1. **源码编译paddle Lite2.8🡪获得最新预测库**
2. 环境准备

gcc、g++、git、make、wget、python、pip、python-dev、patchelf、cmake(>3.10)

sudo apt-get update

sudo apt-get install -y --no-install-recomends \

gcc g++ make wget python unzip patchelf **python3.7-dev**

# 2. install cmake 3.10 or above

wget <https://www.cmake.org/files/v3.10/cmake-3.10.3.tar.gz>

tar -zxvf cmake-3.10.3.tar.gz

cd cmake-3.10.3

./configure

make

sudo make install

1. 编译paddle Lite

git clone https://github.com/PaddlePaddle/Paddle-Lite.git

cd Paddle-Lite && **git checkout release/v2.8**

**# rm -rf third-party 删除此目录，编译脚本会自动从国内CDN下载第三方库文件**

# 编译Paddle-Lite Linux(arm)预测库 (armv8, gcc编译)

*export LITE\_BUILD\_THREADS=1 //单线程，可以设置为4线程*

./lite/tools/build\_linux.sh

1. 编译结果

位于：Paddle-Lite/build.lite.linux.armv8.gcc/inference\_lite\_lib.armlinux.armv8

inference\_lite\_lib.armlinux.armv8

├── cxx C++ 预测库和头文件

│   ├── **include**  C++ 头文件

│   │   ├── paddle\_api.h

│   │   ├── paddle\_image\_preprocess.h

│   │   ├── paddle\_lite\_factory\_helper.h

│   │   ├── paddle\_place.h

│   │   ├── paddle\_use\_kernels.h

│   │   ├── paddle\_use\_ops.h

│   │   └── paddle\_use\_passes.h

│   └── **lib**  C++ 预测库

│   ├── libpaddle\_api\_light\_bundled.a C++ 静态库

│   └── **libpaddle\_light\_api\_shared.so** **C++ 动态库**

├── demo

│   └── python Python 预测库demo

│

└── python Python 预测库(需要打开with\_python选项)

├── install

│   └── dist

│      └── **paddlelite-\*.whl** Python whl包

└── lib

└── lite.so Python 预测库

Python3安装paddle Lite库：pip3 install paddlelite-\*.wh

1. 编译选项说明

* 默认安装

# 默认配置是4线程编译，如果您的设备配置较低（树莓派3B等），可能遇到未知编译错误,

# 建议通过 ```export LITE\_BUILD\_THREADS=1``` 设置为单线程编译

./lite/tools/build\_linux.sh

**./lite/tools/build\_linux.sh --with\_python=ON --python\_version=2.7|3.7 –with\_cv=ON--opt\_model\_dir=YourOptimizedModelDir**

* 其他可选编译命令

--arch: (armv8|armv7|armv7hf) arm版本，默认为armv8

--toolchain: (gcc|clang) 编译器类型，默认为gcc

--with\_extra: (OFF|ON) 是否编译OCR/NLP模型相关kernel&OP，默认为OFF，只编译CV模型相关kernel&OP

--**with\_python: (OFF|ON)** 是否编译python预测库, 默认为 OFF

**--python\_version: (2.7|3.5|3.7) 编译whl的Python版本，默认为 None**

--with\_cv: (OFF|ON) 是否编译CV相关预处理库, 默认为 OFF

--with\_log: (OFF|ON) 是否输出日志信息, 默认为 ON

--with\_exception: (OFF|ON) 是否在错误发生时抛出异常，默认为 OFF

* **裁剪预测库方法**（只编译模型中的kernel&OP，降低预测库体积）

./lite/tools/build\_linux.sh --with\_strip=ON --opt\_model\_dir=YourOptimizedModelDir

# 编译选项说明

--with\_strip: (OFF|ON); 是否根据输入模型裁剪预测库，默认为OFF

--opt\_model\_dir: 输入模型的绝对路径，需要为opt转化之后的模型

1. 验证安装

安装之后进入python3环境，**import paddlelite**验证安装

教程网站：[【超详细】树莓派4B 安装Paddle-Lite 2.8.0](https://blog.csdn.net/weixin_40973138/article/details/114780090)

