密级状态：绝密(　　)　　秘密(　　)　　内部(　　)　　公开(√)

RKNN SDK 快速上手指南

（技术部，图形计算平台中心）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	1.5.2
	作　　者：	HPC
	完成日期：	2023-8-22
	审　　核：	熊伟
	完成日期：	2023-8-22

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd

更新记录

版本	修改人	修改日期	修改说明	核定人
1.2.0	许东晨	2022-1-14	初始版本	卓鸿添
1.3.0	HPC	2022-4-22	更新版本	卓鸿添
1.4.0	HPC	2022-8-22	更新版本	卓鸿添
1.4.2	HPC	2023-2-13	更新版本	卓鸿添
1.5.0	HPC	2023-5-22	更新版本	卓鸿添
1.5.2	HPC	2023-8-22	更新版本	卓鸿添

目 录

1 主要说明

此文档向零基础用户详细介绍如何快速在 ROCKCHIP 芯片的 EVB 板子上使用 RKNN-Toolkit2 和 RKNPU2 工具转换 yolov5s.onnx 模型为 yolov5s.rknn 模型并进行板端推理。

支持的平台：RK3566、RK3568、RK3588、RK3588S、RK3562。为了统一描述，后续说明使用 RK3566_RK3568 来表示 RK3566 或 RK3568 平台。

RKNPU2 工程下载地址：https://github.com/rockchip-linux/rknpu2

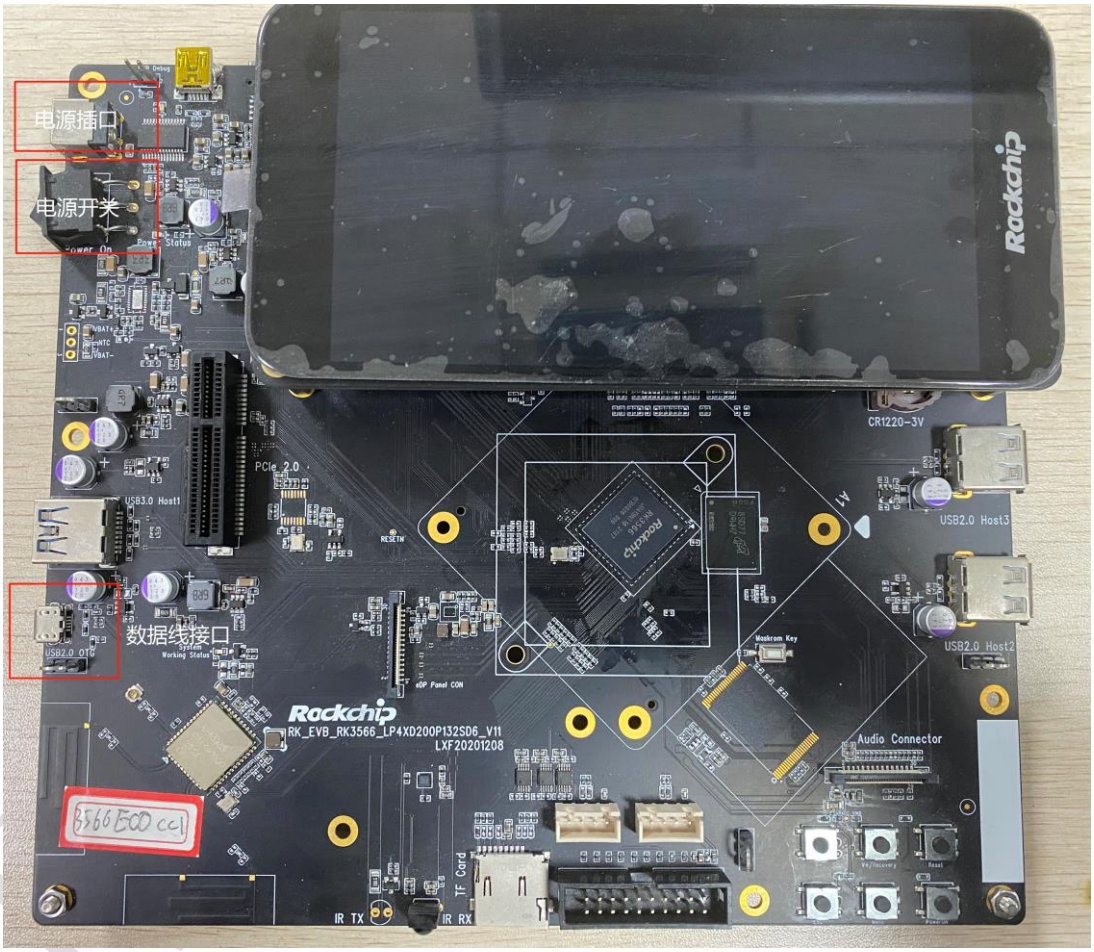
RKNN-Toolkit2 工程下载地址：https://github.com/rockchip-linux/rknn-toolkit2

