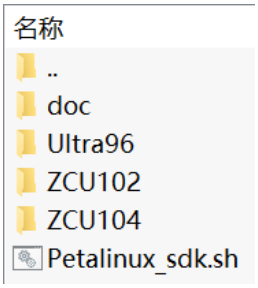
DNNSDK 的使用

一、环境安装

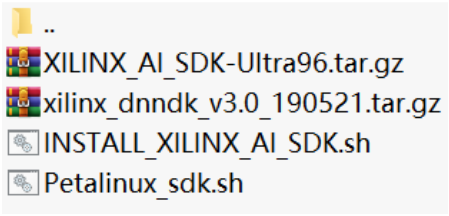
1.文件 xilinx-ultra96-prod-dpu1.4-desktop-buster-2019-05-31 解压后是一个镜像文件，用来给 ultral96 烧写 Linux 镜像。使用 Win32DiskImager 工具完成。

2.文件 xlnx_dnndk_v3.0_190624 下包含两个压缩包 XILINX_AI_SDK-V1.0.0-BUILD-16-2019-05-31.tar.gz 和 xilinx_dnndk_v3.0_190624.tar.gz

XILINX_AI_SDK-V1.0.0-BUILD-16-2019-05-31.tar.gz 文件解压后看到目录结构为：

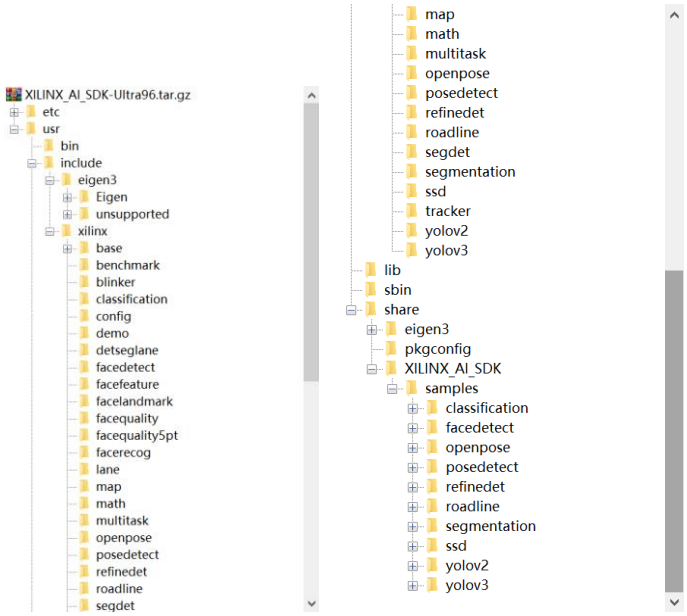


Petalinux_sdk.sh 用于给主机端安装 petalinx sdk 工具，Ultra96 下文件夹由目标执行



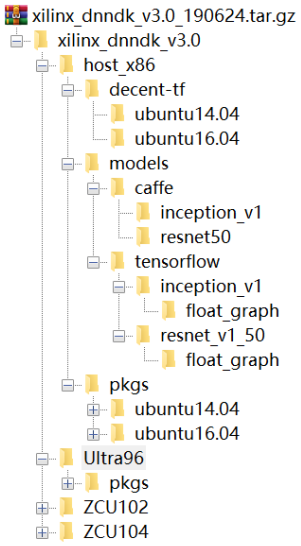
4.INSTALL_XILINX_AI_SDK.sh 主要完成两个功能

1.安装 xilinx_ai_sdk



xilinx_ai_sdk 安装后这个文件被解压放到对应目录,share/XILINX_AI_SDK 下有对应程序

2.xilinx_dnndk_v3.0_190624.tar.gz 文件打开后



可以看到是主机端的安装文件用于完成定点化、编译等功能，ultral96 文件夹下是对应的 DPU 的驱动

二、安装后可以看到 SDK 各个模块被放到对应的文件夹

Model and Library files are stored in `/usr/lib`

The header files are stored in `/usr/include/Xilinx`

Model profiles are stored in `/etc/XILINX_AI_SDK.conf.d`

Samples are stored in `/usr/share/XILINX_AI_SDK/samples`

Documents are stored in `/usr/share/XILINX_AI_SDK/doc`

Build.sh 用来生成目标文件

三、SDK 的使用

1. 使用YOLO完成目标检测，

1.新建文件yolo_test.cpp

2. 包含头文件.

```
#include <xilinx/yolov3/yolov3.hpp>//Necessary head file
#include <iostream>
#include <map>
#include <string>
//The necessary opencv lib
#include <opencv2/opencv.hpp>
```

3. 加载图片

```
Mat img = cv::imread(argv[1]);
```

4. 实例化模型

```
auto yolo = xilinx::yolov3::YOLOv3::create_ex("yolov3_voc_416", true);
```

5. 运行程序

```
auto results = yolo->run(img);
```

6 显示结果

```
for(auto &box : results.bboxes){
    int label = box.label;
    float xmin = box.x * img.cols + 1;
```

```
float ymin = box.y * img.rows + 1;
float xmax = xmin + box.width * img.cols;
float ymax = ymin + box.height * img.rows;
if(xmin < 0.) xmin = 1.;
if(ymin < 0.) ymin = 1.;
if(xmax > img.cols) xmax = img.cols;
if(ymax > img.rows) ymax = img.rows;
float confidence = box.score;
cout << "RESULT: " << label << "/t" << xmin << "/t" << ymin << "/t"<< xmax << "/t" << ymax << "/t" << confidence << "/n";
rectangle(img, Point(xmin, ymin), Point(xmax, ymax), Scalar(0, 255, 0),1, 1, 0);
}
```

7 .make产生可执行程序之后运行

2. 加入摄像头模块

1. 打开摄像头

```
VideoCapture cap(0);
```

2.将摄像头的画面输入网络中。

```
Mat frame;
Cap>>frame;
auto results = yolo->run(img);
```

3. 改用视频流

1.打开文件

```
VideoCapture cap(“test.avi”);
```

2.将视频的画面输入网络中。

```
Mat frame;
Cap>>frame;
auto results = yolo->run(img);
```