在 darknet-master 目录下执行以下指令，下载 YOLO 训练模型文件到当前目录，文件名为 yolo.weights，如下图所示。

Host# wget https://pjreddie.com/media/files/yolo.weights

```
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master> pwd
/home/tronlong/AM57xx/Yolo_Darknet/darknet-master
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master> wget https://pjreddie.com/m
edia/files/yolo.weights
--2018-07-18 16:49:18--  https://pjreddie.com/media/files/yolo.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.3.39
Connecting to pjreddie.com (pjreddie.com)|128.208.3.39|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 203934260 (194M) [application/octet-stream]
Saving to: 'yolo.weights'

100%[======================================>] 203,934,260  118KB/s   in 15m 19s

2018-07-18 17:04:38 (217 KB/s) - 'yolo.weights' saved [203934260/203934260]

[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master> ls
cfg       include        LICENSE.fuck  LICENSE.mit  python     src
darknet   libdarknet.a   LICENSE.gen   LICENSE.v1   README.md  yolo.weights
data      libdarknet.so  LICENSE.gpl   Makefile     results
examples  LICENSE        LICENSE.meta  obj          scripts
[linux-devkit]:~/AM57xx/Yolo_Darknet/darknet-master>
```

图 9

下载完成后，将编译过的整个 darknet-master 文件夹拷贝到开发板文件系统“/home/root”目录下。

1.3 开发板运行程序

开发板上电进入文件系统 darknet-master 目录，执行以下指令，运行程序，如下图所示：

Target# cd darknet-master/

Target# ./darknet detect cfg/yolov2.cfg yolo.weights data/dog.jpg

```
root@AM57xx-Tronlong:~# ls
darknet-master
root@AM57xx-Tronlong:~# cd darknet-master/
root@AM57xx-Tronlong:~/darknet-master# ./darknet detect cfg/yolov2.cfg yolo.weights da
dog.jpg
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32  0.299 BFLOPs
    1 max          2 x 2 / 2   416 x 416 x  32   ->   208 x 208 x  32
    2 conv     64  3 x 3 / 1   208 x 208 x  32   ->   208 x 208 x  64  1.595 BFLOPs
    3 max          2 x 2 / 2   208 x 208 x  64   ->   104 x 104 x  64
    4 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128  1.595 BFLOPs
    5 conv     64  1 x 1 / 1   104 x 104 x 128   ->   104 x 104 x  64  0.177 BFLOPs
    6 conv    128  3 x 3 / 1   104 x 104 x  64   ->   104 x 104 x 128  1.595 BFLOPs
    7 max          2 x 2 / 2   104 x 104 x 128   ->    52 x  52 x 128
    8 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
    9 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
   10 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
   11 max          2 x 2 / 2    52 x  52 x 256   ->    26 x  26 x 256
   12 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   13 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   14 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   15 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   16 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   17 max          2 x 2 / 2    26 x  26 x 512   ->    13 x  13 x 512
   18 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   19 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   20 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   21 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   22 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   23 conv   1024  3 x 3 / 1    13 x  13 x1024   ->    13 x  13 x1024  3.190 BFLOPs
   24 conv   1024  3 x 3 / 1    13 x  13 x1024   ->    13 x  13 x1024  3.190 BFLOPs
   25 route  16
   26 conv     64  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x  64  0.044 BFLOPs
   27 reorg              / 2    26 x  26 x  64   ->    13 x  13 x 256
   28 route  27 24
   29 conv   1024  3 x 3 / 1    13 x  13 x1280   ->    13 x  13 x1024  3.987 BFLOPs
   30 conv    425  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 425  0.147 BFLOPs
   31 detection
mask_scale: Using default '1.000000'
Loading weights from yolo.weights...Done!
data/dog.jpg: Predicted in 37.349659 seconds.
dog: 81%
truck: 74%
bicycle: 83%
root@AM57xx-Tronlong:~/darknet-master# ls
LICENSE          LICENSE.mit      darknet          libdarknet.so    scripts
LICENSE.fuck     LICENSE.v1       data             obj              src
LICENSE.gen      Makefile         examples         predictions.png  yolo.weights
LICENSE.gpl      README.md        include          python
LICENSE.meta     cfg              libdarknet.a     results
root@AM57xx-Tronlong:~/darknet-master#
```

图 10

程序运行完毕后，会在 darknet-master 目录下生成 predictions.png 文件，将 predictions.png 文件拷贝到 PC 端查看，图片效果如下所示。

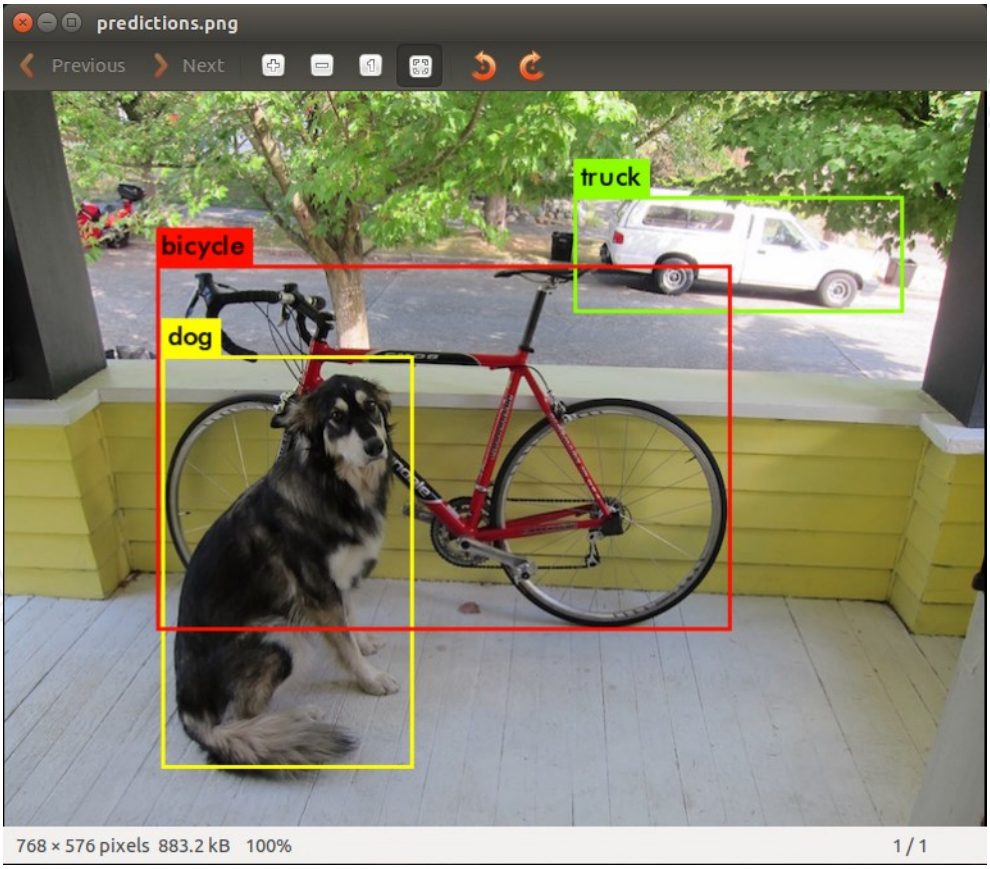
predictions.png
Previous
Next
truck
bicycle
dog
768 × 576 pixels 883.2 kB 100%
1 / 1

图 11

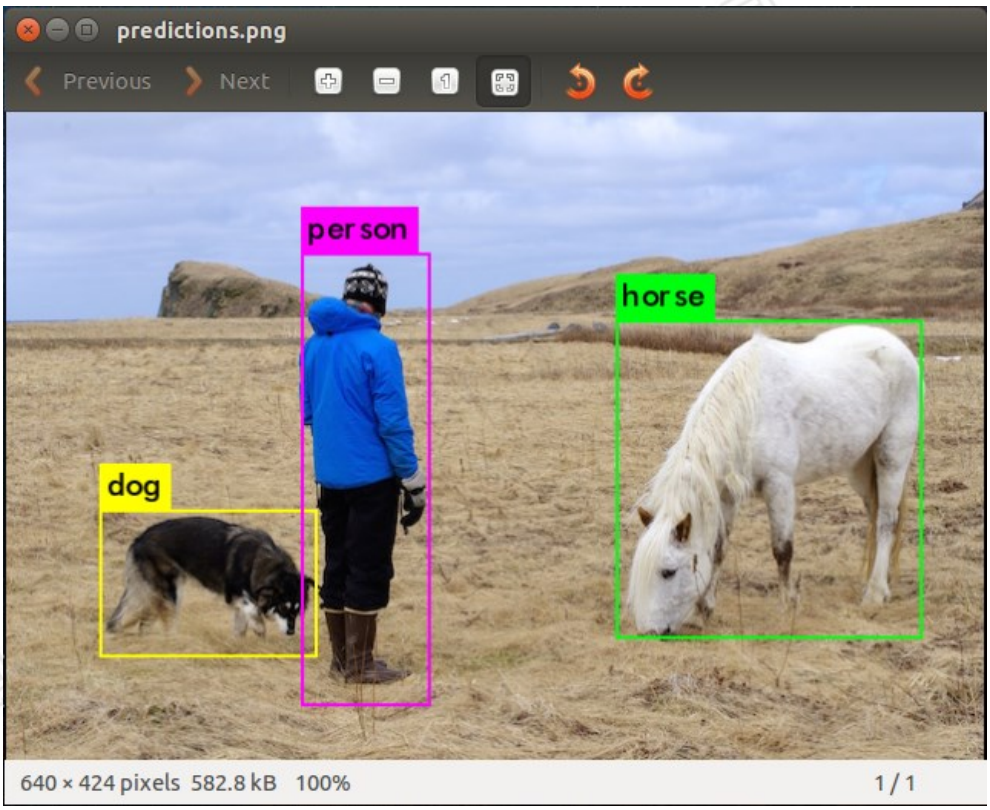
predictions.png
Previous
Next
person
horse
dog
640 × 424 pixels 582.8 kB 100%
1 / 1

图 12

2 Caffe 框架移植

表 2

开发板型号	是否支持本实验
TL5728-EasyEVM	支持
TL5728-IDK	支持
TL5728F-EVM	支持

本文档主要演示将 Caffe 移植到 ARM 平台的方法，本实验编译的 Caffe 依赖库要求 CMake 版本高于 cmake-3.9，同时我司提供的 cmake-3.11 版本要求 Ubuntu g++编译器支持 C++11 标准。

2.1 Ubuntu 移植环境搭建

2.1.1 安装 g++编译器

由于我司默认使用的 Ubuntu 14.04 版本默认情况没有安装 g++编译器，执行如下指令安装 g++编译器，输入“Y”确认安装。

Host# sudo apt-get install g++

```
tronlong@tronlong-virtual-machine:~$ sudo apt-get install g++
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  g++-4.8 libstdc++-4.8-dev
Suggested packages:
  g++-multilib g++-4.8-multilib gcc-4.8-doc libstdc++6-4.8-dbg
  libstdc++-4.8-doc
The following NEW packages will be installed:
  g++ g++-4.8 libstdc++-4.8-dev
0 upgraded, 3 newly installed, 0 to remove and 415 not upgraded.
Need to get 19.1 MB of archives.
After this operation, 40.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

图 13

安装成功后，执行如下指令查看 g++编译器版本信息：

Host# g++ -v

```
Selecting previously unselected package g++-4.8.
Preparing to unpack .../g++-4.8_4.8.4-2ubuntu1~14.04.4_amd64.deb ...
Unpacking g++-4.8 (4.8.4-2ubuntu1~14.04.4) ...
Selecting previously unselected package g++.
Preparing to unpack .../g++_4%3a4.8.2-1ubuntu6_amd64.deb ...
Unpacking g++ (4:4.8.2-1ubuntu6) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up libstdc++-4.8-dev:amd64 (4.8.4-2ubuntu1~14.04.4) ...
Setting up g++-4.8 (4.8.4-2ubuntu1~14.04.4) ...
Setting up g++ (4:4.8.2-1ubuntu6) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mo
de
tronlong@tronlong-virtual-machine:~$ g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/4.8/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 4.8.4-2ubuntu1~14
.04.4' --with-bugurl=file:///usr/share/doc/gcc-4.8/README.Bugs --enable-language
s=c,c++,java,go,d,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.8 --ena
ble-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-get
text --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.8 --libdir
=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-
debug --enable-libstdcxx-time=yes --enable-gnu-unique-object --disable-libmudfla
p --enable-plugin --with-system-zlib --disable-browser-plugin --enable-java-awt=
gtk --enable-gtk-cairo --with-java-home=/usr/lib/jvm/java-1.5.0-gcj-4.8-amd64/jr
e --enable-java-home --with-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-4.8-amd64 -
-with-jvm-jar-dir=/usr/lib/jvm-exports/java-1.5.0-gcj-4.8-amd64 --with-arch-dire
ctory=amd64 --with-ecj-jar=/usr/share/java/eclipse-ecj.jar --enable-objc-gc --en
able-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multil
ib-list=m32,m64,mx32 --with-tune=generic --enable-checking=release --build=x86_6
4-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.4)
tronlong@tronlong-virtual-machine:~$
```

图 14

2.1.2 安装 cmake 工具

将位于光盘“Demo\app\caffe_transplant”下的“cmake-3.11.0.tar.gz”压缩文件拷贝到 Ubuntu AM57xx 工作目录，执行如下指令新建“/home/tronlong/AM57xx/caffe_transplant”文件夹，并将 cmake-3.11.0.tar.gz 压缩包解压到该目录下。

Host# mkdir -p /home/tronlong/AM57xx/caffe_transplant

Host# tar -xvf cmake-3.11.0.tar.gz -C caffe_transplant/