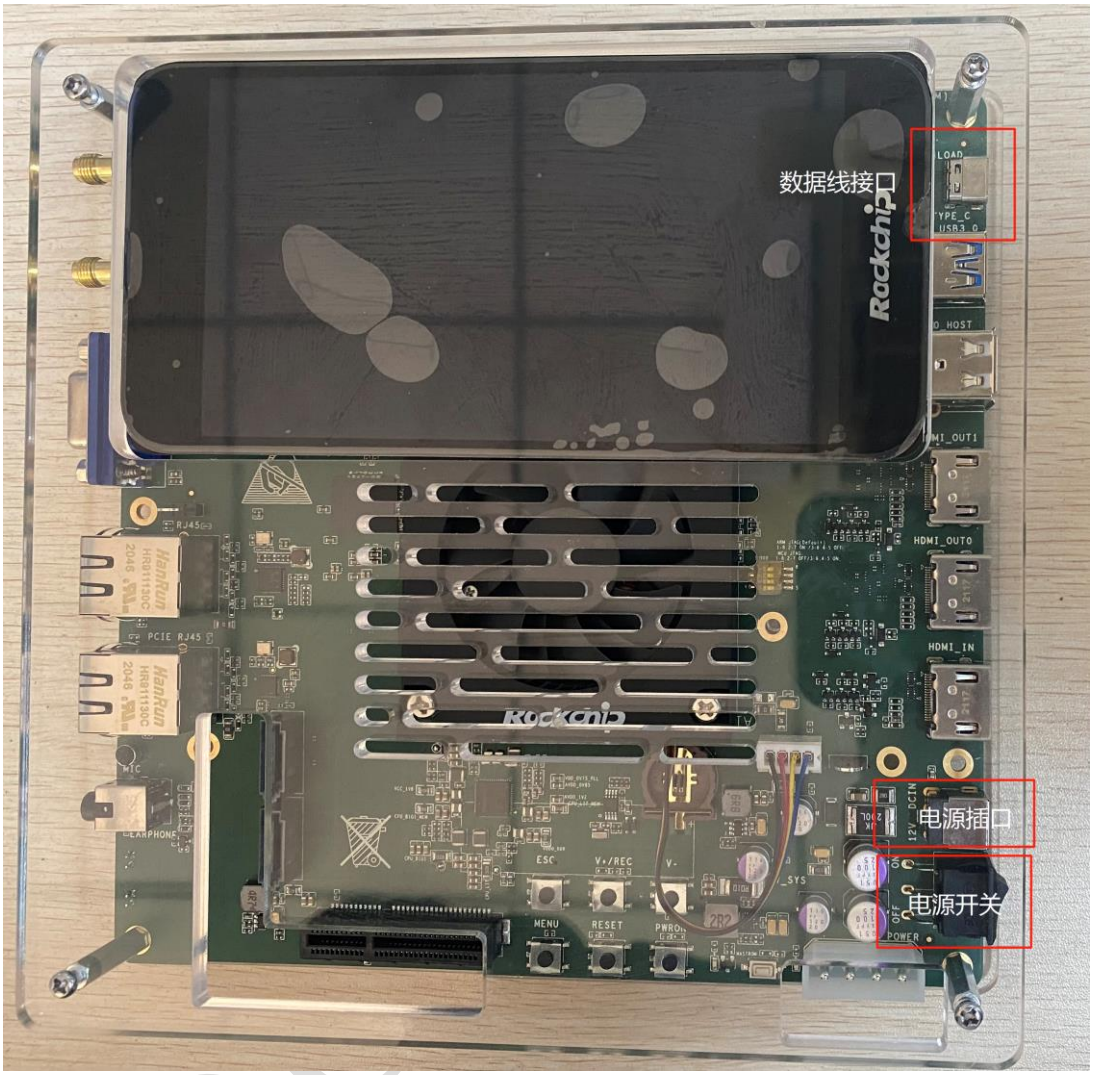
2 准备工具

1. 一台操作系统 Ubuntu18.04 / Ubuntu20.04 / Ubuntu22.04 的电脑。

2. 一块 EVB 板子（RK3566、RK3568、RK3588、RK3588S、RK3562）。

RK3566





RK3562

3. 一条连接板子和电脑的数据线。

RK3566_RK3568：USB-A——Micro USB

RK3588/RK3562：USB-A——USB-C

4. 一个电源适配器。

RK3562/RK3566_RK3568：输出 12V-2A

RK3588：输出 12V-3A

3 快速入门使用 RKNN-Toolkit2 和 RKNPU2

3.1 安装 RKNN-Toolkit2

本章节介绍两种安装使用 RKNN-Toolkit2 的方法，“通过 pip install 安装”和“通过 Docker 镜像安装”，用户可自行选择安装方式。如果不是 Ubuntu18.04 / Ubuntu20.04 / Ubuntu22.04 系统的电脑推荐使用“通过 Docker 镜像安装”方式，已集成所有所需的安装包依赖，简单可靠。

以下操作以 Ubuntu18.04 和 Python3.6 为例。

3.1.1 通过 Docker 镜像安装并推理

1. 电脑若没有安装 Docker 工具，请先按照此安装教程（https://mirrors.tuna.tsinghua.edu.cn/help/docker-ce/）安装 Docker 工具,再进行下一步。
2. 打开一个终端命令行窗口，cd 进入 RKNN-Toolkit2 工程的 docker 文件夹，根据工程的保存路径修改 cd 命令中的路径。

   cd <输入 RKNN-Toolkit2 工程其中 docker 文件夹的路径>

   命令：

   ```
   cd ~/Projects/rknn-toolkit2-1.x.x/docker/docker_full
   ls
   ```

   查看到当前目录下有一个 docker 镜像文件 rknn-toolkit2-1.x.x-cp36-docker.tar.gz。
3. 加载 docker 镜像。

   ```
   docker load --input rknn-toolkit2-1.x.x-cp36-docker.tar.gz
   ```

4. 查看当前所有的 docker 镜像。

   命令：

   ```
   docker images
   ```

   能查询到 REPOSITORY 为 rknn-toolkit2，TAG 为 1.x.x-cp36 则表示加载成功。

5. 运行 docker 容器。

命令：

```
docker run -t -i --privileged -v /dev/bus/usb:/dev/bus/usb \
-v ~/Projects/rknn-toolkit2-1.x.x/examples/onnx/yolov5:/rknn_yolov5_demo \
rknn-toolkit2:1.x.x-cp36 /bin/bash
```

将目录映射进 Docker 环境可通过附加“-v <host src folder>:<image dst folder>”。

绿色部分为 RKNN-Toolkit2 工程中 examples/onnx/yolov5 本地文件夹路径（根据本机路径修改）映射到 docker 容器中/rknn_yolov5_demo 文件夹。