来源网站：[源码编译Paddle Lite](https://paddle-lite.readthedocs.io/zh/latest/source_compile/compile_linux.html) [Paddle Lite2.8部署之树莓派3B](https://www.smallsung.com/archives/12/)

1. [**部署Paddle Lite Demo**](https://github.com/PaddlePaddle/Paddle-Lite-Demo)
2. 环境准备

gcc g++ [**opencv**](#opencv)**3.X.X** cmake的安装（以下所有命令均在设备上操作）

$ sudo apt-get update

$ sudo apt-get install gcc g++ make wget unzip libopencv-dev pkg-config

$ wget https://www.cmake.org/files/v3.10/cmake-3.10.3.tar.gz

$ tar -zxvf cmake-3.10.3.tar.gz

$ cd cmake-3.10.3

$ ./configure

$ make

$ sudo make install

1. 安装Demo

$ git clone <https://github.com/PaddlePaddle/Paddle-Lite-Demo>

1. 模型和预测库下载

$ cd Paddle-Lite-Demo/PaddleLite-armlinux-demo

$ ./download\_models\_and\_libs.sh # **下载模型和预测库v2.8**

1. 图像分类Demo的编译与运行（以下所有命令均在设备上操作）

$ cd Paddle-Lite-Demo/PaddleLite-armlinux-demo/image\_classification\_demo

$ ./run.sh armv8 # RK3399

$ ./run.sh armv7hf # 树莓派3B

在终端打印预测结果和性能数据，同时在build目录中生成result.jpg。

1. 目标检测Demo的编译与运行（以下所有命令均在设备上操作）

$ cd Paddle-Lite-Demo/PaddleLite-armlinux-demo/object\_detection\_demo

$ ./run.sh armv8 # RK3399

$ ./run.sh armv7hf # 树莓派3B

在终端打印预测结果和性能数据，同时在build目录中生成result.jpg。

1. ARMLinux更新预测库 [paddle Lite编译产生的cxx文件](#cxx)
2. 替换头文件目录，将生成的cxx中的include目录替换Paddle-Lite-Demo/PaddleLite-armlinux-demo/Paddle-Lite/include目录；
3. 替换armv8动态库，将生成的cxx/libs中的libpaddle\_light\_api\_shared.so替换Paddle-Lite-Demo/PaddleLite-armlinux-demo/Paddle-Lite/libs/armv8/libpaddle\_light\_api\_shared.so；

**注意**：*paddle Lite是为paddle Lite服务的，区别在于，paddle Lite编译后生成的python whl可以给pip3安装paddlelite第三方库，****注意标明版本号 –python-version=3.7***

1. **源码编译**[**OpenCV**](https://opencv.org/)**4.5.2**

**[官方说明文档](https://docs.opencv.org/4.5.2/d6/d15/tutorial_building_tegra_cuda.html)**

1. 下载opencv opencv\_contrib

wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.2.zip

wget -O opencv\_contrib.zip https://github.com/opencv/opencv\_contrib/archive/4.5.2.zip

[OpenCV\_GitHub](https://github.com/opencv/opencv)