```
tronlong@tronlong-virtual-machine:~/AM57xx$ pwd
/home/tronlong/AM57xx
tronlong@tronlong-virtual-machine:~/AM57xx$ ls cmake-3.11.0.tar.gz
cmake-3.11.0.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx$ mkdir -p /home/tronlong/AM57xx/caffe
_transplant
tronlong@tronlong-virtual-machine:~/AM57xx$ tar -xvf cmake-3.11.0.tar.gz -C caff
e_transplant/
```

图 15

```
cmake-3.11.0/Utilities/Sphinx/fixup_qthelp_names.cmake
cmake-3.11.0/Utilities/Sphinx/static/
cmake-3.11.0/Utilities/Sphinx/static/cmake-favicon.ico
cmake-3.11.0/Utilities/Sphinx/static/cmake-logo-16.png
cmake-3.11.0/Utilities/Sphinx/static/cmake.css
cmake-3.11.0/Utilities/Sphinx/templates/
cmake-3.11.0/Utilities/Sphinx/templates/layout.html
tronlong@tronlong-virtual-machine:~/AM57xx$ ls caffe_transplant/cmake-3.11.0/
Auxiliary                  configure              Modules
bootstrap                  CONTRIBUTING.rst       Packaging
CMakeCPack.cmake           Copyright.txt          README.rst
CMakeCPackOptions.cmake.in CTestConfig.cmake      Source
CMakeGraphVizOptions.cmake CTestCustom.cmake.in   Templates
CMakeLists.txt             DartConfig.cmake       Tests
CMakeLogo.gif              doxygen.config         Utilities
cmake_uninstall.cmake.in   Help
CompileFlags.cmake         Licenses
tronlong@tronlong-virtual-machine:~/AM57xx$
```

图 16

进入 cmake-3.11.0 安装目录，执行 bootstrap 脚本文件。

Host# cd caffe_transplant/cmake-3.11.0/

Host# sudo ./bootstrap

```
tronlong@tronlong-virtual-machine:~/AM57xx$ cd caffe_transplant/cmake-3.11.0/
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ ls
Auxiliary                   configure             Modules
bootstrap                   CONTRIBUTING.rst      Packaging
CMakeCPack.cmake            Copyright.txt         README.rst
CMakeCPackOptions.cmake.in  CTestConfig.cmake     Source
CMakeGraphVizOptions.cmake  CTestCustom.cmake.in  Templates
CMakeLists.txt              DartConfig.cmake      Tests
CMakeLogo.gif               doxygen.config        Utilities
cmake_uninstall.cmake.in    Help
CompileFlags.cmake          Licenses
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ sudo
./bootstrap
[sudo] password for tronlong:
-----------------------------------------
CMake 3.11.0, Copyright 2000-2018 Kitware, Inc. and Contributors
Found GNU toolchain
C compiler on this system is: gcc
C++ compiler on this system is: g++  -std=gnu++1y
Makefile processor on this system is: make
g++ has setenv
g++ has unsetenv
g++ does not have environ in stdlib.h
g++ has stl wstring
g++ has <ext/stdio_filebuf.h>
-----------------------------------------
g++  -std=gnu++1y         -I/home/tronlong/AM57xx/caffe_transplant/cmake-3.11.0
/Bootstrap.cmk   -I/home/tronlong/AM57xx/caffe_transplant/cmake-3.11.0/Source
 -I/home/tronlong/AM57xx/caffe_transplant/cmake-3.11.0/Source/LexerParser   -I/
home/tronlong/AM57xx/caffe_transplant/cmake-3.11.0/Utilities  -c /home/tronlong
/AM57xx/caffe_transplant/cmake-3.11.0/Source/cmAddCustomCommandCommand.cxx -o c
mAddCustomCommandCommand.o
```

图 17

```
-- Looking for elf.h
-- Looking for elf.h - found
-- Looking for a Fortran compiler
-- Looking for a Fortran compiler - NOTFOUND
qmake: could not exec '/usr/lib/x86_64-linux-gnu/qt4/bin/qmake': No such file or
 directory
qmake: could not exec '/usr/lib/x86_64-linux-gnu/qt4/bin/qmake': No such file or
 directory
-- Performing Test run_pic_test
-- Performing Test run_pic_test - Success
-- Performing Test run_inlines_hidden_test
-- Performing Test run_inlines_hidden_test - Success
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tronlong/AM57xx/caffe_transplant/cmak
e-3.11.0
-----------------------------------------
CMake has bootstrapped.  Now run make.
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$
```

图 18

执行以下指令，编译和安装 cmake 工具。

Host# sudo make

Host# sudo make install

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ sudo make
Scanning dependencies of target cmsys
[  0%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/ProcessUNIX.c.o
[  1%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/Base64.c.o
[  1%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/EncodingC.c.o
[  1%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/MD5.c.o
[  1%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/Terminal.c.o
[  1%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/System.c.o
[  1%] Building C object Source/kwsys/CMakeFiles/cmsys.dir/String.c.o
[  1%] Building CXX object Source/kwsys/CMakeFiles/cmsys.dir/Directory.cxx.o
[  1%] Building CXX object Source/kwsys/CMakeFiles/cmsys.dir/DynamicLoader.cxx.o
[  1%] Building CXX object Source/kwsys/CMakeFiles/cmsys.dir/EncodingCXX.cxx.o
[  2%] Building CXX object Source/kwsys/CMakeFiles/cmsys.dir/Glob.cxx.o
[  2%] Building CXX object Source/kwsys/CMakeFiles/cmsys.dir/RegularExpression.cxx.o
```

图 19

```
[ 99%] Linking C executable pseudo_emulator_custom_command
[ 99%] Built target pseudo_emulator_custom_command
Scanning dependencies of target pseudo_tidy
[ 99%] Building C object Tests/RunCMake/CMakeFiles/pseudo_tidy.dir/pseudo_tidy.c.o
[ 99%] Linking C executable pseudo_tidy
[ 99%] Built target pseudo_tidy
Scanning dependencies of target pseudo_cppcheck
[ 99%] Building C object Tests/RunCMake/CMakeFiles/pseudo_cppcheck.dir/pseudo_cppche
ck.c.o
[100%] Linking C executable pseudo_cppcheck
[100%] Built target pseudo_cppcheck
Scanning dependencies of target foo
[100%] Building CXX object Tests/FindPackageModeMakefileTest/CMakeFiles/foo.dir/foo.
cpp.o
[100%] Linking CXX static library libfoo.a
[100%] Built target foo
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ sudo make
install
```

图 20

安装成功后，执行以下指令，查看 cmake 工具版本为 3.11.0。

Host# cmake -version

```
-- Up-to-date: /usr/local/share/cmake-3.11/editors/vim/indent
-- Up-to-date: /usr/local/share/cmake-3.11/editors/vim/indent/cmake.vim
-- Up-to-date: /usr/local/share/cmake-3.11/editors/vim/syntax
-- Up-to-date: /usr/local/share/cmake-3.11/editors/vim/syntax/cmake.vim
-- Up-to-date: /usr/local/share/cmake-3.11/editors/emacs/cmake-mode.el
-- Up-to-date: /usr/local/share/aclocal/cmake.m4
-- Up-to-date: /usr/local/share/cmake-3.11/completions/cmake
-- Up-to-date: /usr/local/share/cmake-3.11/completions/cpack
-- Up-to-date: /usr/local/share/cmake-3.11/completions/ctest
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ cmake -ver
sion
cmake version 3.11.0

CMake suite maintained and supported by Kitware (kitware.com/cmake).
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$
```

图 21

2.2 交叉编译 Caffe 依赖库

2.2.1 配置交叉编译工具

在 Ubuntu 下执行如下命令打开“/.bashrc”文件，按照下图方法将 Processor-SDK 开发包中的交叉编译工具链路径加入系统环境变量中，实际路径请根据自身情况修改。

Host# gedit ~/.bashrc

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ gedit ~/.b
ashrc
```

图 22

export PATH="$PATH: /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/"

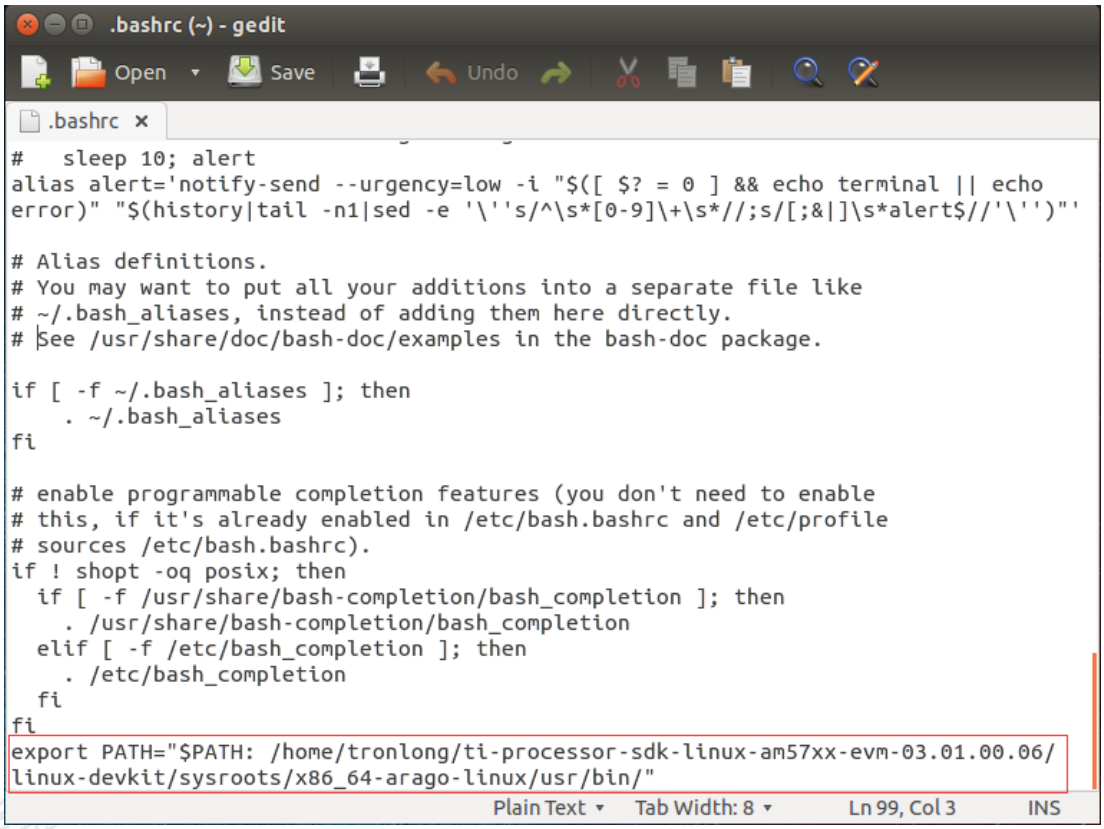

图 23

修改完成后保存退出，执行以下指令，使能“/.bashrc”文件。

Host# source ~/.bashrc

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$ source ~/.
bashrc
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/cmake-3.11.0$
```

图 24

2.2.2 编译 Openblas 依赖库

将位于光盘“Demo\app\caffe_transplant\src”下的“OpenBLAS-0.2.20.tar.gz”压缩文件拷贝到“/home/tronlong/AM57xx/caffe_transplant”目录，执行如下指令将其解压到当前目录下。

Host# tar -xvf OpenBLAS-0.2.20.tar.gz -C .