成功进入 docker 容器后，命令 ls 能查看到文件夹 rknn_yolov5_demo 则表示映射成功。

6. 在 docker 容器中进入 rknn_yolov5_demo 目录。

命令：

```
cd rknn_yolov5_demo
```

7. 转换 yolov5s_relu.onnx 为 rknn 模型并行推理图片。

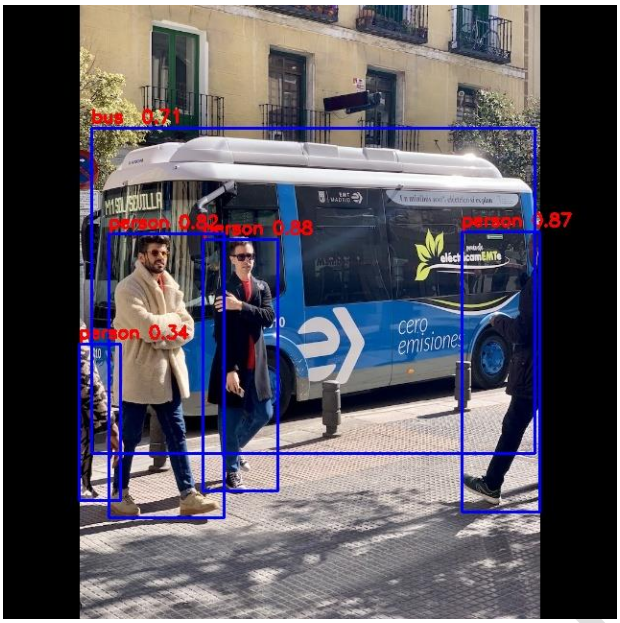
```
python3 ./test.py
```

```
I rknn building ...
I rknn buiding done.
done
--> Export rknn model
done
--> Init runtime environment
W init_runtime: Target is None, use simulator!
done
--> Running model
W inference: The 'data_format' has not been set and defaults is nhwc!
Analysing : 100%|| 153/153 [00:00<00:00, 10985.12it/s]
Preparing : 100%|| 153/153 [00:00<00:00, 414.30it/s]
W get_input_img: The dims of input(ndarray) shape (640, 640, 3) is wrong, expect dims is 4! Try expand dims to (1, 640, 640, 3)!
done
class: person, score: 0.884139358997345
box coordinate left,top,right,down: [208.8534364104271, 244.4304337501526, 286.3236876130104, 506.7466902732849]
class: person, score: 0.8676778078079224
box coordinate left,top,right,down: [478.82631254196167, 236.96079683303833, 559.7779355049133, 528.6053652763367]
class: person, score: 0.8246847987174988
box coordinate left,top,right,down: [110.32201385498047, 238.8315395116806, 230.29569244384766, 534.2514072656631]
class: person, score: 0.3394228219985962
box coordinate left,top,right,down: [79.96397459506989, 353.70939338207245, 122.13020265102386, 516.948687672615]
class: bus , score: 0.7050552368164062
box coordinate left,top,right,down: [92.08940839767456, 128.53247582912445, 554.9980549812317, 467.08300268650055]
```

此脚本是在 PC 模拟器上运行，若需要连板调试请参考章节 3.2。

转换模型和推理脚本 test.py 运行成功后，转换后的模型默认保存路径为 examples/onnx/yolov5/yolov5s_relu.rknn ，推理的图片结果保存在

examples/onnx/yolov5/result.jpg。



3.1.2 通过 pip install 安装并推理

1. 打开一个终端命令行窗口，安装 Python3.6 和 pip3。

   命令：

```
sudo apt-get install python3 python3-dev python3-pip
```

2. 安装所需的依赖包。

   命令：

```
sudo apt-get install libxslt1-dev zlib1g-dev libglib2.0 libsm6 \
libgl1-mesa-glx libprotobuf-dev gcc
```

3. 进入 Toolkit2 工程文件夹，根据工程的保存路径修改 cd 命令中的路径。

   cd <输入 Toolkit2 工程的路径>

   命令：

```
cd ~/rknn-toolkit2-1.x.x
```

4. 安装必要相应版本的依赖包。

命令：

```
pip3 install -r doc/requirements_cp36-1.x.x.txt
```

备注：

1) 若在安装过程中出现“匹配不到 XX 版本”的错误，可能由于是 pip 版本太低造成。可先执行以下升级 pip 版本命令，pip 升级至版本 21.3.1，再执行重新执行上述安装命令。

```
python3 -m pip install --upgrade pip
```

5. 安装 RKNN-Toolkit2（Python3.6 for x86_64）。

命令：

```
pip3 install \
package/rknn_toolkit2-1.x.x+xxxxxxxx-cp36-cp36m-linux_x86_64.whl
```

6. 检查 RKNN-Toolkit2 是否安装成功。

命令：

```
python3
from rknn.api import RKNN
```

```
Python 3.6.9 (default, Dec  8 2021, 21:08:43)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from rknn.api import RKNN
>>>
```

若没有出现错误，说明安装成功。同时按住 Ctrl+D 退出 Python3。

7. cd 进入 rknn-toolkit2-1.x.x/examples/onnx/yolov5 目录。

```
cd examples/onnx/yolov5
```