1. 安装依赖

sudo apt-get update

sudo apt-get upgrade

pip3 install numpy

sudo apt-get install build-essential cmake unzip pkg-config

sudo apt-get install libjpeg-dev libpng-dev libtiff-dev

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev

sudo apt-get install libgtk-3-dev

sudo apt-get install libatlas-base-dev gfortran

sudo apt-get install python3.7-dev

sudo apt-get install openexr libatlas-base-dev

sudo apt-get install libtbb2 libtbb-dev libdc1394-22-dev libopenexr-dev

sudo apt-get install libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev

1. 创建工作文件夹并编译

cd opencv

mkdir build

**cd build**

进入虚拟环境 **source ~/MyEnv/cv4.5.2/bin/activate**

**#输入时格式为 cmake -D flag -D flag …… 编译时间过长，不要用SSH远程操作**

/\*\* CMAKE\_BUILD\_TYPE是编译方式

\* CMAKE\_INSTALL\_PREFIX是安装目录

\* OPENCV\_EXTRA\_MODULES\_PATH是加载额外模块

\* INSTALL\_PYTHON\_EXAMPLES是安装官方python例程

\* BUILD\_EXAMPLES是编译例程（这两个可以不加，不加编译稍微快一点点）

\*\*/

cmake -D CMAKE\_BUILD\_TYPE=RELEASE

-D CMAKE\_INSTALL\_PREFIX=/usr/local

#-D WITH\_TBB=ON

#-D WITH\_V4L=ON

#-D WITH\_QT=ON

#-D WITH\_GTK=ON

#-D WITH\_OPENGL=ON

#-D WITH\_VTK=ON

#-D OPENCV\_GENERATE\_PKGCONFIG=YES

-D INSTALL\_C\_EXAMPLES=ON

#-D INSTALL\_PYTHON\_EXAMPLES=ON

#-D OPENCV\_GENERATE\_PKGCONFIG=ON

#-D OPENCV\_ENABLE\_NONFREE = ON #surf特征检测用到

-D OPENCV\_EXTRA\_MODULES\_PATH=

~/Work/OpenCV/opencv-4.5/opencv\_contrib-4.5.2/modules

-D BUILD\_EXAMPLES=ON ..

cmake **-D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D OPENCV\_EXTRA\_MODULES\_PATH=~/Work/OpenCV/opencv-4.5/opencv\_contrib-4.5.2/modules** -D ENABLE\_PRECOMPILED\_HEADERS=OFF -D OPENCV\_ENABLE\_NONFREE=ON -D ENABLE\_NEON=ON -D WITH\_TBB=ON -D WITH\_V4L=ON -D CMAKE\_SHARED\_LINKER\_FLAGS=-latomic -D BUILD\_opencv\_python3=ON **-D BUILD\_EXAMPLES=OFF ..**

1. 备份build

cd ..

cp -r build ./build\_temp

1. 扩大内存空间

sudo rapsi-config

选择7 Advanced Option🡪Expand System ……

sudo reboot

1. 增大虚拟内存(2048MB)

sudo nano /etc/dphys-swapfile

找到CONF\_SWAPSIZE=100，将100改为2048

sudo /etc/init.d/dphys-swapfile stop

sudo /etc/init.d/dphys-swapfile start

注意：安装完成之后，要将虚拟内存改为100MB

1. 安装

make -j4 #4个线程 视情况而定

sudo make install

opencv\_version  结果为4.5.2，安装完成

1. 虚拟环境路径配置

sudo ldconfig

1. 验证安装
2. C++
3. Python
4. 卸载

1、删掉opencv4.conf文件，

2、在编译的文件夹(我的是～/opencv4.5)下运行命令sudo make uninstall

3、删除build文件夹：cd .. && rm -rf build

4、删除源码目录：cd .. && rm -rf opencv4.5

5、删除系统安装的文件：

　　 sudo rm -r /usr/local/include/opencv2

　　 /usr/local/include/opencv

　　 /usr/include/opencv

　　 /usr/include/opencv2

　　 /usr/local/share/opencv

　　 /usr/local/share/OpenCV

　　 /usr/share/opencv

　　 /usr/share/OpenCV

　　 /usr/local/bin/opencv\*

　　 /usr/local/lib/libopencv

注意：由于paddleLite对应opencv版本为3.x.x，安装CV4之后，需要更改.cc文件

1. 添加头文件

#include <opencv2/imgproc/types\_c.h>

#include <fstream>

1. 更改参数名

CV\_CAP\_PROP\_FRAME\_WIDTH🡪cv::CAP\_PROP\_FRAME\_WIDTH,其他类似

cap.set(cv::CAP\_PROP\_FRAME\_WIDTH, WIDTH);

cap.set(cv::CAP\_PROP\_FRAME\_HEIGHT, HEIGHT);

1. Camera video号

-1🡪0/1/2 [一般情况下，CSI摄像头为0，USB摄像头为1。]

cv::VideoCapture cap(video);