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ pwd
/home/tronlong/AM57xx/caffe_transplant
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ ls OpenBLAS-0.2.20.tar.
gz
OpenBLAS-0.2.20.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ tar -xvf OpenBLAS-0.2.2
0.tar.gz -C .
```

图 25

在 caffe_transplant 目录下新建一个 tmp 文件夹，文件名可自拟。为便于移植，后面步骤编译生成的其他 Caffe 依赖库文件都会安装到 tmp 文件夹下。

```
OpenBLAS-0.2.20/utest/test_fork.c
OpenBLAS-0.2.20/utest/test_potrs.c
OpenBLAS-0.2.20/utest/test_rot.c
OpenBLAS-0.2.20/utest/test_rotmg.c
OpenBLAS-0.2.20/utest/test_swap.c
OpenBLAS-0.2.20/utest/utest_main.c
OpenBLAS-0.2.20/version.h
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ ls
cmake-3.11.0  OpenBLAS-0.2.20  OpenBLAS-0.2.20.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ mkdir tmp
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ ls
cmake-3.11.0  OpenBLAS-0.2.20  OpenBLAS-0.2.20.tar.gz  tmp
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$
```

图 26

进入 OpenBLAS-0.2.20 目录，执行以下指令编译 OpenBLAS 依赖库，编译完成如下图所示。

Host# cd OpenBLAS-0.2.20/

Host# make TARGET=ARMV7 HOSTCC=gcc BINARY=32 CC=arm-linux-gnueabihf-gcc NOFORTRAN=1

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ cd OpenBLAS-0.2.20/
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/OpenBLAS-0.2.20$ mak
e TARGET=ARMV7 HOSTCC=gcc BINARY=32 CC=arm-linux-gnueabihf-gcc NOFORTRAN=1
OpenBLAS: Detecting fortran compiler failed. Cannot compile LAPACK. Only compile
 BLAS.
make[1]: Entering directory `/home/tronlong/AM57xx/caffe_transplant/OpenBLAS-0.2
.20/interface'
arm-linux-gnueabihf-ar  -ru ../libopenblas_armv7p-r0.2.20.a saxpy.o sswap.o scop
y.o sscal.o sdot.o sdsdot.o dsdot.o sasum.o snrm2.o smax.o samax.o ismax.o isama
x.o smin.o samin.o ismin.o isamin.o srot.o srotg.o srotm.o srotmg.o saxpby.o cbl
as_isamax.o cblas_sasum.o cblas_saxpy.o cblas_scopy.o cblas_sdot.o cblas_sdsdot.
o cblas_dsdot.o cblas_srot.o cblas_srotg.o cblas_srotm.o cblas_srotmg.o cblas_ss
```

图 27

```
make[1]: Leaving directory `/home/tronlong/AM57xx/caffe_transplant/OpenBLAS-0.2.20/e
xports'

 OpenBLAS build complete. (BLAS CBLAS)

  OS               ... Linux
  Architecture     ... arm
  BINARY           ... 32bit
  C compiler       ... GCC  (command line : arm-linux-gnueabihf-gcc)
  Library Name     ... libopenblas_armv7p-r0.2.20.a (Multi threaded; Max num-threads
 is 4)

To install the library, you can run "make PREFIX=/path/to/your/installation install"
.

tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/OpenBLAS-0.2.20$
```

图 28

执行以下指令，将编译生成的 OpenBLAS 依赖库文件安装到指定的“/home/tronlong/AM57xx/caffe_transplant/tmp”目录下，安装路径以实际情况为准。执行安装指令后，依赖库文件将安装在“tmp/lib”目录下，头文件会安装在“tmp/include”目录下，如下图所示：

Host# make PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp install

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/OpenBLAS-0.2.20$ make PR
EFIX=/home/tronlong/AM57xx/caffe_transplant/tmp install
make -j 4 -f Makefile.install install
make[1]: Entering directory `/home/tronlong/AM57xx/caffe_transplant/OpenBLAS-0.2.20'
Generating openblas_config.h in /home/tronlong/AM57xx/caffe_transplant/tmp/include
Generating f77blas.h in /home/tronlong/AM57xx/caffe_transplant/tmp/include
Generating cblas.h in /home/tronlong/AM57xx/caffe_transplant/tmp/include
Copying the static library to /home/tronlong/AM57xx/caffe_transplant/tmp/lib
Copying the shared library to /home/tronlong/AM57xx/caffe_transplant/tmp/lib
Generating openblas.pc in /home/tronlong/AM57xx/caffe_transplant/tmp/lib/pkgconfig
Generating OpenBLASConfig.cmake in /home/tronlong/AM57xx/caffe_transplant/tmp/lib/cm
ake/openblas
Generating OpenBLASConfigVersion.cmake in /home/tronlong/AM57xx/caffe_transplant/tmp
/lib/cmake/openblas
Install OK!
make[1]: Leaving directory `/home/tronlong/AM57xx/caffe_transplant/OpenBLAS-0.2.20'
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/OpenBLAS-0.2.20$
```

图 29

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/OpenBLAS-0.2.20$ cd ../t
mp/
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$ pwd
/home/tronlong/AM57xx/caffe_transplant/tmp
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$ ls
bin  include  lib
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$
```

图 30

2.2.3 编译 snnapy 依赖库

参照前面步骤，将光盘“Demo\app\caffe_transplant\src\google-snappy-1.1.7-6-g4f7bd2d.tar.gz”压缩文件拷贝到 Ubuntu，并解压到“/home/tronlong/AM57xx/caffe_transplant”目录，如下图：

Host# tar -xvf google-snappy-1.1.7-6-g4f7bd2d.tar.gz -C .

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ ls google-snappy-1.1.7-
6-g4f7bd2d.tar.gz
google-snappy-1.1.7-6-g4f7bd2d.tar.gz
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ tar -xvf google-snappy-
1.1.7-6-g4f7bd2d.tar.gz -C .
google-snappy-4f7bd2d/
google-snappy-4f7bd2d/.appveyor.yml
google-snappy-4f7bd2d/.travis.yml
google-snappy-4f7bd2d/AUTHORS
google-snappy-4f7bd2d/CMakeLists.txt
```

图 31

执行如下命令，使用对应平台的 Linux Processor-SDK 加载环境变量，Linux Processor-SDK 路径请根据实际情况修改。进入 google-snappy-4f7bd2d 目录，新建一个 build 文件夹。

Host# source /home/tronlong/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup

Host# cd google-snappy-4f7bd2d/

Host# mkdir build

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant$ source /home/tronlo
ng/ti-processor-sdk-linux-am57xx-evm-03.01.00.06/linux-devkit/environment-setup
[linux-devkit]:~/AM57xx/caffe_transplant> cd google-snappy-4f7bd2d/
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d> ls
AUTHORS                 README.md             snappy-stubs-internal.cc
cmake                   snappy.cc             snappy-stubs-internal.h
CMakeLists.txt          snappy-c.cc           snappy-stubs-public.h.in
CONTRIBUTING.md         snappy-c.h            snappy-test.cc
COPYING                 snappy.h              snappy-test.h
format_description.txt  snappy-internal.h     snappy_unittest.cc
framing_format.txt      snappy-sinksource.cc  testdata
NEWS                    snappy-sinksource.h
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d> mkdir build
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d> ls
AUTHORS                 NEWS                  snappy-sinksource.h
build                   README.md             snappy-stubs-internal.cc
cmake                   snappy.cc             snappy-stubs-internal.h
CMakeLists.txt          snappy-c.cc           snappy-stubs-public.h.in
CONTRIBUTING.md         snappy-c.h            snappy-test.cc
COPYING                 snappy.h              snappy-test.h
format_description.txt  snappy-internal.h     snappy_unittest.cc
framing_format.txt      snappy-sinksource.cc  testdata
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d>
```

图 32

进入 build 目录，执行以下指令，由 cmake -DCMAKE_INSTALL_PREFIX 指定 snnapy 依赖库安装目录。本次测试指定目录为“/home/tronlong/AM57xx/caffe_transplant/tmp”，可根据实际情况修改。

Host# cd build/

Host# cmake -DCMAKE_INSTALL_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..

```
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d> cd build/
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d/build> cmake -DCM
AKE_INSTALL_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..
-- The C compiler identification is GNU 5.3.1
-- The CXX compiler identification is GNU 5.3.1
-- Check for working C compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-ev
m-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabi
hf-gcc
-- Check for working C compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-ev
m-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabi
hf-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-
evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnuea
bihf-g++
-- Check for working CXX compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-
evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnuea
bihf-g++ -- works
-- Detecting CXX compiler ABI info
```

图 33

执行以下指令，编译并安装 snnapy 依赖库到指定的“caffe_transplant/tmp”目录。

Host# make

Host# make install

```
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d/build> make
Scanning dependencies of target snappy
[ 12%] Building CXX object CMakeFiles/snappy.dir/snappy-c.cc.o
[ 25%] Building CXX object CMakeFiles/snappy.dir/snappy-sinksource.cc.o
[ 37%] Building CXX object CMakeFiles/snappy.dir/snappy-stubs-internal.cc.o
[ 50%] Building CXX object CMakeFiles/snappy.dir/snappy.cc.o
[ 62%] Linking CXX static library libsnappy.a
[ 62%] Built target snappy
Scanning dependencies of target snappy_unittest
[ 75%] Building CXX object CMakeFiles/snappy_unittest.dir/snappy_unittest.cc.o
[ 87%] Building CXX object CMakeFiles/snappy_unittest.dir/snappy-test.cc.o
[100%] Linking CXX executable snappy_unittest
[100%] Built target snappy_unittest
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d/build>
```

图 34

```
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d/build> make insta
ll
[ 62%] Built target snappy
[100%] Built target snappy_unittest
Install the project...
-- Install configuration: ""
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/libsnappy.a
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/snappy-c.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/snappy-sinksou
rce.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/snappy.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/snappy-stubs-p
ublic.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/cmake/Snappy/Snapp
yTargets.cmake
Ubuntu Software Center tronlong/AM57xx/caffe_transplant/tmp/lib/cmake/Snappy/Snapp
yTargets-noconfig.cmake
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/cmake/Snappy/Snapp
yConfig.cmake
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/cmake/Snappy/Snapp
yConfigVersion.cmake
[linux-devkit]:~/AM57xx/caffe_transplant/google-snappy-4f7bd2d/build>
```

图 35

2.2.4 编译 leveldb 依赖库

参照前面步骤，将光盘“Demo\app\caffe_transplant\src\leveldb-master.zip”压缩文件拷贝到 Ubuntu，并解压到“/home/tronlong/AM57xx/caffe_transplant”目录，如下图：

Host# unzip leveldb-master.zip

```
[linux-devkit]:~/AM57xx/caffe_transplant> ls
cmake-3.11.0  google-snappy-4f7bd2d  leveldb-master.zip  OpenBLAS-0.2.20  tmp
[linux-devkit]:~/AM57xx/caffe_transplant> unzip leveldb-master.zip
Archive:  leveldb-master.zip
6fa45666703add49f77652b2eadd874d49aedaf6
   creating: leveldb-master/
  inflating: leveldb-master/.gitignore
  inflating: leveldb-master/.travis.yml
  inflating: leveldb-master/AUTHORS
  inflating: leveldb-master/CMakeLists.txt
  inflating: leveldb-master/CONTRIBUTING.md
```

图 36

请使用对应平台的 Linux Processor-SDK 加载环境变量。

进入 leveldb-master 目录，执行以下指令，新建一个 build 文件夹。

Host# cd leveldb-master/

Host# mkdir build

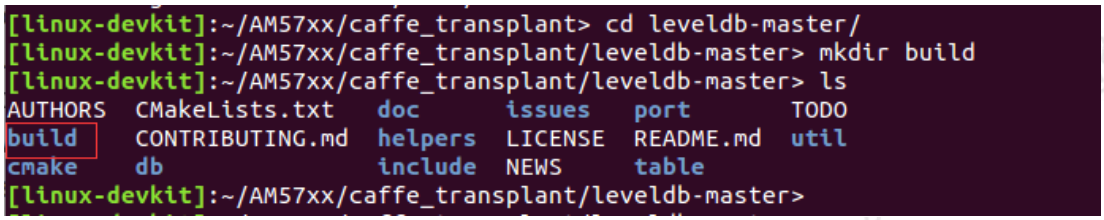

图 37

进入 build 目录，执行以下指令，将指定 leveldb 依赖库安装目录为“/home/tronlong/AM57xx/caffe_transplant/tmp”。

Host# cd build/

Host# CXXFLAGS="-fPIC" cmake -DCMAKE_INSTALL_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..