8. 转换 yolov5s_relu.onnx 为 rknn 模型并运行模型推理图片。

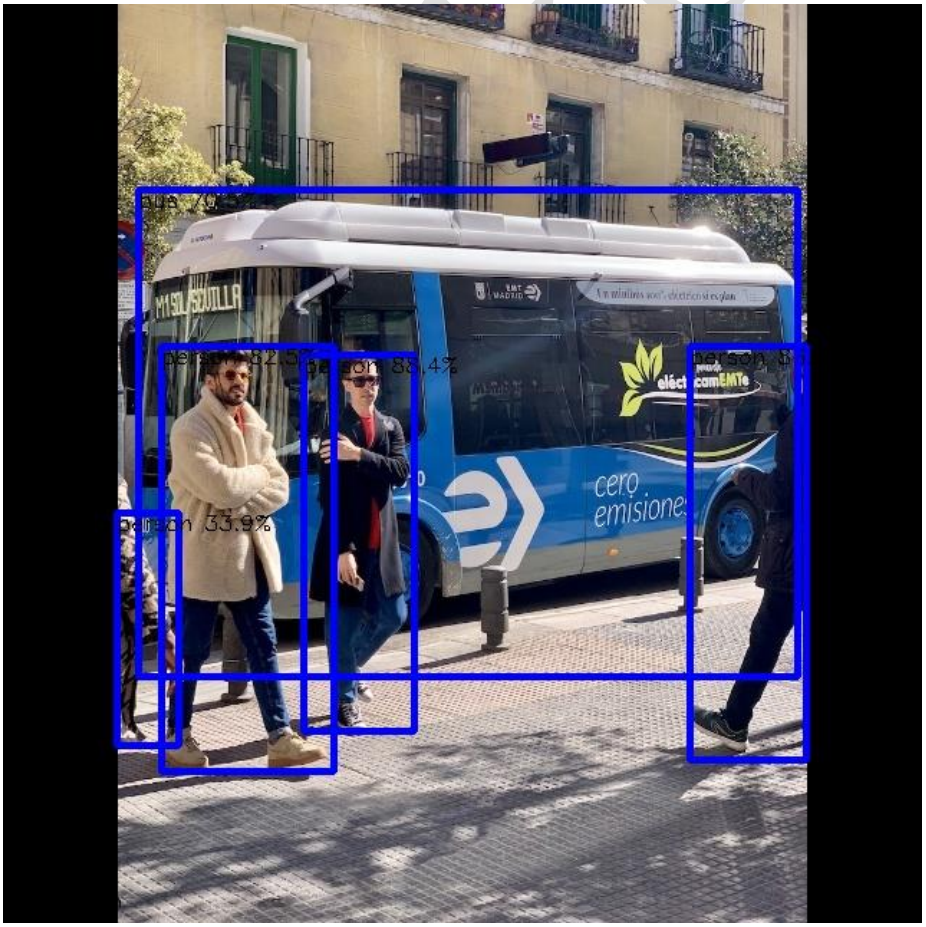
命令：

```
python3 test.py
```

```
I rknn building ...
I rknn buiding done.
done
--> Export rknn model
done
--> Init runtime environment
W init_runtime: Target is None, use simulator!
done
--> Running model
W inference: The 'data_format' has not been set and defaults is nhwc!
Analysing : 100%|████████████████████████████████| 153/153 [00:00<00:00, 10985.12it/s]
Preparing : 100%|████████████████████████████████| 153/153 [00:00<00:00, 414.30it/s]
W get_input_img: The dims of input(ndarray) shape (640, 640, 3) is wrong, expect dims is 4! Try expand dims to (1, 640, 640, 3)!
done
class: person, score: 0.884139358997345
box coordinate left,top,right,down: [208.8534364104271, 244.4304337501526, 286.3236876130104, 506.7466902732849]
class: person, score: 0.8676778078079224
box coordinate left,top,right,down: [478.82631254196167, 236.96079683303833, 559.7779355049133, 528.6053652763367]
class: person, score: 0.8246847987174988
box coordinate left,top,right,down: [110.32201385498047, 238.8315395116806, 230.29569244384766, 534.2514072656631]
class: person, score: 0.3394228219985962
box coordinate left,top,right,down: [79.96397459506989, 353.70939338207245, 122.13020265102386, 516.948687672615]
class: bus , score: 0.7050552368164062
box coordinate left,top,right,down: [92.08940839767456, 128.53247582912445, 554.9980549812317, 467.08300268650055]
```

此脚本是在 PC 模拟器上运行，若需要连板调试请参考章节 3.2。

转换模型和推理脚本 test.py 运行成功后，转换后的模型默认保存路径为 examples/onnx/yolov5/yolov5s_relu.rknn ，推理的图片结果保存在 examples/onnx/yolov5/result.jpg。