参考网站: [ubuntu18.04安装opencv4.5.1](https://blog.csdn.net/z1026544682/article/details/114225729) [ubuntu20.04安装opencv4.5.1](https://blog.csdn.net/m0_46621859/article/details/112274740?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-1.control&dist_request_id=1332048.21037.16195116427115789&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-1.control)

[树莓派3和4B安装OpenCV教程](https://blog.csdn.net/zqxdsy/article/details/102673226?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_baidulandingword-0&spm=1001.2101.3001.4242) [树莓派4b Opencv安装](https://blog.csdn.net/cold_hl/article/details/106195325?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_title-0&spm=1001.2101.3001.4242)

[树莓派4B基于python3安装opencv4全教程](https://zhuanlan.zhihu.com/p/127959573)

案例测试: [口罩检测](https://aistudio.baidu.com/aistudio/projectdetail/315730) [安全帽检测](https://aistudio.baidu.com/aistudio/projectdetail/1059610)

API：[paddle Lite C++ API文档](https://paddle-lite.readthedocs.io/zh/latest/api_reference/cxx_api_doc.html)

1. Paddle Lite Demo的使用
2. 操作目录

Paddle-Lite-Demo/PaddleLite-armlinux-demo

├── object\_dection\_demo code部分 自己编写

│   ├── labels 标注信息文件夹

│   │   ├── pascalvoc\_label\_list 存放标注信息

│   ├── image 目标图片

│   ├── **model**  预测模型

│   │   ├── **model.nb** 源自paddleDection,由opt转化

│   ├──  **CMakeLists.txt**  编译文件,更改最后两行

│   ├── **object\_detection\_demo.cc**  识别程序(C++,此处调用opencv)

│   ├── **run.sh**  可执行脚本 修改#run下内容

│   └── build  run.sh执行后产生的编译文件夹

└── **PaddleLite** 可由[paddle Lite](#paddlebuild)更新模型和预测库,**version要与model.nb一致**

├── include

   │   ├── paddle\_api.h

   │   ├── paddle\_image\_preprocess.h

   │   ├── paddle\_lite\_factory\_helper.h

   │   ├── paddle\_place.h

   │   ├── paddle\_use\_kernels.h

   │   ├── paddle\_use\_ops.h

   │   └── paddle\_use\_passes.h

└── lib

   ├── libpaddle\_api\_light\_bundled.a C++ 静态库

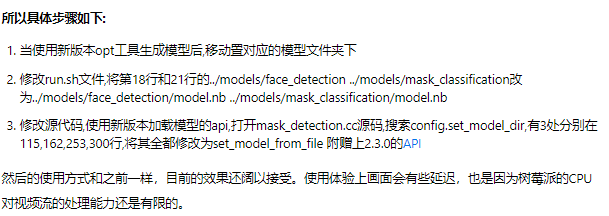
   └── libpaddle\_light\_api\_shared.so C++ 动态库

**注意**：model、PaddleLite由[download\_models\_and\_libs.sh](#download)产生

1. model.nb

在服务器/电脑上，使用paddleDection训练、评估、预测之后，获得权重，使用paddleDection对其进行压缩[剪裁、蒸馏、量化]，然后在x86的linux环境中用paddleLite下的opt转化成lite模型final\_model.nb

**注意opt转换版本为要与本地paddleLite版本号一致，即与PaddleLite文件下的模型和预测库相同;另外V2.3之后的版本,.cc都使用config.set\_model\_from\_file(model\_dir)，V2.3之前的版本都使用config.set\_model\_dir(model\_dir)**



1. CMakeLists.txt

add\_executable(**object\_detection\_demo** **object\_detection\_demo.cc**)

target\_link\_libraries(**object\_detection\_demo** paddle\_light\_api\_shared ${OpenCV\_LIBS})

paddle\_light\_api\_shared位于object\_dection\_demo/PaddleLite/lib,C++动态库

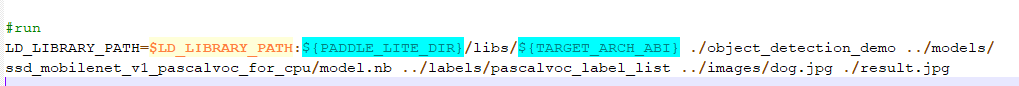
object\_detection\_demo更改为工作主目录

object\_detection\_demo.cc更改为自己编写的识别程序(C++)