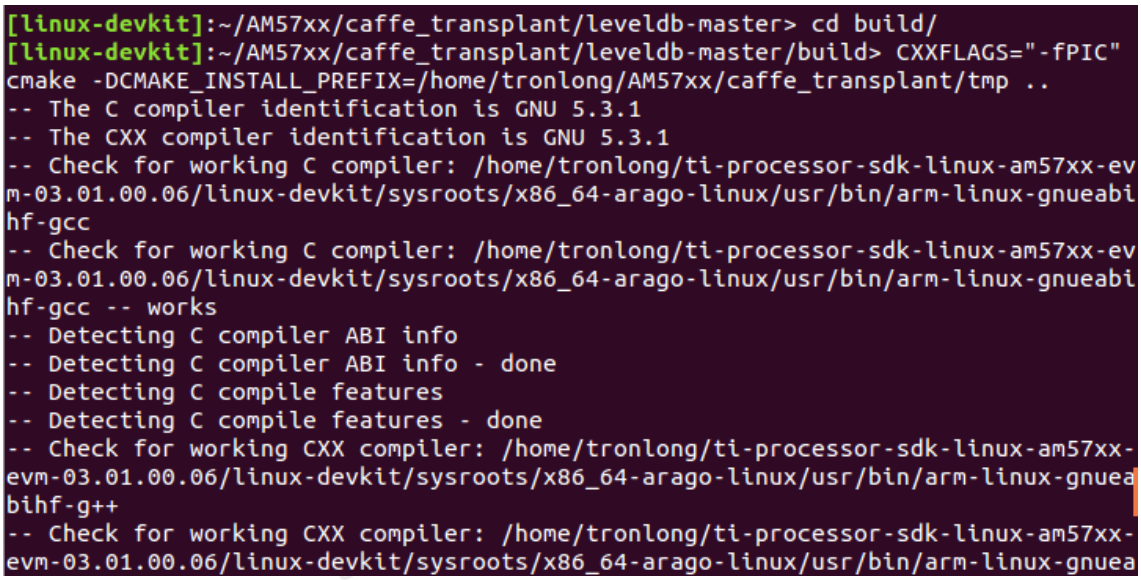

图 38

执行以下指令，编译并安装 leveldb 依赖库到指定的“caffe_transplant/tmp”目录。

Host# make

Host# make install

```
-- Looking for sqlite3_open in sqlite3
-- Looking for sqlite3_open in sqlite3 - found
-- Performing Test HAVE_KYOTOCABINET
-- Performing Test HAVE_KYOTOCABINET - Failed
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tronlong/AM57xx/caffe_transplant/leve
ldb-master/build
[linux-devkit]:~/AM57xx/caffe_transplant/leveldb-master/build> make
Scanning dependencies of target leveldb_port_posix
[  1%] Building CXX object CMakeFiles/leveldb_port_posix.dir/port/port_posix.cc.
o
[  1%] Built target leveldb_port_posix
Scanning dependencies of target leveldb
[  2%] Building CXX object CMakeFiles/leveldb.dir/db/builder.cc.o
[  2%] Building CXX object CMakeFiles/leveldb.dir/db/c.cc.o
[  3%] Building CXX object CMakeFiles/leveldb.dir/db/db_impl.cc.o
```

图 39

```
[ 98%] Building CXX object CMakeFiles/filter_block_test.dir/util/testharness.cc.
o
[ 99%] Building CXX object CMakeFiles/filter_block_test.dir/util/testutil.cc.o
[100%] Building CXX object CMakeFiles/filter_block_test.dir/table/filter_block_t
est.cc.o
[100%] Linking CXX executable filter_block_test
[100%] Built target filter_block_test
[linux-devkit]:~/AM57xx/caffe_transplant/leveldb-master/build> make install
[  1%] Built target leveldb_port_posix
[ 26%] Built target leveldb
[ 29%] Built target db_bench
[ 31%] Built target env_posix_test
```

图 40

2.2.5 编译 lmdb 依赖库

参照前面步骤，将光盘“Demo\app\caffe_transplant\src\lmdb-mdb.master.zip”压缩文件拷贝到 Ubuntu，并解压到“/home/tronlong/AM57xx/caffe_transplant”目录，如下图：

Host# unzip leveldb-master.zip

```
[linux-devkit]:~/AM57xx/caffe_transplant> pwd
/home/tronlong/AM57xx/caffe_transplant
[linux-devkit]:~/AM57xx/caffe_transplant> ls lmdb-mdb.master.zip
lmdb-mdb.master.zip
[linux-devkit]:~/AM57xx/caffe_transplant> unzip lmdb-mdb.master.zip
Archive:  lmdb-mdb.master.zip
0a2622317f189c7062d03d050be6766586a548b2
   creating: lmdb-mdb.master/
   creating: lmdb-mdb.master/libraries/
   creating: lmdb-mdb.master/libraries/liblmdb/
  inflating: lmdb-mdb.master/libraries/liblmdb/.gitignore
  inflating: lmdb-mdb.master/libraries/liblmdb/COPYRIGHT
  inflating: lmdb-mdb.master/libraries/liblmdb/Doxyfile
  inflating: lmdb-mdb.master/libraries/liblmdb/LICENSE
```

图 41

请使用对应平台的 Linux Processor-SDK 加载环境变量。

进入“lmdb-mdb.master/libraries/liblmdb/”目录，执行如下指令打开 Makefile 文件，按照下图内容修改 Makefile 文件内容：

Host# cd lmdb-mdb.master/libraries/liblmdb/

Host# vi Makefile

```
[linux-devkit]:~/AM57xx/caffe_transplant> cd lmdb-mdb.master/libraries/liblmdb/
[linux-devkit]:~/AM57xx/caffe_transplant/lmdb-mdb.master/libraries/liblmdb> ls
COPYRIGHT  Makefile    mdb_drop.c  mdb_stat.1  mtest3.c  sample-bdb.txt
Doxyfile   mdb.c       mdb_dump.1  mdb_stat.c  mtest4.c  sample-mdb.txt
intro.doc  mdb_copy.1  mdb_dump.c  midl.c      mtest5.c  tooltag
LICENSE    mdb_copy.c  mdb_load.1  midl.h      mtest6.c
lmdb.h     mdb_drop.1  mdb_load.c  mtest2.c    mtest.c
[linux-devkit]:~/AM57xx/caffe_transplant/lmdb-mdb.master/libraries/liblmdb> vi
Makefile
```

图 42

修改内容如下：

21 CC = arm-linux-gnueabihf-gcc

22 AR = arm-linux-gnueabihf-ar

30 prefix = /home/tronlong/AM57xx/caffe_transplant/tmp //指定依赖库安装目录

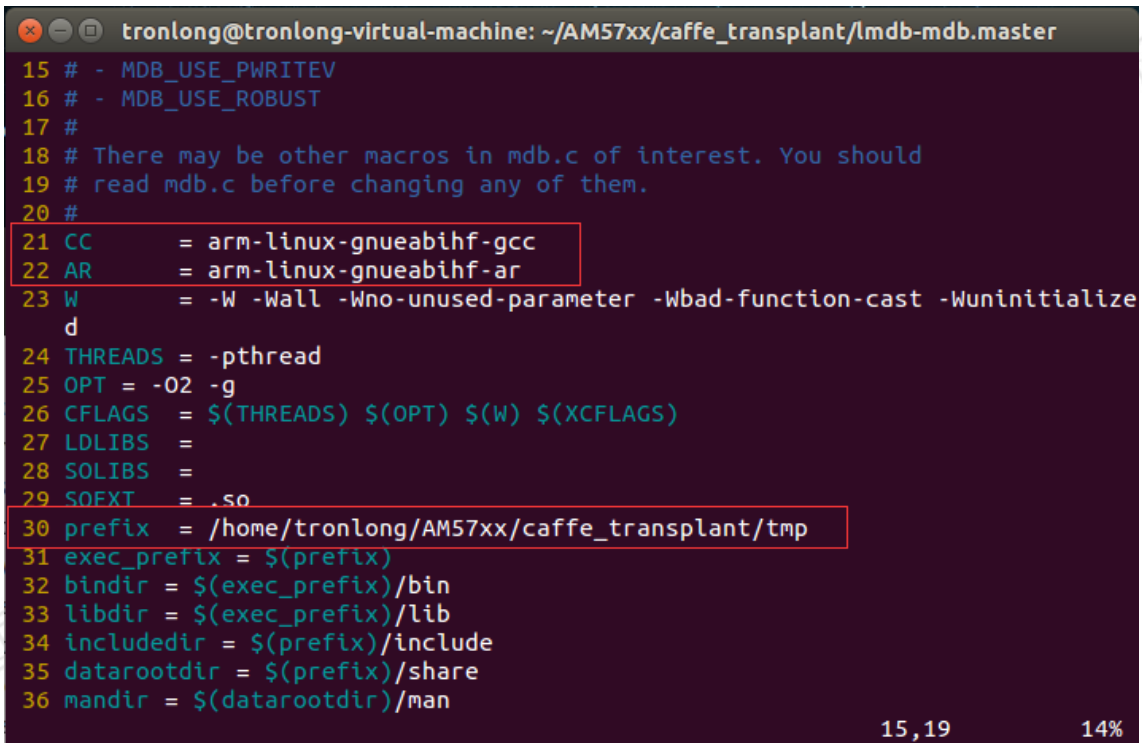

图 43

修改完成保存退出，执行以下指令，编译并安装 lmdb 依赖库到指定的“caffe_transplant/tmp”目录。

Host# make

Host# make install

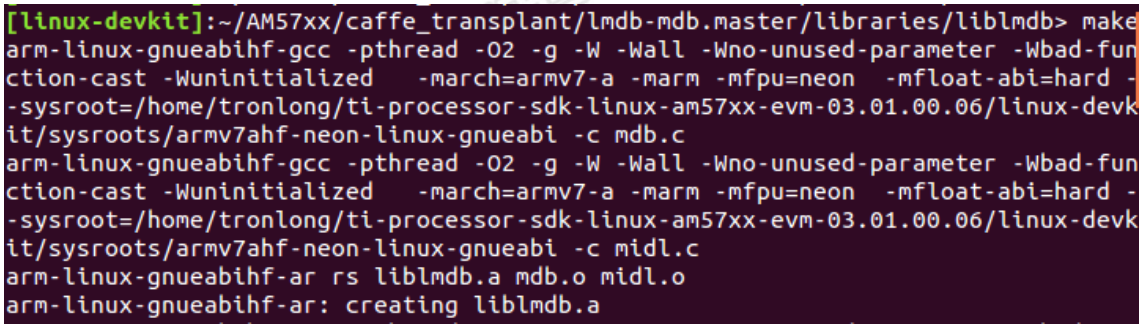

图 44

```
[linux-devkit]:~/AM57xx/caffe_transplant/lmdb-mdb.master/libraries/liblmdb> make
 install
mkdir -p /home/tronlong/AM57xx/caffe_transplant/tmp/bin
mkdir -p /home/tronlong/AM57xx/caffe_transplant/tmp/lib
mkdir -p /home/tronlong/AM57xx/caffe_transplant/tmp/include
mkdir -p /home/tronlong/AM57xx/caffe_transplant/tmp/share/man/man1
for f in mdb_stat mdb_copy mdb_dump mdb_load mdb_drop; do cp $f /home/tronlong/A
M57xx/caffe_transplant/tmp/bin; done
for f in liblmdb.a liblmdb.so; do cp $f /home/tronlong/AM57xx/caffe_transplant/t
mp/lib; done
for f in lmdb.h; do cp $f /home/tronlong/AM57xx/caffe_transplant/tmp/include; do
ne
for f in mdb_stat.1 mdb_copy.1 mdb_dump.1 mdb_load.1 mdb_drop.1; do cp $f /home/
tronlong/AM57xx/caffe_transplant/tmp/share/man/man1; done
[linux-devkit]:~/AM57xx/caffe_transplant/lmdb-mdb.master/libraries/liblmdb>
```

图 45

2.2.6 编译 gflag 依赖库

参照前面步骤，将光盘“Demo\app\caffe_transplant\src\gflags-master.zip”压缩文件拷贝到 Ubuntu，并解压到“/home/tronlong/AM57xx/caffe_transplant”目录，如下图：

Host# unzip gflags-master.zip

```
[linux-devkit]:~/AM57xx/caffe_transplant> pwd
/home/tronlong/AM57xx/caffe_transplant
[linux-devkit]:~/AM57xx/caffe_transplant> ls
cmake-3.11.0       google-snappy-4f7bd2d  lmdb-mdb.master  tmp
gflags-master.zip  leveldb-master         OpenBLAS-0.2.20
[linux-devkit]:~/AM57xx/caffe_transplant> unzip gflags-master.zip
Archive:  gflags-master.zip
e292e0452fcfd5a8ae055b59052fc041cbab4abf
   creating: gflags-master/
  inflating: gflags-master/.gitattributes
  inflating: gflags-master/.gitignore
  inflating: gflags-master/.gitmodules
```

图 46

请使用对应平台的 Linux Processor-SDK 加载环境变量。

进入 gflags-master 目录，执行以下指令，新建一个 build 文件夹。

Host# cd gflags-master/

Host# mkdir build

```
[linux-devkit]:~/AM57xx/caffe_transplant> cd gflags-master/
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master> mkdir build
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master> ls
appveyor.yml  build          cmake           doc          src
AUTHORS.txt   BUILD          CMakeLists.txt  INSTALL.md   test
bazel         ChangeLog.txt  COPYING.txt     README.md    WORKSPACE
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master>
```

图 47

进入 build 文件夹，执行以下指令，指定 gflag 依赖库安装目录为“/home/tronlong/AM57xx/caffe_transplant/tmp”。

Host# cd build/

Host# CXXFLAGS="-fPIC" cmake -DCMAKE_INSTALL_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..

```
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master> cd build/
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master/build> CXXFLAGS="-fPIC" c
make -DCMAKE_INSTALL_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..
-- The CXX compiler identification is GNU 5.3.1
-- Check for working CXX compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-
evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnuea
bihf-g++
-- Check for working CXX compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-
evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnuea
bihf-g++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
```