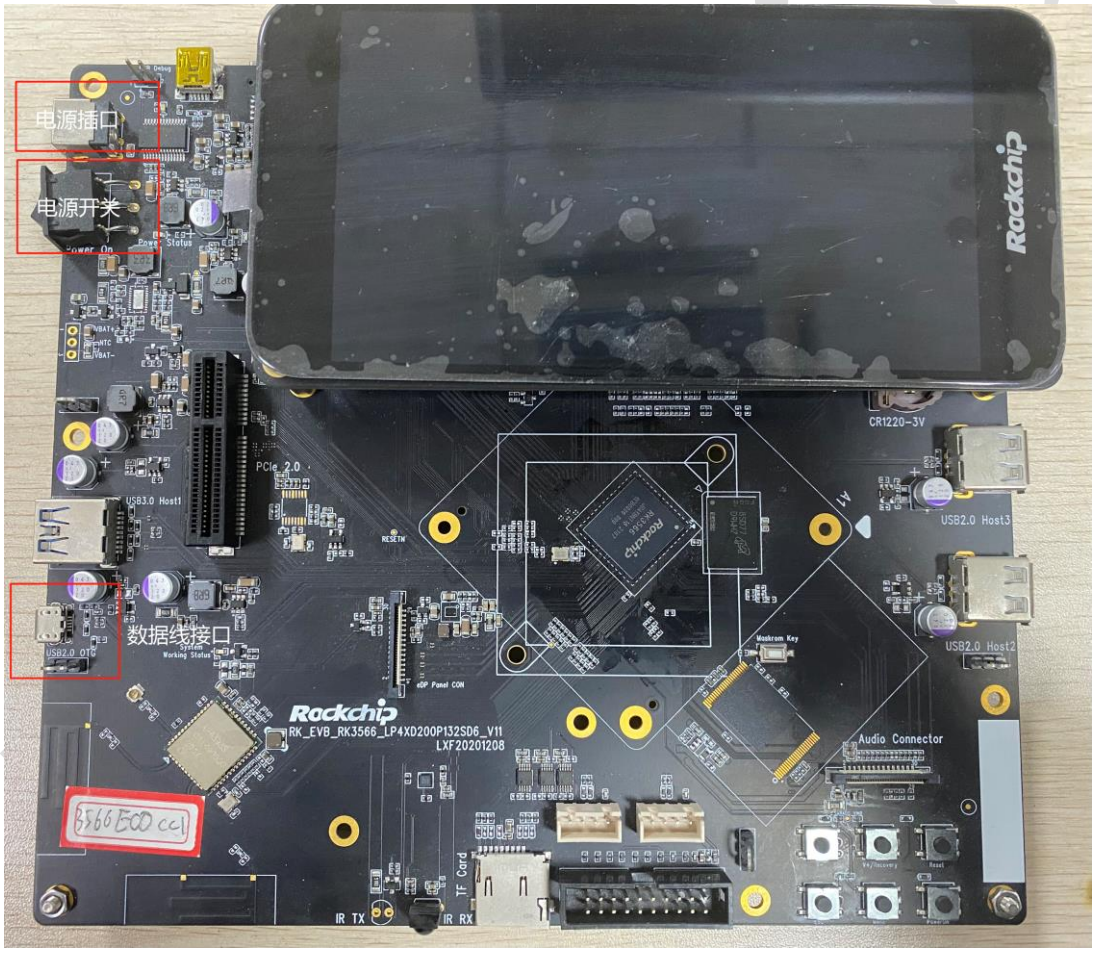


3.2 RKNN-Toolkit2 连板调试

通过 3.1.1 和 3.1.2 方式转换和推理模型均运行在 PC 端的模拟器环境中，可设置脚本中的 target 和 device_id 进行连板调试。此章节以 RK3566 为例说明。

3.2.1 连接板子至电脑

1. 按图中接口连接 RK3566 EVB 板子的电源、连接数据线至电脑，打开电源开关。



2. 查看板子设备。

命令：

```
adb devices
```

```
root@9e5c8ee3530b:/# adb devices
List of devices attached
* daemon not running; starting now at tcp:5037
* daemon started successfully
DKUZ8B0PAB      device
```

查看 RK3566 设备的 ID 为 DKUZ8B0PAB，连接成功。若无设备显示可参考附录 5.2。

3.2.2 更新板子的 rknn_server 和 librknnrt.so

librknnrt.so: 是一个板端的 runtime 库。

rknn_server: 是一个运行在板子上的后台代理服务，用于接收 PC 通过 USB 传输过来的协议，然后执行板端 runtime 对应的接口，并返回结果给 PC。

详细可参考 rknpu2/rknn_server_proxy.md 的详细说明。

下面以 RK3566_RK3568 安卓平台 64 位为例子介绍说明。

1) 打开一个终端命令窗口，进入 RKNPU2 工程目录。

```
cd ~/Projects/rknpu2
```

```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~$ cd ~/Projects/rknpu2
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ ls
doc  examples  LICENSE  README.md  rknn_server_proxy.md  runtime
```

2) 更新板子的 rknn_server 和 librknnrt.so。

命令：

```
adb root
adb remount

adb push \
runtime/RK356X/Android/rknn_server/arm64-v8a/vendor/bin/rknn_server \
/vendor/bin/

adb push \
runtime/RK356X/Android/librknn_api/arm64-v8a/librknnrt.so /vendor/lib64/

adb shell
chmod +x /vendor/bin/rknn_server
sync
reboot
```

```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ adb root
* daemon not running; starting now at tcp:5037
* daemon started successfully
adbd is already running as root
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ adb remount
remount succeeded
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ adb push runtime/RK356X/Android/rknn_serve
r/arm64-v8a/vendor/bin/rknn_server /vendor/lib/
runtime/RK356X/Android/rknn_server/arm64-v8a/ve...file pushed. 26.8 MB/s (850160 bytes in 0.030s)
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ adb push runtime/RK356X/Android/librknn_ap
i/arm64-v8a/librknnrt.so /vendor/lib/
runtime/RK356X/Android/librknn_api/arm64-v8a/librkn... 1 file pushed. 27.1 MB/s (5039024 bytes in 0.177s)
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$
```

3) 查看更新是否成功。

命令：

```
adb shell
su
setenforce 0
pgrep rknn_server
```

```
les/onnx/yolov5$ adb shell
rk3566_r:/ $ pgrep rknn_server
155
```

查看到有 rknn_server 进程 id，更新成功。

3.2.3 连板调试

1) 查看设备 ID 命令 adb devices。

```
root@4da66f8c2824:/rknn_yolov5_demo# adb devices
List of devices attached
* daemon not running; starting now at tcp:5037
* daemon started successfully
DKUZ8B0PAB      device
```

查看到此例子 RK3566 设备的 ID 为 DKUZ8B0PAB。若无设备显示可参考附录 5.2。

2) 修改脚本 target 和 device_id。

修改对应平台类型值（”rk3566”、”rk3568”、”rk3588”、”rv1103”、”rv1106”、”rk3562”）和设备 ID，保存后再执行脚本生成适用于板子的模型并进行推理图片，本例子中使用 rk3566 平台板子进行推理。

```python
# Create RKNN object
rknn = RKNN(verbose=True)


rknn.config(mean_values=[[0, 0, 0]], std_values=[[255, 255, 255]], target_platform='rk3566')
# Load model
print('--> Loading model')
ret = rknn.load_onnx(MODEL_PATH)
```

```python
# init runtime environment
print('--> Init runtime environment')
ret = rknn.init_runtime(target='rk3566', device_id='DKUZ8B0PAB')
if ret != 0:
    print('Init runtime environment failed')
    exit(ret)
print('done')
```