1. object\_detection\_demo.cc

结合官方程序、自己项目数据集、模型进行改写

1. run.sh[工作目录object\_dection\_demo/]



更改cc名称、模型路径、标注路径、图片路径，如果调用摄像头，则删除图片路径

**.cc最好只更改内容，名称，路径不要变。Model和label可以更改文件名称及路径**

1. OpenCV调节/获取摄像头参数

VideoCapture capture(0); 设置摄像头参数 不要随意修改

capture.set(CV\_CAP\_PROP\_FRAME\_WIDTH, 1080);//宽度

capture.set(CV\_CAP\_PROP\_FRAME\_HEIGHT, 960);//高度

capture.set(CV\_CAP\_PROP\_FPS, 30);//帧率 帧/秒

capture.set(CV\_CAP\_PROP\_BRIGHTNESS, 1);//亮度

capture.set(CV\_CAP\_PROP\_CONTRAST,40);//对比度 40

capture.set(CV\_CAP\_PROP\_SATURATION, 50);//饱和度 50

capture.set(CV\_CAP\_PROP\_HUE, 50);//色调 50

capture.set(CV\_CAP\_PROP\_EXPOSURE, 50);//曝光 50 获取摄像头参数

capture.get(CV\_CAP\_PROP\_FRAME\_WIDTH);

capture.get(CV\_CAP\_PROP\_FRAME\_HEIGHT);

capture.get(CV\_CAP\_PROP\_FPS);

capture.get(CV\_CAP\_PROP\_BRIGHTNESS);

capture.get(CV\_CAP\_PROP\_CONTRAST);

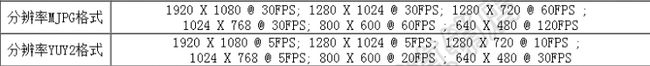
capture.get(CV\_CAP\_PROP\_SATURATION);

capture.get(CV\_CAP\_PROP\_HUE);

capture.get(CV\_CAP\_PROP\_EXPOSURE); 获取视频参数：

capture.get(CV\_CAP\_PROP\_FRAME\_COUNT);//视频帧数

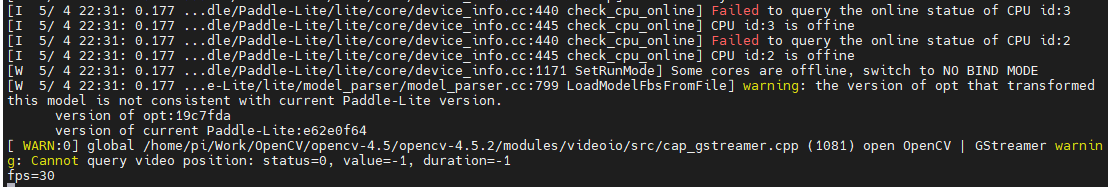
设置分辨率的前提是，摄像头要支持，否则无法调用摄像头[金贵支支持YUV]



cap.set(cv::CAP\_PROP\_FOURCC,cv::CAP\_OPENCV\_MJPEG)

1. [实例-口罩检测](https://aistudio.baidu.com/aistudio/projectdetail/315730)
2. 问题

(1)



opt版本不匹配，可使用paddleLiteV2.8下的opt对模型进行转化

**CPU offline**

export CPU\_NUM=4

const int **CPU\_THREAD\_NUM**= 2;

const paddle::lite\_api::PowerMode CPU\_POWER\_MODE =paddle::lite\_api::PowerMode::**LITE\_POWER\_HIGH**;

config.set\_threads(**CPU\_THREAD\_NUM**);

config.set\_power\_mode(**CPU\_POWER\_MODE**);

**Opencv|gstreamer:Cannot query video position**

(2)外部使用mjpg\_stream

打开网页提示not open file