图 48

执行以下指令，编译并安装 gflag 依赖库到指定的“caffe_transplant/tmp”目录。

Host# make

Host# sudo make install

```
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master/build> make
Scanning dependencies of target gflags_static
[ 12%] Building CXX object CMakeFiles/gflags_static.dir/src/gflags.cc.o
[ 25%] Building CXX object CMakeFiles/gflags_static.dir/src/gflags_reporting.cc.
o
[ 37%] Building CXX object CMakeFiles/gflags_static.dir/src/gflags_completions.c
c.o
[ 50%] Linking CXX static library lib/libgflags.a
[ 50%] Built target gflags_static
Scanning dependencies of target gflags_nothreads_static
[ 62%] Building CXX object CMakeFiles/gflags_nothreads_static.dir/src/gflags.cc.
o
[ 75%] Building CXX object CMakeFiles/gflags_nothreads_static.dir/src/gflags_rep
orting.cc.o
[ 87%] Building CXX object CMakeFiles/gflags_nothreads_static.dir/src/gflags_com
pletions.cc.o
[100%] Linking CXX static library lib/libgflags_nothreads.a
[100%] Built target gflags_nothreads_static
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master/build>
```

图 49

```
[linux-devkit]:~/AM57xx/caffe_transplant/gflags-master/build> make install
[ 50%] Built target gflags_static
[100%] Built target gflags_nothreads_static
Install the project...
-- Install configuration: ""
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/libgflags.a
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/libgflags_nothread
s.a
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/gflags/gflags.
h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/gflags/gflags_
declare.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/gflags/gflags_
completions.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/gflags/gflags_
gflags.h
```

图 50

2.2.7 编译 glog 依赖库

参照前面步骤，将光盘“Demo\app\caffe_transplant\src\glog-master.zip”压缩文件拷贝到 Ubuntu，并解压到“/home/tronlong/AM57xx/caffe_transplant”目录，如下图：

Host# unzip glog-master.zip

```
[linux-devkit]:~/AM57xx/caffe_transplant> pwd
/home/tronlong/AM57xx/caffe_transplant
[linux-devkit]:~/AM57xx/caffe_transplant> ls
cmake-3.11.0   glog-master.zip      leveldb-master   OpenBLAS-0.2.20
gflags-master  google-snappy-4f7bd2d  lmdb-mdb.master  tmp
[linux-devkit]:~/AM57xx/caffe_transplant> unzip glog-master.zip
Archive:  glog-master.zip
2f493d292c92abf16ebd46cfd0cc0bf8eef5724d
   creating: glog-master/
  inflating: glog-master/.gitignore
  inflating: glog-master/AUTHORS
  inflating: glog-master/BUILD
```

图 51

请使用对应平台的 Linux Processor-SDK 加载环境变量。

进入 glog-master 目录，执行以下指令，新建一个 build 文件夹。

Host# cd glog-master/

Host# mkdir build

```
[linux-devkit]:~/AM57xx/caffe_transplant> pwd
/home/tronlong/AM57xx/caffe_transplant
[linux-devkit]:~/AM57xx/caffe_transplant> cd glog-master/
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master> mkdir build
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master> ls
AUTHORS      CMakeLists.txt        INSTALL         README
autogen.sh   configure.ac          libglog.pc.in   README.windows
bazel        CONTRIBUTING.md       ltmain.sh       src
build        CONTRIBUTORS          m4              WORKSPACE
BUILD        COPYING               Makefile.am
ChangeLog    doc                   NEWS
cmake        glog-config.cmake.in  packages
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master>
```

图 52

进入 build 目录，执行以下指令，将指定 glog 依赖库安装目录为“/home/tronlong/AM57xx/caffe_transplant/tmp”。

Host# cd build/

Host# cmake -DCMAKE_INSTALL_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..

```
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master> cd build/
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master/build> cmake -DCMAKE_INSTAL
L_PREFIX=/home/tronlong/AM57xx/caffe_transplant/tmp ..
-- The C compiler identification is GNU 5.3.1
-- The CXX compiler identification is GNU 5.3.1
-- Check for working C compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-ev
m-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabi
hf-gcc
-- Check for working C compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-ev
m-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabi
hf-gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-
evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnuea
bihf-g++
-- Check for working CXX compiler: /home/tronlong/ti-processor-sdk-linux-am57xx-
evm-03.01.00.06/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnuea
```

图 53

执行以下指令，编译并安装 glog 依赖库到指定的“caffe_transplant/tmp”文件夹。

Host# make

Host# make install

```
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master/build> make
Scanning dependencies of target glog
[  4%] Building CXX object CMakeFiles/glog.dir/src/demangle.cc.o
[  9%] Building CXX object CMakeFiles/glog.dir/src/logging.cc.o
[ 13%] Building CXX object CMakeFiles/glog.dir/src/raw_logging.cc.o
[ 18%] Building CXX object CMakeFiles/glog.dir/src/symbolize.cc.o
[ 22%] Building CXX object CMakeFiles/glog.dir/src/utilities.cc.o
[ 27%] Building CXX object CMakeFiles/glog.dir/src/vlog_is_on.cc.o
[ 31%] Building CXX object CMakeFiles/glog.dir/src/signalhandler.cc.o
[ 36%] Linking CXX static library libglog.a
[ 36%] Built target glog
Scanning dependencies of target signalhandler_unittest
```

图 54

```
[100%] Linking CXX executable utilities_unittest
[100%] Built target utilities_unittest
[linux-devkit]:~/AM57xx/caffe_transplant/glog-master/build> make install
[ 36%] Built target glog
[ 45%] Built target signalhandler_unittest
[ 54%] Built target stacktrace_unittest
[ 63%] Built target demangle_unittest
[ 72%] Built target symbolize_unittest
[ 81%] Built target logging_unittest
[ 90%] Built target stl_logging_unittest
[100%] Built target utilities_unittest
Install the project...
-- Install configuration: ""
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/lib/libglog.a
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/glog/logging.h
-- Installing: /home/tronlong/AM57xx/caffe_transplant/tmp/include/glog/raw_loggi
ng.h
```

图 55

将“caffe_transplant/tmp/bin”文件夹下的所有文件依次拷贝到开发板文件系统“/usr/lib”目录，“caffe_transplant/tmp/bin”目录下的文件是安装好的 Caffe 依赖库文件；将“caffe_transplant/tmp/include”文件夹下的所有文件依次拷贝到开发板文件系统“/usr/include”目录，该文件夹下包含的文件为 Caffe 所依赖的头文件。需要严格确认 tmp 文件夹下的文件已正确拷贝到开发板文件系统，否则会导致后面的编译出错。

```
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$ pwd
/home/tronlong/AM57xx/caffe_transplant/tmp
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$ ls
bin  include  lib  share
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$ ls lib/
cmake                  liblmdb.a                    libopenblas.so
libgflags.a            liblmdb.so                   libopenblas.so.0
libgflags_nothreads.a  libopenblas.a                libsnappy.a
libglog.a              libopenblas_armv7p-r0.2.20.a pkgconfig
libleveldb.a           libopenblas_armv7p-r0.2.20.so
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$ ls include/
cblas.h    glog     openblas_config.h  snappy-sinksource.h
f77blas.h  leveldb  snappy-c.h         snappy-stubs-public.h
gflags     lmdb.h   snappy.h
tronlong@tronlong-virtual-machine:~/AM57xx/caffe_transplant/tmp$
```

图 56

2.3 移植 Caffe 框架到开发板

将位于光盘“Demo\app\caffe_transplant\src”目录下的“hdf5-1.10.1.tar.gz”依赖库和“caffe-master.zip”例程压缩文件拷贝并开发板文件系统“/home/root”目录下。

```
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls
caffe-master.zip    hdf5-1.10.1.tar.gz
root@AM57xx-Tronlong:~#
```

图 57

开发板上电进入文件系统，依次执行如下指令将其解压到当前目录。

将开发板系统时钟设置为当前时间，在开发板上编译 Caffe 依赖库时，系统会检测开发板系统时钟，时间设置不当会导致编译报错。

Target# tar -xvf hdf5-1.10.1.tar.gz -C .

Target# unzip caffe-master.zip

Target# date -s "2018-07-10 11:32:30"

Target# date

```
root@AM57xx-Tronlong:~# tar -xvf hdf5-1.10.1.tar.gz -C .
root@AM57xx-Tronlong:~# unzip caffe-master.zip
root@AM57xx-Tronlong:~# ls
caffe-master        caffe-master.zip    hdf5-1.10.1         hdf5-1.10.1.tar.gz
root@AM57xx-Tronlong:~#
root@AM57xx-Tronlong:~# date -s "2018-07-10 11:32:30"
Tue Jul 10 11:32:30 UTC 2018
root@AM57xx-Tronlong:~# date
Tue Jul 10 11:32:32 UTC 2018
root@AM57xx-Tronlong:~#
```

图 58

2.3.1 编译 hdf5 依赖库

进入 hdf5-1.10.1 目录，执行以下指令，指定 hdf5 依赖库安装在“/usr”目录下。

Target# cd hdf5-1.10.1

Target# ./configure -prefix=/usr/

```
root@AM57xx-Tronlong:~# cd hdf5-1.10.1
root@AM57xx-Tronlong:~/hdf5-1.10.1# ./configure  -prefix=/usr/
checking for a BSD-compatible install... bin/install-sh -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... bin/install-sh -c -d
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether make supports nested variables... (cached) yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking build system type... armv7l-unknown-linux-gnueabihf
checking host system type... armv7l-unknown-linux-gnueabihf
checking shell variables initial values... done
checking if basename works... yes
checking if xargs works... yes
checking for cached host... none
```

图 59

```
Features:
---------
                  Parallel HDF5: no
              High-level library: yes
                    Threadsafety: no
             Default API mapping: v110
 With deprecated public symbols: yes
         I/O filters (external): deflate(zlib)
                            MPE: no
                     Direct VFD: no
                        dmalloc: no
 Packages w/ extra debug output: none
                    API tracing: no
           Using memory checker: no
Memory allocation sanity checks: no
             Metadata trace file: no
          Function stack tracing: no
      Strict file format checks: no
   Optimization instrumentation: no
root@AM57xx-Tronlong:~/hdf5-1.10.1#
```

图 60

执行以下指令，编译 hdf5 依赖库（耗时约 15 min），并将其安装到文件系统“/usr”目录下。

Target# make

Target# make install

```
root@AM57xx-Tronlong:~/hdf5-1.10.1# make
Making all in src
make[1]: Entering directory '/home/root/hdf5-1.10.1/src'
make  all-am
make[2]: Entering directory '/home/root/hdf5-1.10.1/src'
  CC       H5.lo
  CC       H5checksum.lo
  CC       H5dbg.lo
  CC       H5system.lo
  CC       H5timer.lo
  CC       H5trace.lo
  CC       H5A.lo
  CC       H5Abtree2.lo
  CC       H5Adense.lo
  CC       H5Adeprec.lo
  CC       H5Aint.lo
H5Aint.c: In function 'H5A_create':
H5Aint.c:200:45: warning: passing argument 1 of 'H5T_copy' discards 'const' qualifier from
 pointer target type [-Wdiscarded-qualifiers]
```

图 61

```
  CC       extend_dset-extend_dset.o
  CCLD     extend_dset
make[3]: Leaving directory '/home/root/hdf5-1.10.1/hl/tools/h5watch'
make[3]: Entering directory '/home/root/hdf5-1.10.1/hl/tools'
make[3]: Nothing to be done for 'all-am'.
make[3]: Leaving directory '/home/root/hdf5-1.10.1/hl/tools'
make[2]: Leaving directory '/home/root/hdf5-1.10.1/hl/tools'
make[2]: Entering directory '/home/root/hdf5-1.10.1/hl'
make[2]: Nothing to be done for 'all-am'.
make[2]: Leaving directory '/home/root/hdf5-1.10.1/hl'
make[1]: Leaving directory '/home/root/hdf5-1.10.1/hl'
root@AM57xx-Tronlong:~/hdf5-1.10.1# make install
Making install in src
make[1]: Entering directory '/home/root/hdf5-1.10.1/src'
make[2]: Entering directory '/home/root/hdf5-1.10.1/src'
 ../bin/install-sh -c -d '/usr/lib'
 /bin/sh ../libtool   --mode=install ../bin/install-sh -c   libhdf5.la '/usr/lib'
libtool: install: ../bin/install-sh -c .libs/libhdf5.so.101.0.0 /usr/lib/libhdf5.so.101.0.
0
libtool: install: (cd /usr/lib && { ln -s -f libhdf5.so.101.0.0 libhdf5.so.101 || { rm -f
libhdf5.so.101 && ln -s libhdf5.so.101.0.0 libhdf5.so.101; }; })
libtool: install: (cd /usr/lib && { ln -s -f libhdf5.so.101.0.0 libhdf5.so || { rm -f libh
df5.so && ln -s libhdf5.so.101.0.0 libhdf5.so; }; })
libtool: install: ../bin/install-sh -c .libs/libhdf5.lai /usr/lib/libhdf5.la
libtool: install: ../bin/install-sh -c .libs/libhdf5.a /usr/lib/libhdf5.a
libtool: install: chmod 644 /usr/lib/libhdf5.a
libtool: install: ranlib /usr/lib/libhdf5.a
libtool: finish: PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/sbin"
 ldconfig -n /usr/lib
----------------------------------------------------------------------
Libraries have been installed in:
```