3) 执行转换和推理模型的 test.py 脚本：

```
python3 test.py
```

3.3 RKNPU2 的编译及使用方法

此章节以 rknn_yolov5_demo 在 RK3566 Android 64 位平台上运行为例，介绍如何使用 RKNPU2。

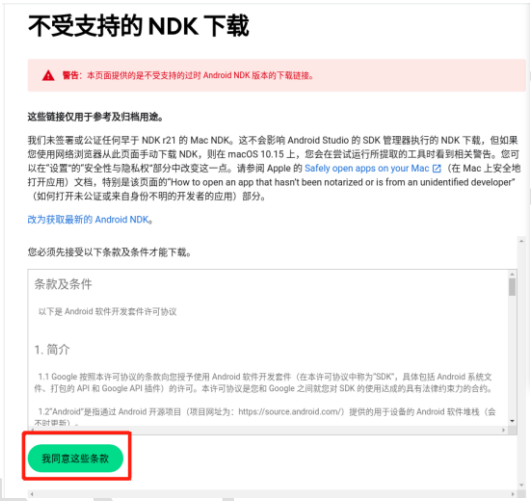
3.3.1 下载编译所需工具

下载完成后进行解压无需安装，记录文件夹的绝对路径。

1) 板子为 Android 系统则需 NDK，下载地址:

https://developer.android.google.cn/ndk/downloads/older_releases#ndk-17c-downloads

点击“我同意这些条款”。



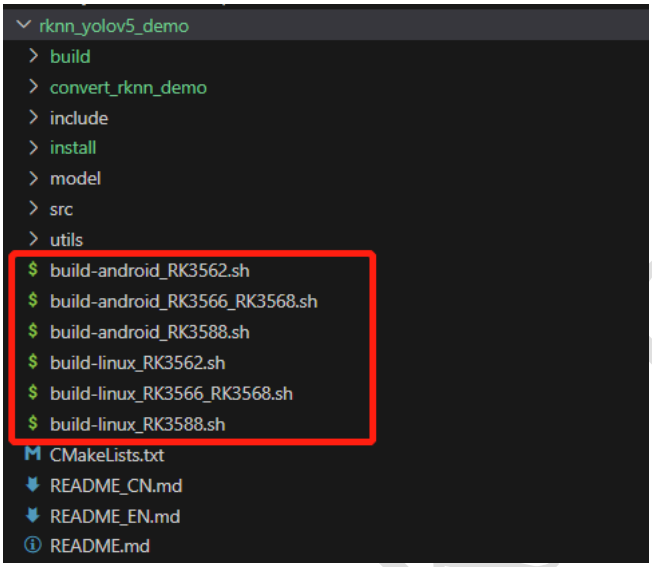
往下找到 Android NDK r17c（建议的版本）选择 Linux 64 位（x86）的软件包。

2) 板子为 linux 系统则需下载 gcc 交叉编译器。

推荐版本 gcc-9.3.0-x86_64_arrch64-linux-gnu，下载地址:

https://github.com/airockchip/gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu

3.3.2 修改 examples/rknn_yolov5_demo/build-XXX.sh 的编译工具路径



1) Android 系统

修改 build-android_RK3566_RK3568.sh 脚本中的 ANDROID_NDK_PATH 为本地电脑 android-ndk-r17c 的保存路径并保存。

2) Linux 系统

修改 TOOL_CHAIN 为本地电脑 gcc-9.3.0-x86_64_arrch64-linux-gnu 的保存路径并保存。

3.3.3 更新 RKNN 模型

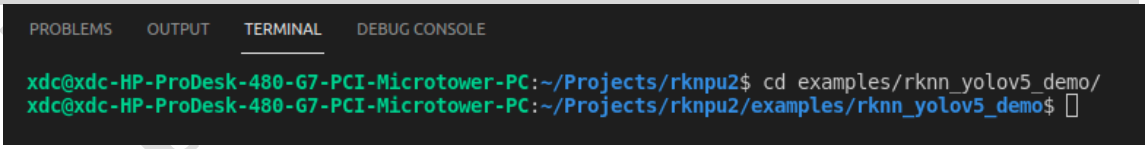
把章节 3.2 转换后的 RK3566 平台模型 yolov5s-640-640.rknn 复制到 rknpu2/examples/rknn_yolov5_demo/model/RK3566_RK3568/目录下。

3.3.4 编译 rknn_yolov5_demo

1) 在终端命令窗口进入 rknn_yolov5_demo 文件夹。

命令：

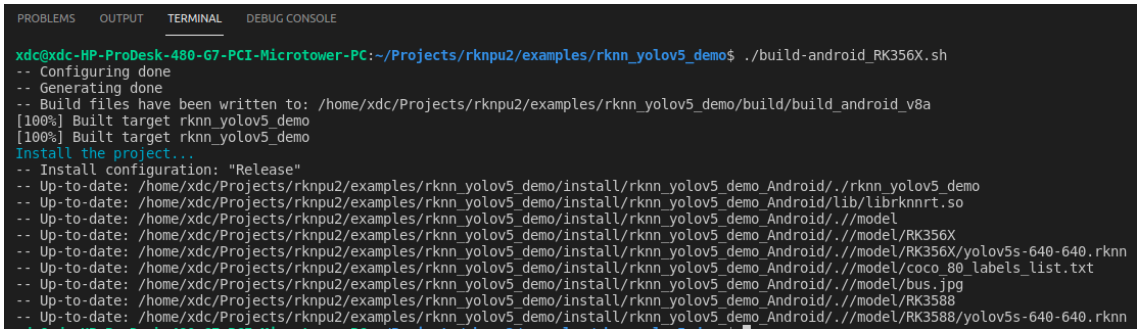
```
cd examples/rknn_yolov5_demo/
```



2) 运行 build-android_RK3566_RK3568.sh 脚本编译程序。

命令：

```
./build-android_RK3566_RK3568.sh
```



备注：

1) 此例子为编译 RK3566_RK3568 的安卓 64 位平台。若需要编译其他平台请选择相应的脚本。详情可参考/rknpu2/examples/rknn_yolov5_demo/README.md。