图 62

```
+ ../../bin/install-sh -c ./ex_table_07.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./ex_table_08.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./ex_table_09.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./ex_table_10.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./ex_table_11.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./ex_table_12.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./ex_ds1.c /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./image24pixel.txt /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./image8.txt /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./pal_rgb.h /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c run-hlc-ex.sh /usr/share/hdf5_examples/hl/c/.
+ ../../bin/install-sh -c ./run-hl-ex.sh /usr/share/hdf5_examples/hl/.
make[2]: Leaving directory '/home/root/hdf5-1.10.1/hl/examples'
make[1]: Leaving directory '/home/root/hdf5-1.10.1/hl'
root@AM57xx-Tronlong:~/hdf5-1.10.1#
```

图 63

2.3.2 移植 caffe-master 到开发板

■ **修改 Makefile 文件**

进入 caffe-master 安装源码所在路径，打开 Makefile 文件，按照下图方法修改 Makefile 文件内容，修改完成保存退出。

Target# cd caffe-master

Target# vi Makefile

```
root@AM57xx-Tronlong:~# cd caffe-master
root@AM57xx-Tronlong:~/caffe-master# ls
CMakeLists.txt          README.md                  include
CONTRIBUTING.md         caffe.cloc                 matlab
CONTRIBUTORS.md         cmake                      models
INSTALL.md              data                       python
LICENSE                 docker                     scripts
Makefile                docs                       src
Makefile.config.example examples                   tools
root@AM57xx-Tronlong:~/caffe-master# vi Makefile
```

图 64

- BLAS ？ = open //将原来的“atlas”修改为“open”
- USE_PKG_CONFIG？=1 //将原来的“0”修改为“1”

```
# BLAS configuration (default = ATLAS)
BLAS ?= open
ifeq ($(BLAS), mkl)
        # MKL
        LIBRARIES += mkl_rt
        COMMON_FLAGS += -DUSE_MKL
        MKLROOT ?= /opt/intel/mkl
        BLAS_INCLUDE ?= $(MKLROOT)/include
        BLAS_LIB ?= $(MKLROOT)/lib $(MKLROOT)/lib/intel64
else ifeq ($(BLAS), open)
        # OpenBLAS
        LIBRARIES += openblas
else
        # ATLAS
        ifeq ($(LINUX), 1)
                ifeq ($(BLAS), atlas)
                        # Linux simply has cblas and atlas
                        LIBRARIES += cblas atlas
                endif
        else ifeq ($(OSX), 1)
                # OS X packages atlas as the vecLib framework
                LIBRARIES += cblas
                # 10.10 has accelerate while 10.9 has veclib
                XCODE_CLT_VER := $(shell pkgutil --pkg-info=com.apple.pkg.CLTools_Executab
                XCODE_CLT_GEQ_7 := $(shell [ $(XCODE_CLT_VER) -gt 6 ] && echo 1)
                XCODE_CLT_GEQ_6 := $(shell [ $(XCODE_CLT_VER) -gt 5 ] && echo 1)
                ifeq ($(XCODE_CLT_GEQ_7), 1)
                        BLAS_INCLUDE ?= /Applications/Xcode.app/Contents/Developer/Platfor
                else ifeq ($(XCODE_CLT_GEQ_6), 1)
                        BLAS_INCLUDE ?= /System/Library/Frameworks/Accelerate.framework/Ve
                        LDFLAGS += -framework Accelerate
                else
                        BLAS_INCLUDE ?= /System/Library/Frameworks/vecLib.framework/Versio
                        LDFLAGS += -framework vecLib
                endif
        endif
endif
INCLUDE_DIRS += $(BLAS_INCLUDE)
LIBRARY_DIRS += $(BLAS_LIB)

LIBRARY_DIRS += $(LIB_BUILD_DIR)

# Automatic dependency generation (nvcc is handled separately)
CXXFLAGS += -MMD -MP

# Complete build flags.
COMMON_FLAGS += $(foreach includedir,$(INCLUDE_DIRS),-I$(includedir))
CXXFLAGS += -pthread -fPIC $(COMMON_FLAGS) $(WARNINGS)
NVCCFLAGS += -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON_FLAGS)
# mex may invoke an older gcc that is too liberal with -Wuninitalized
MATLAB_CXXFLAGS := $(CXXFLAGS) -Wno-uninitialized
LINKFLAGS += -pthread -fPIC $(COMMON_FLAGS) $(WARNINGS)

USE_PKG_CONFIG ?= 1
ifeq ($(USE_PKG_CONFIG), 1)
        PKG_CONFIG := $(shell pkg-config opencv --libs)
else
        PKG_CONFIG :=
```

图 65

■ **修改 Makefile.config 文件**

执行以下指令，将重命名 Makefile.config.example 为 Makefile.config，打开 Makefile.config 文件，按照以下方法修改其内容：

Target# mv Makefile.config.example Makefile.config

Target# vi Makefile.config

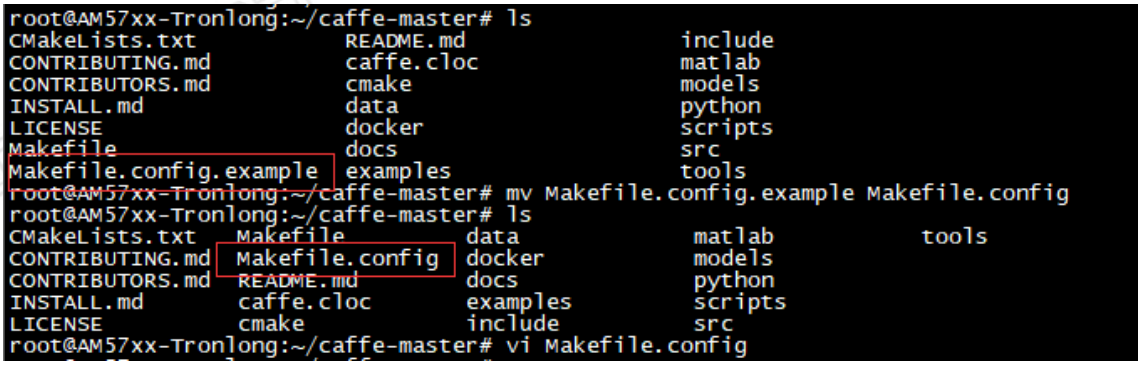

图 66

#CPU_ONLY := 1 //将指令前的“#”注释符去掉，使能该指令
#OPENCV_VERSION := 3 //将指令前的“#”注释符去掉，使能该指令

BLAS := open //将原来的“atlas”修改为“open”

Homebrew puts openblas in a directory that is not on the standard search path
BLAS_INCLUDE := /usr/include
BLAS_LIB := /usr/lib

Whatever else you find you need goes here.
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include /usr/include
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib /usr/lib

#USE_PKG_CONFIG := 1 //将指令前的“#”注释符去掉，使能该指令

```
# cuDNN acceleration switch (uncomment to build with cuDNN).
# USE_CUDNN := 1

# CPU-only switch (uncomment to build without GPU support).
CPU_ONLY := 1

# uncomment to disable IO dependencies and corresponding data layers
# USE_OPENCV := 0
# USE_LEVELDB := 0
# USE_LMDB := 0

# uncomment to allow MDB_NOLOCK when reading LMDB files (only if necessary)
#       You should not set this flag if you will be reading LMDBs with any
#       possibility of simultaneous read and write
# ALLOW_LMDB_NOLOCK := 1

# Uncomment if you're using OpenCV 3
OPENCV_VERSION := 3

# To customize your choice of compiler, uncomment and set the following.
# N.B. the default for Linux is g++ and the default for OSX is clang++
# CUSTOM_CXX := g++

# CUDA directory contains bin/ and lib/ directories that we need.
CUDA_DIR := /usr/local/cuda
# On Ubuntu 14.04, if cuda tools are installed via
# "sudo apt-get install nvidia-cuda-toolkit" then use this instead:
# CUDA_DIR := /usr

# CUDA architecture setting: going with all of them.
# For CUDA < 6.0, comment the *_50 through *_61 lines for compatibility.
# For CUDA < 8.0, comment the *_60 and *_61 lines for compatibility.
# For CUDA >= 9.0, comment the *_20 and *_21 lines for compatibility.
CUDA_ARCH := -gencode arch=compute_20,code=sm_20 \
             -gencode arch=compute_20,code=sm_21 \
             -gencode arch=compute_30,code=sm_30 \
             -gencode arch=compute_35,code=sm_35 \
             -gencode arch=compute_50,code=sm_50 \
             -gencode arch=compute_52,code=sm_52 \
             -gencode arch=compute_60,code=sm_60 \
             -gencode arch=compute_61,code=sm_61 \
             -gencode arch=compute_61,code=compute_61

# BLAS choice:
# atlas for ATLAS (default)
# mkl for MKL
# open for OpenBlas
BLAS := open
# Custom (MKL/ATLAS/OpenBLAS) include and lib directories.
# Leave commented to accept the defaults for your choice of BLAS
# (which should work)!
# BLAS_INCLUDE := /path/to/your/blas
# BLAS_LIB := /path/to/your/blas

# Homebrew puts openblas in a directory that is not on the standard search path
BLAS_INCLUDE := /usr/include
BLAS_LIB := /usr/lib

# This is required only if you will compile the matlab interface.
```

图 67

```
# Uncomment to support layers written in Python (will link against Python libs)
# WITH_PYTHON_LAYER := 1

# Whatever else you find you need goes here.
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include /usr/include
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib  /usr/lib

# If Homebrew is installed at a non standard location (for example your home directory) an
# INCLUDE_DIRS += $(shell brew --prefix)/include
# LIBRARY_DIRS += $(shell brew --prefix)/lib

# NCCL acceleration switch (uncomment to build with NCCL)
# https://github.com/NVIDIA/nccl (last tested version: v1.2.3-1+cuda8.0)
# USE_NCCL := 1

# Uncomment to use `pkg-config` to specify OpenCV library paths.
# (Usually not necessary -- OpenCV libraries are normally installed in one of the above $L
USE_PKG_CONFIG := 1

# N.B. both build and distribute dirs are cleared on `make clean`
BUILD_DIR := build
DISTRIBUTE_DIR := distribute

# Uncomment for debugging. Does not work on OSX due to https://github.com/BVLC/caffe/issue
# DEBUG := 1

# The ID of the GPU that 'make runtest' will use to run unit tests.
TEST_GPUID := 0
```

图 68

■ 修改 opencv.pc 文件

用 cat 指令查看“/usr/lib/pkgconfig/opencv.pc”文件内容，将对应位置的“-lopencv_dnn”代码删除。可通过 cat 指令查看 opencv.pc 文件是否已修改成功。

Target# cat /usr/lib/pkgconfig/opencv.pc

Target# vi /usr/lib/pkgconfig/opencv.pc

```
root@AM57xx-Tronlong:~/caffe-master# cat /usr/lib/pkgconfig/opencv.pc
# Package Information for pkg-config

prefix=/usr
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir_old=${prefix}/include/opencv
includedir_new=${prefix}/include

Name: OpenCV
Description: Open Source Computer Vision Library
Version: 3.1.0
Libs: -L${exec_prefix}/lib -L${exec_prefix}/share/OpenCV/3rdparty/lib -lopencv_stitching -
lopencv_superres -lopencv_videostab -lopencv_aruco -lopencv_bgsegm -lopencv_bioinspired -l
opencv_ccalib -lopencv_cvv -lopencv_dnn -lopencv_dpm -lopencv_fuzzy -lopencv_line_descript
or -lopencv_optflow -lopencv_plot -lopencv_reg -lopencv_saliency -lopencv_stereo -lopencv_
structured_light -lopencv_rgbd -lopencv_surface_matching -lopencv_tracking -lopencv_datase
ts -lopencv_text -lopencv_face -lopencv_xfeatures2d -lopencv_shape -lopencv_video -lopencv
_ximgproc -lopencv_calib3d -lopencv_features2d -lopencv_flann -lopencv_xobjdetect -lopencv
_objdetect -lopencv_ml -lopencv_xphoto -lopencv_highgui -lopencv_videoio -lopencv_imgcodec
s -lopencv_photo -lopencv_imgproc -lopencv_core
Libs.private: -L${exec_prefix}/lib -lOpenCL -lQt5Test -lQt5Concurrent -ljpeg -lwebp -lpng
-lz -ltiff -lgstvideo-1.0 -lgstapp-1.0 -lgstbase-1.0 -lgstriff-1.0 -lgstpbutils-1.0 -lgstr
eamer-1.0 -lgobject-2.0 -lglib-2.0 -lv4l1 -lv4l2 -lgphoto2 -lgphoto2_port -lexif -lQt5Core
 -lQt5Gui -lQt5Widgets -ldl -lm -lpthread -lrt -ltbb
Cflags: -I${includedir_old} -I${includedir_new}
root@AM57xx-Tronlong:~/caffe-master#
root@AM57xx-Tronlong:~/caffe-master#
root@AM57xx-Tronlong:~/caffe-master# vi /usr/lib/pkgconfig/opencv.pc
```