2) 若在编译时出现 cmake 错误，可执行以下命令安装 cmake 后再运行编译脚本。

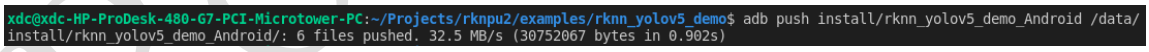
```
sudo apt install cmake
```

3.3.5 在板端运行 rknn_yolov5_demo

1) 把编译好的程序和所需的文件 install/rknn_yolov5_demo_Android 文件夹上传到板子的/data/文件夹下。

命令：

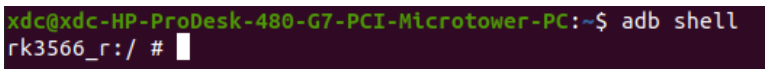
```
adb root
adb push install/rknn_yolov5_demo_Android /data/
```



2) 进入板子系统。

命令：

```
adb shell
```



3) cd 进入程序所在的目录。

命令：

```
cd /data/rknn_yolov5_demo_Android/
```

```
rk3566_r:/ # cd /data/rknn_yolov5_demo_Android/
rk3566_r:/data/rknn_yolov5_demo_Android # ls
lib  model  rknn_yolov5_demo
```

4) 设置库文件路径。

命令：

```
export LD_LIBRARY_PATH=./lib
```

5) 运行程序识别相应的图片中物体的类别。

用法 Usage: ./rknn_yolov5_demo <rknn model> <jpg>

命令：

```
./rknn_yolov5_demo ./model/RK3566_RK3568/yolov5s-640-640.rknn ./model/bus.jpg
```

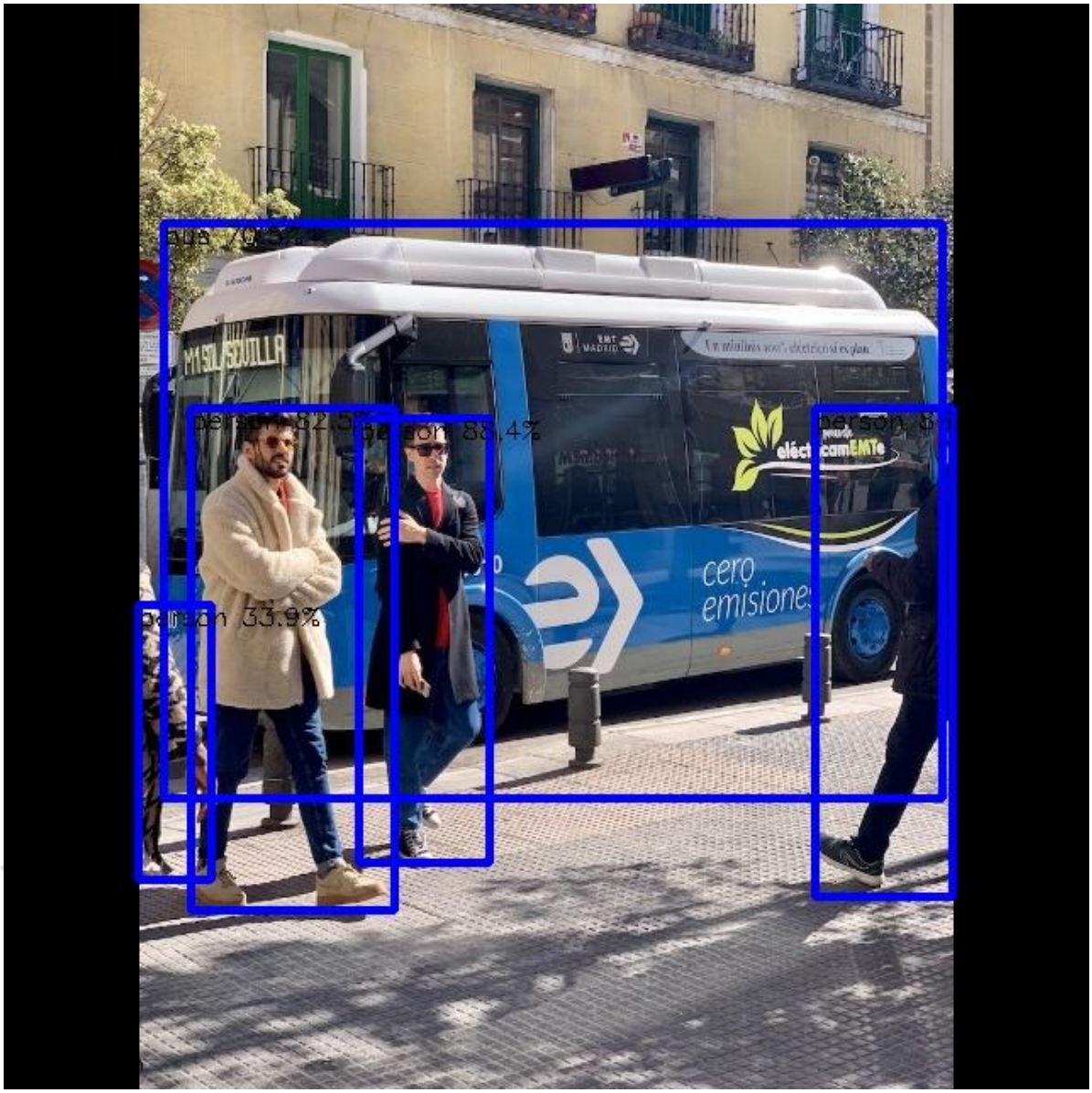
```
/rknn_yolov5_demo model/RK3566_RK3568/yolov5s-640-640.rknn model/bus.jpg                                    <
post process config: box_conf_threshold = 0.25, nms_threshold = 0.45
Read model/bus.jpg ...
img width = 640, img height = 640
Loading mode...
sdk version: 1.5.2 (a1529deb8@2023-08-22T14:55:12) driver version: 0.8.0
model input num: 1, output num: 3
  index=0, name=images, n_dims=4, dims=[1, 640, 640, 3], n_elems=1228800, size=1228800, w_stride = 640, size_with_stride=122
8800, fmt=NHWC, type=INT8, qnt_type=AFFINE, zp=-128, scale=0.003922
  index=0, name=output, n_dims=4, dims=[1, 255, 80, 80], n_elems=1632000, size=1632000, w_stride = 0, size_with_stride=16384
00, fmt=NCHW, type=INT8, qnt_type=AFFINE, zp=-128, scale=0.003860
  index=1, name=283, n_dims=4, dims=[1, 255, 40, 40], n_elems=408000, size=408000, w_stride = 0, size_with_stride=409600, fm
t=NCHW, type=INT8, qnt_type=AFFINE, zp=-128, scale=0.003922
  index=2, name=285, n_dims=4, dims=[1, 255, 20, 20], n_elems=102000, size=102000, w_stride = 0, size_with_stride=122880, fm
t=NCHW, type=INT8, qnt_type=AFFINE, zp=-128, scale=0.003915
model is NHWC input fmt
model input height=640, width=640, channel=3
once run use 71.482000 ms
loadLabelName ./model/coco_80_labels_list.txt
person @ (209 244 286 506) 0.884139
person @ (478 238 559 526) 0.867678
person @ (110 238 230 534) 0.824685
bus @ (94 129 553 468) 0.705055
person @ (79 354 122 516) 0.339254
loop count = 10 , average run  54.913700 ms
```

6) 打开一个新终端窗口并下载结果图片 out.jpg 至本地电脑中查看。

命令：

```
adb pull /data/rknn_yolov5_demo_Android/out.jpg ./
```

```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Pictures$ adb pull /data/rknn_yolov5_demo_Android/out.jpg ./
/data/rknn_yolov5_demo_Android/out.jpg: 1 file pulled. 25.2 MB/s (189698 bytes in 0.007s)
```



4 参考文档

有关 RKNN-Toolkit2 更详细的用法和接口说明，请参考《Rockchip_User_Guide_RKNN_Toolkit2_CN.pdf》手册。

有关 RKNPU API 更详细的用法和接口说明，请参考《Rockchip_RKNPU_User_Guide_RKNN_API_CN.pdf》手册。

5 附录

5.1 查看和设置开发板的 CPU、DDR 和 NPU 频率

通常，板子上的各个单元的频率是动态调频，这种情况下测试出来的模型性能会有波动。为了防止性能测试结果不一致，在性能评估时，建议固定板子上的相关单元的频率再做测试。相关单元的频率查看和设置命令如下：

5.1.1 CPU 定频命令

1) 查看 CPU 频率：

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

2) 固定 CPU 频率：

```
# 查看 CPU 可用频率（不同平台显示的可用频率会有所不同）
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
408000 600000 816000 1008000 1200000 1416000 1608000 1704000
# 设置 CPU 频率，例如，设置 1.7GHz
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
echo 1704000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
```

5.1.2 DDR 定频命令

1) 查看 DDR 频率（**需固件支持**）：

```
cat /sys/class/devfreq/dmc/cur_freq
或
cat /sys/kernel/debug/clk/clk_summary | grep ddr
```

2) 固定 DDR 频率（**需固件支持**）：

```
# 查看 DDR 可用频率
cat /sys/class/devfreq/dmc/available_frequencies
# 设置 DDR 频率，例如，设置 1560MHz
echo userspace > /sys/class/devfreq/dmc/governor
echo 1560000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

5.1.3 NPU 定频命令

1) 查看 NPU 频率（**需固件支持**）：

对于 RK3566_RK3568:

```
cat /sys/kernel/debug/clk/clk_summary | grep npu
或
cat /sys/class/devfreq/fde40000.npu/cur_freq
```

对于 RK3588:

```
cat /sys/kernel/debug/clk/clk_summary | grep clk_npu_dsu0
```

对于 RK3562:

```
cat /sys/class/devfreq/ff300000.npu/cur_freq
```

2) 固定 NPU 频率（**需固件支持**）：

对于 RK3566_RK3568:

```
# 查看 NPU 可用频率
cat /sys/class/devfreq/fde40000.npu/available_frequencies
# 设置 NPU 频率，例如，设置 1 GHz
echo userspace > /sys/class/devfreq/fde40000.npu/governor
echo 1000000000 > /sys/kernel/debug/clk/clk_scmi_npu/clk_rate
```

对于 RK3588:

```
# 设置 NPU 频率，例如，设置 1GHz
echo 1000000000 > /sys/kernel/debug/clk/clk_npu_dsu0/clk_rate
```

对于 RK3562:

```
# 查看 NPU 可用频率
cat /sys/class/devfreq/ff300000.npu/available_frequencies
# 设置 NPU 频率，例如，设置 600MHz
echo userspace > /sys/class/devfreq/ff300000.npu/governor
echo 600000000 > /sys/class/devfreq/ff300000.npu/userspace/set_freq
```

5.2 命令 adb devices 查看不到设备

1. 检查连线是否正确、重插拔或更换电脑 USB 插口。
2. 在本地电脑和 docker 容器中使用 USB 连接的板子时，同一时间只能有一端使用 adb server。

   故若在一端执行命令（adb devices）时查看不到设备，可在另一命令端执行命令。

```
adb kill-server
```

终止外部 adb service，再返回原先命令终端窗口执行命令（adb devices）查看设备。

3. 出现以下错误时，是未安装 adb。需要执行安装命令安装 adb。

```
Command 'adb' not found, but can be installed with:
sudo apt install adb
```

命令：

```
sudo apt install adb
```