图 69

执行以下指令，编译 caffe 工程，耗时约 15 min。

Target# make

```
root@AM57xx-Tronlong:~/caffe-master# make
find: unrecognized: -or
BusyBox v1.24.1 (2016-10-03 18:14:31 EDT) multi-call binary.

Usage: find [-HL] [PATH]... [OPTIONS] [ACTIONS]
find: unrecognized: -or
BusyBox v1.24.1 (2016-10-03 18:14:31 EDT) multi-call binary.

Usage: find [-HL] [PATH]... [OPTIONS] [ACTIONS]
make: Warning: File '.build_release/src/caffe/proto/caffe.pb.o' has modification time 55696278 s
 in the future
CXX src/caffe/blob.cpp
CXX src/caffe/parallel.cpp
CXX src/caffe/data_transformer.cpp
CXX src/caffe/solvers/adadelta_solver.cpp
CXX src/caffe/solvers/adam_solver.cpp
```

图 70

```
CXX/LD -o .build_release/tools/caffe.bin
CXX tools/upgrade_net_proto_text.cpp
CXX/LD -o .build_release/tools/upgrade_net_proto_text.bin
CXX tools/upgrade_solver_proto_text.cpp
CXX/LD -o .build_release/tools/upgrade_solver_proto_text.bin
CXX tools/compute_image_mean.cpp
CXX/LD -o .build_release/tools/compute_image_mean.bin
CXX tools/convert_imageset.cpp
CXX/LD -o .build_release/tools/convert_imageset.bin
CXX examples/siamese/convert_mnist_siamese_data.cpp
CXX/LD -o .build_release/examples/siamese/convert_mnist_siamese_data.bin
CXX examples/mnist/convert_mnist_data.cpp
CXX/LD -o .build_release/examples/mnist/convert_mnist_data.bin
CXX examples/cifar10/convert_cifar_data.cpp
CXX/LD -o .build_release/examples/cifar10/convert_cifar_data.bin
CXX examples/cpp_classification/classification.cpp
CXX/LD -o .build_release/examples/cpp_classification/classification.bin
make: warning:  Clock skew detected.  Your build may be incomplete.
root@AM57xx-Tronlong:~/caffe-master#
```

图 71

执行以下指令，将编译生成的库文件拷贝到“/usr/lib”目录中。

Target# cd build/lib

Target# cp * /usr/lib

```
root@AM57xx-Tronlong:~/caffe-master# cd build/lib
root@AM57xx-Tronlong:~/caffe-master/build/lib# ls
libcaffe.a          libcaffe.so         libcaffe.so.1.0.0
root@AM57xx-Tronlong:~/caffe-master/build/lib# cp * /usr/lib
root@AM57xx-Tronlong:~/caffe-master/build/lib#
```

图 72

执行 reboot 指令重启开发板，进入 caffe-master 目录，执行以下指令，生成 caffe.pb.h 文件，并将其拷贝到新建的“include/caffe/proto”目录下。

Target# protoc src/caffe/proto/caffe.proto --cpp_out=.

Target# mkdir include/caffe/proto

Target# mv src/caffe/proto/caffe.pb.h include/caffe/proto/

```
root@AM57xx-Tronlong:~# cd caffe-master
root@AM57xx-Tronlong:~/caffe-master# protoc  src/caffe/proto/caffe.proto  --cpp_out=.
root@AM57xx-Tronlong:~/caffe-master# mkdir include/caffe/proto
/oot@AM57xx-Tronlong:~/caffe-master# mv  src/caffe/proto/caffe.pb.h include/caffe/proto/
root@AM57xx-Tronlong:~/caffe-master#
```

图 73

2.3.3 Caffe 框架测试

在 caffe-master 目录下执行以下指令，打开“examples/mnist/lenet_solver.prototxt”文件，按照下图方法修改文件内容，然后开启文件权限。

Target# vi examples/mnist/lenet_solver.prototxt

Target# chmod -R 777 *

```
root@AM57xx-Tronlong:~/caffe-master# vi examples/mnist/lenet_solver.prototxt
root@AM57xx-Tronlong:~/caffe-master# chmod -R 777 *
root@AM57xx-Tronlong:~/caffe-master#
```

图 74

修改如下内容：solver_mode:GPU //修改为 solver_mode:CPU

```
# The train/test net protocol buffer definition
net: "examples/mnist/lenet_train_test.prototxt"
# test_iter specifies how many forward passes the test should carry out.
# In the case of MNIST, we have test batch size 100 and 100 test iterations,
# covering the full 10,000 testing images.
test_iter: 100
# Carry out testing every 500 training iterations.
test_interval: 500
# The base learning rate, momentum and the weight decay of the network.
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
# The learning rate policy
lr_policy: "inv"
gamma: 0.0001
power: 0.75
# Display every 100 iterations
display: 100
# The maximum number of iterations
max_iter: 10000
# snapshot intermediate results
snapshot: 5000
snapshot_prefix: "examples/mnist/lenet"
# solver mode: CPU or GPU
solver_mode: CPU
```

图 75

将网线接入开发板，确认开发板网络连通正常。在 caffe-master 目录下执行以下指令，下载训练标本，运行官方测试脚本，测试 Caffe 框架是否移植成功。

Target# ./data/mnist/get_mnist.sh

Target# ./examples/mnist/create_mnist.sh

Target# ./examples/mnist/train_lenet.sh //本指令测试耗时约 40min

```
root@AM57xx-Tronlong:~/caffe-master# ./data/mnist/get_mnist.sh
Downloading...
Connecting to yann.lecun.com (216.165.22.6:80)
train-images-idx3-ub 100% |*******************************|  9680k  0:00:00 ETA
Connecting to yann.lecun.com (216.165.22.6:80)
train-labels-idx1-ub 100% |*******************************| 28881   0:00:00 ETA
Connecting to yann.lecun.com (216.165.22.6:80)
t10k-images-idx3-uby 100% |*******************************|  1610k  0:00:00 ETA
Connecting to yann.lecun.com (216.165.22.6:80)
t10k-labels-idx1-uby 100% |*******************************|  4542   0:00:00 ETA
root@AM57xx-Tronlong:~/caffe-master#
```

图 76

```
root@AM57xx-Tronlong:~/caffe-master# ./examples/mnist/create_mnist.sh
Creating lmdb...
I1003 22:40:03.950618  1187 db_lmdb.cpp:35] Opened lmdb examples/mnist/mnist_train_lmdb
I1003 22:40:03.953341  1187 convert_mnist_data.cpp:88] A total of 60000 items.
I1003 22:40:03.953382  1187 convert_mnist_data.cpp:89] Rows: 28 Cols: 28
I1003 22:40:46.864773  1187 convert_mnist_data.cpp:108] Processed 60000 files.
I1003 22:40:47.124292  1189 db_lmdb.cpp:35] Opened lmdb examples/mnist/mnist_test_lmdb
I1003 22:40:47.124959  1189 convert_mnist_data.cpp:88] A total of 10000 items.
I1003 22:40:47.125001  1189 convert_mnist_data.cpp:89] Rows: 28 Cols: 28
I1003 22:40:54.709431  1189 convert_mnist_data.cpp:108] Processed 10000 files.
Done.
root@AM57xx-Tronlong:~/caffe-master#
```

图 77

```
root@AM57xx-Tronlong:~/caffe-master# ./examples/mnist/train_lenet.sh
I1003 22:41:39.201617  1192 caffe.cpp:197] Use CPU.
I1003 22:41:39.202481  1192 solver.cpp:45] Initializing solver from parameters:
test_iter: 100
test_interval: 500
base_lr: 0.01
display: 100
max_iter: 10000
lr_policy: "inv"
gamma: 0.0001
power: 0.75
momentum: 0.9
weight_decay: 0.0005
snapshot: 5000
snapshot_prefix: "examples/mnist/lenet"
solver_mode: CPU
net: "examples/mnist/lenet_train_test.prototxt"
train_state {
  level: 0
  stage: ""
}
I1003 22:41:39.203186  1192 solver.cpp:102] Creating training net from net file: examples
/mnist/lenet_train_test.prototxt
I1003 22:41:39.204963  1192 net.cpp:294] The NetState phase (0) differed from the phase (
1) specified by a rule in layer mnist
I1003 22:41:39.205014  1192 net.cpp:294] The NetState phase (0) differed from the phase (
1) specified by a rule in layer accuracy
I1003 22:41:39.205063  1192 net.cpp:51] Initializing net from parameters:
name: "LeNet"
state {
  phase: TRAIN
  level: 0
  stage: ""
```

图 78

```
I1003 21:57:36.917675  1146 sgd_solver.cpp:112] Iteration 9900, lr = 0.00596843
I1003 21:58:01.316712  1146 solver.cpp:468] Snapshotting to binary proto file examples/mn
ist/lenet_iter_10000.caffemodel
I1003 21:58:01.344424  1146 sgd_solver.cpp:280] Snapshotting solver state to binary proto
 file examples/mnist/lenet_iter_10000.solverstate
I1003 21:58:01.454618  1146 solver.cpp:331] Iteration 10000, loss = 0.00556045
I1003 21:58:01.454717  1146 solver.cpp:351] Iteration 10000, Testing net (#0)
I1003 21:58:15.474603  1149 data_layer.cpp:73] Restarting data prefetching from start.
I1003 21:58:16.054745  1146 solver.cpp:418]     Test net output #0: accuracy = 0.9904
I1003 21:58:16.054860  1146 solver.cpp:418]     Test net output #1: loss = 0.0321676 (* 1
 = 0.0321676 loss)
I1003 21:58:16.054891  1146 solver.cpp:336] Optimization Done.
I1003 21:58:16.054913  1146 caffe.cpp:250] Optimization Done.
root@AM57xx-Tronlong:~/caffe-master#
```

图 79

Target# ./build/tools/caffe.bin test -model=examples/mnist/lenet_train_test.prototxt -weights=examples/mnist/lenet_iter_10000.caffemodel

```
root@AM57xx-Tronlong:~/caffe-master# ./build/tools/caffe.bin test -model=examples/mnist
enet_train_test.prototxt -weights=examples/mnist/lenet_iter_10000.caffemodel
I1003 22:02:24.879449  1158 caffe.cpp:275] Use CPU.
I1003 22:02:24.886248  1158 net.cpp:294] The NetState phase (1) differed from the phase
 (0) specified by a rule in layer mnist
I1003 22:02:24.886354  1158 net.cpp:51] Initializing net from parameters:
name: "LeNet"
state {
  phase: TEST
  level: 0
  stage: ""
}
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TEST
  }
  transform_param {
```

图 80

```
I1003 23:29:32.252969  1197 caffe.cpp:304] Batch 40, accuracy = 0.99
I1003 23:29:32.253067  1197 caffe.cpp:304] Batch 40, loss = 0.0319143
I1003 23:29:32.397748  1197 caffe.cpp:304] Batch 41, accuracy = 0.99
I1003 23:29:32.397846  1197 caffe.cpp:304] Batch 41, loss = 0.060275
I1003 23:29:32.541981  1197 caffe.cpp:304] Batch 42, accuracy = 0.99
I1003 23:29:32.542078  1197 caffe.cpp:304] Batch 42, loss = 0.039674
I1003 23:29:32.687121  1197 caffe.cpp:304] Batch 43, accuracy = 1
I1003 23:29:32.687219  1197 caffe.cpp:304] Batch 43, loss = 0.0128712
I1003 23:29:32.833200  1197 caffe.cpp:304] Batch 44, accuracy = 0.99
I1003 23:29:32.833295  1197 caffe.cpp:304] Batch 44, loss = 0.0371363
I1003 23:29:32.985296  1197 caffe.cpp:304] Batch 45, accuracy = 0.98
I1003 23:29:32.985393  1197 caffe.cpp:304] Batch 45, loss = 0.0333522
I1003 23:29:33.130074  1197 caffe.cpp:304] Batch 46, accuracy = 0.99
I1003 23:29:33.130167  1197 caffe.cpp:304] Batch 46, loss = 0.0131839
I1003 23:29:33.282572  1197 caffe.cpp:304] Batch 47, accuracy = 1
I1003 23:29:33.282671  1197 caffe.cpp:304] Batch 47, loss = 0.00922051
I1003 23:29:33.427114  1197 caffe.cpp:304] Batch 48, accuracy = 0.96
I1003 23:29:33.427207  1197 caffe.cpp:304] Batch 48, loss = 0.0651368
I1003 23:29:33.571867  1197 caffe.cpp:304] Batch 49, accuracy = 0.98
I1003 23:29:33.571964  1197 caffe.cpp:304] Batch 49, loss = 0.0250327
I1003 23:29:33.571993  1197 caffe.cpp:309] Loss: 0.0420002
I1003 23:29:33.572088  1197 caffe.cpp:321] accuracy = 0.9852
I1003 23:29:33.572165  1197 caffe.cpp:321] loss = 0.0420002 (* 1 = 0.0420002 loss)
root@AM57xx-Tronlong:~/caffe-master#
```

图 81

3 基于 Caffe 框架的人脸检测案例

表 3

开发板型号	是否支持本实验
TL5728-EasyEVM	支持
TL5728-IDK	支持
TL5728F-EVM	支持

本实验基于 Caffe 框架，实现人脸检测案例，进行本实验测试前，请参照前面步骤移植 Caffe 框架到开发板。

由于例程源码中的 tmp.jpg 试验文件分辨率较大，使用开发板配套的 4.3 寸或者 7 寸 LCD 显示屏会导致图片显示不全，影响显示效果，因此使用 HDMI 显示屏配合本次实验效果更佳。

3.1 编译 face_detection

将光盘“Demo\app\caffe_transplant\caf_face_detection.zip”例程压缩源文件拷贝到开发板文件系统“/home/root”目录。执行以下指令，解压“caf_face_detection.zip”文件到当前目录。

Target# unzip caf_face_detection.zip

```
root@AM57xx-Tronlong:~# pwd
/home/root
root@AM57xx-Tronlong:~# ls
caf_face_detection.zip  caffe-master.zip        hdf5-1.10.1.tar.gz
caffe-master            hdf5-1.10.1
root@AM57xx-Tronlong:~# unzip caf_face_detection.zip
Archive:  caf_face_detection.zip
   creating: caf_face_detection/
   creating: caf_face_detection/caf_face_detection/
   creating: caf_face_detection/caf_face_detection/Eigen/
  inflating: caf_face_detection/caf_face_detection/Eigen/Cholesky
  inflating: caf_face_detection/caf_face_detection/Eigen/CholmodSupport
  inflating: caf_face_detection/caf_face_detection/Eigen/CMakeLists.txt
  inflating: caf_face_detection/caf_face_detection/Eigen/Core
  inflating: caf_face_detection/caf_face_detection/Eigen/Dense
  inflating: caf_face_detection/caf_face_detection/Eigen/Eigen
  inflating: caf_face_detection/caf_face_detection/Eigen/Eigenvalues
  inflating: caf_face_detection/caf_face_detection/Eigen/Geometry
  inflating: caf_face_detection/caf_face_detection/Eigen/Householder
  inflating: caf_face_detection/caf_face_detection/Eigen/IterativeLinearSolvers
  inflating: caf_face_detection/caf_face_detection/Eigen/Jacobi
```

图 82

```
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/face_detection.cpp
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/face_detection.hpp
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/face_full_conv.caff
emodel
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/face_full_conv.prot
otxt
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/face_full_conv2.pro
totxt
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/main.cpp
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/Makefile
  inflating: caf_face_detection/caf_face_detection/integrated_caffe/tmp.jpg
root@AM57xx-Tronlong:~# ls
caf_face_detection      caffe-master            hdf5-1.10.1
caf_face_detection.zip  caffe-master.zip        hdf5-1.10.1.tar.gz
root@AM57xx-Tronlong:~#
```

图 83

执行以下指令，进入 caf_face_detection 源码目录，修改 Makefile 文件，如下图所示：

Target# cd caf_face_detection/caf_face_detection/integrated_caffe/

Target# vi Makefile

```
root@AM57xx-Tronlong:~# cd caf_face_detection/caf_face_detection/integrated_caffe/
root@AM57xx-Tronlong:~/caf_face_detection/caf_face_detection/integrated_caffe# ls
Makefile                  face_detection.hpp         face_full_conv2.prototxt
data_transformer.hpp      face_full_conv.caffemodel  main.cpp
face_detection.cpp        face_full_conv.prototxt    tmp.jpg
e ot@AM57xx-Tronlong:~/caf_face_detection/caf_face_detection/integrated_caffe# vi Makefile
```

图 84

在打开的 Makefile 文件中，根据 Caffe 框架的实际移植安装目录，按照如下方法修改 Makefile 文件相关参数路径。

- LIBS := -L/usr/lib/ -L**/home/root/caffe-master/build/lib**
- INCLUDE := -I. -I**/home/root/caffe-master/include**

```
root@AM57xx-Tronlong:~/caffe-master/build/lib# pwd
/home/root/caffe-master/build/lib
root@AM57xx-Tronlong:~/caffe-master/build/lib# ls
libcaffe.a          libcaffe.so         libcaffe.so.1.0.0
root@AM57xx-Tronlong:~/caffe-master/build/lib# cd ../../include/
root@AM57xx-Tronlong:~/caffe-master/include# pwd
/home/root/caffe-master/include
root@AM57xx-Tronlong:~/caffe-master/include# ls
caffe
root@AM57xx-Tronlong:~/caffe-master/include#
```

图 85

```
SOURCE  := $(wildcard *.c) $(wildcard *.cpp)
OBJS    := $(patsubst %.c,%.o,$(patsubst %.cpp,%.o,$(SOURCE)))

TARGET  := face_detection

#compile and lib parameter
CC      := g++
LIBS    := -L/usr/lib/ -L/home/root/caffe-master/build/lib -lcaffe -lopencv_highgui -lope
LDFLAGS :=
DEFINES :=
INCLUDE := -I. -I/home/root/caffe-master/include -I../
CFLAGS  := -g -Wall -O3 $(DEFINES) $(INCLUDE) -std=gnu++11
CXXFLAGS:= $(CFLAGS) -DHAVE_CONFIG_H

.PHONY : everything objs clean veryclean rebuild

everything : $(TARGET)

all : $(TARGET)

objs : $(OBJS)

rebuild: veryclean everything
```

图 86

执行 make 指令，编译完成后在当前目录生成 face_detection 可执行文件。

Target# make

```
root@AM57xx-Tronlong:~/caf_face_detection/caf_face_detection/integrated_caffe# make
g++ -g -Wall -O3  -I. -I/home/root/caffe-master/include -I../ -std=gnu++11 -DHAVE_CONFIG_
H   -c -o face_detection.o face_detection.cpp
face_detection.cpp: In function 'void nms_max(std::vector<std::pair<cv::Rect_<float>, dou
ble> >&, std::vector<std::pair<cv::Rect_<float>, double> >&, double)':
face_detection.cpp:32:23: warning: comparison between signed and unsigned integer express
ions [-Wsign-compare]
     for (int i = 0; i < bd.size(); i++)
                     ^
face_detection.cpp:35:18: warning: comparison between signed and unsigned integer express
ions [-Wsign-compare]
         for (; j < final_bd.size(); j++)
                  ^
face_detection.cpp:60:15: warning: comparison between signed and unsigned integer express
ions [-Wsign-compare]
         if (j == final_bd.size())
               ^
face_detection.cpp: In function 'void nms_average(std::vector<std::pair<cv::Rect_<float>,
 double> >&, std::vector<std::pair<cv::Rect_<float>, double> >&, double)':
face_detection.cpp:85:31: warning: comparison between signed and unsigned integer express
ions [-Wsign-compare]
             for (int j = 1; j < bd.size(); j++)
```

图 87

```
/home/root/caffe-master/include/caffe/blob.hpp:56:23: warning: comparison between signed
and unsigned integer expressions [-Wsign-compare]
     for (int i = 0; i < shape_.size(); ++i) {
                     ^
g++ -g -Wall -O3  -I. -I/home/root/caffe-master/include -I../ -std=gnu++11 -DHAVE_CONFIG_
H   -c -o main.o main.cpp
g++ -g -Wall -O3  -I. -I/home/root/caffe-master/include -I../ -std=gnu++11 -DHAVE_CONFIG_
H -o face_detection face_detection.o main.o  -L/usr/lib/ -L/home/root/caffe-master/build/
lib -lcaffe -lopencv_highgui -lopencv_imgcodecs -lopencv_imgproc -lopencv_core -lboost_sy
stem
root@AM57xx-Tronlong:~/caf_face_detection/caf_face_detection/integrated_caffe# ls
Makefile                  face_detection.hpp         face_full_conv2.prototxt
data_transformer.hpp      face_detection.o           main.cpp
face_detection            face_full_conv.caffemodel  main.o
face_detection.cpp        face_full_conv.prototxt    tmp.jpg
root@AM57xx-Tronlong:~/caf_face_detection/caf_face_detection/integrated_caffe#
```

图 88

3.2 人脸检测案例

确认将 HDMI 显示屏连接到开发板，执行如下指令运行 face_detection 可执行文件，face_detection 程序将检测该目录下的 tmp.jpg 文件，完成 tmp.jpg 文件的人脸检测。串口终端打印信息，HDMI 显示屏显示内容如下图所示：

Target# ./face_detection

```
root@AM57xx-Tronlong:~/caf_face_detection/caf_face_detection/integrated_caffe# ./face_det
ection
Image size: 800(Width) 480(Height)
Scaling: 1
[ 7904.258167] omap-iommu 41501000.mmu: 41501000.mmu: version 3.0
[ 7904.264111] omap-iommu 41502000.mmu: 41502000.mmu: version 3.0
[ 7904.278034] omap-iommu 40d01000.mmu: 40d01000.mmu: version 3.0
[ 7904.284348] omap-iommu 40d02000.mmu: 40d02000.mmu: version 3.0
WARNING: Logging before InitGoogleLogging() is written to STDERR
I1003 23:17:47.544889  1203 upgrade_proto.cpp:69] Attempting to upgrade input file specif
ied using deprecated input fields: face_full_conv2.prototxt
I1003 23:17:47.545132  1203 upgrade_proto.cpp:72] Successfully upgraded file specified us
ing deprecated input fields.
W1003 23:17:47.545156  1203 upgrade_proto.cpp:74] Note that future Caffe releases will on
ly support input layers and not input fields.
I1003 23:17:47.545518  1203 net.cpp:51] Initializing net from parameters:
name: "CaffeNetConv"
```

图 89

```
I1003 23:26:50.060597  1135 net.cpp:200] conv1 does not need backward computation.
I1003 23:26:50.060616  1135 net.cpp:200] input does not need backward computation.
I1003 23:26:50.060631  1135 net.cpp:242] This network produces output prob
I1003 23:26:50.416162  1135 net.cpp:255] Network initialization done.
I1003 23:27:11.991001  1135 upgrade_proto.cpp:69] Attempting to upgrade input file specif
ied using deprecated input fields: face_full_conv.caffemodel
I1003 23:27:11.991125  1135 upgrade_proto.cpp:72] Successfully upgraded file specified us
ing deprecated input fields.
W1003 23:27:11.991147  1135 upgrade_proto.cpp:74] Note that future Caffe releases will on
ly support input layers and not input fields.


face_full_conv2.prototxt has been updated and net has been initial!


W1003 23:27:12.414906  1135 net.hpp:41] DEPRECATED: ForwardPrefilled() will be removed in
 a future version. Use Forward().
init done
Using Wayland-EGL
wlpvr: PVR Services Initialised
```

图 90

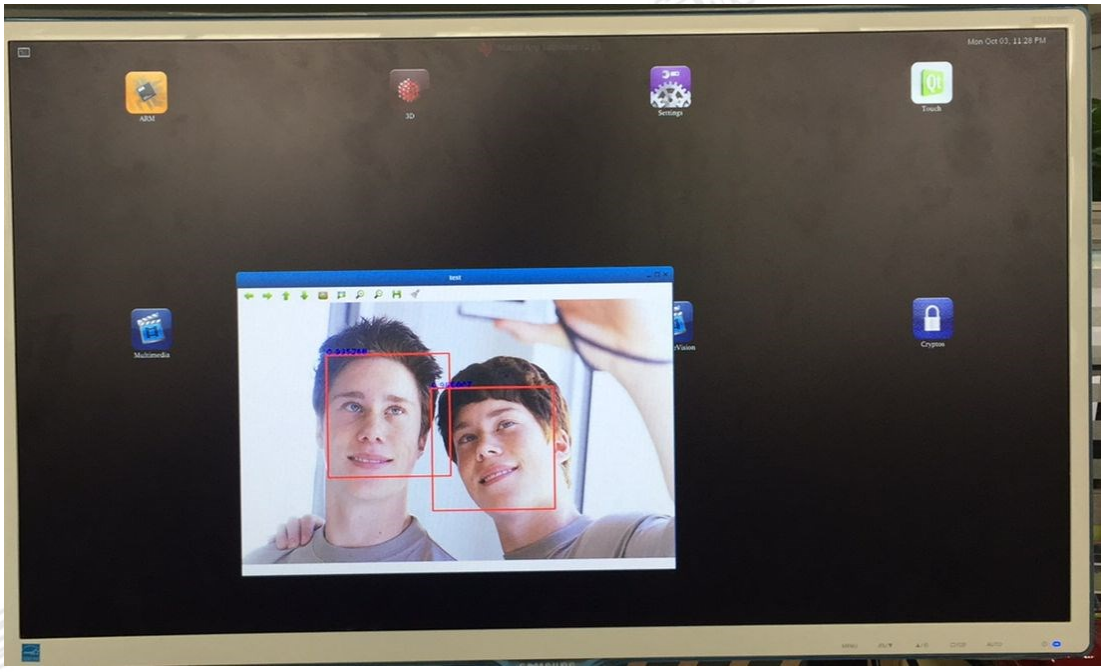

图 91

更多帮助

销售邮箱：sales@tronlong.com

技术邮箱：support@tronlong.com

创龙总机：020-8998-6280

技术热线：020-3893-9734

创龙官网：www.tronlong.com

技术论坛：www.51ele.net

线上商城：https://tronlong.taobao.com