Classification Level: Top secret ( ) Secret ( ) Internal ( ) Public (√)

RKNN SDK Quick Start Guide

(Technology Department, Graphic & Computing Platform Center)

Mark: [ ] Editing [√] Released	Version:	1.5.2
	Author:	HPC
	Completion Date:	2023-8-22
	Auditor:	Vincent
	Reviewed Date:	2023-8-22

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd

Revision History

Version	Author	Revision Date	Revision Notes	Auditor
1.2.0	Roy	2022-1-14	initial version	Randall
1.3.0	HPC	2022-4-22	updated version	Randall
1.4.0	HPC	2022-8-22	updated version	Randall
1.4.2	HPC	2023-2-13	updated version	Randall
1.5.0	HPC	2023-5-22	updated version	Randall
1.5.2	HPC	2023-8-17	updated version	Randall

Contents

1 Overview

This document guides users on how to quickly use RKNN-Toolkit2 and RKNPU2 tools on the EVB board of the ROCKCHIP chip to convert the yolov5s.onnx model to the yolov5s.rknn model and perform edge inference.

Supported platforms: RK3566, RK3568, RK3588, RK3588S, RK3562. In order to simplify the description, the follow-up instructions use RK3566_RK3568 to represent the RK3566 or RK3568 platform.

RKNPU2 project download address: https://github.com/rockchip-linux/rknpu2

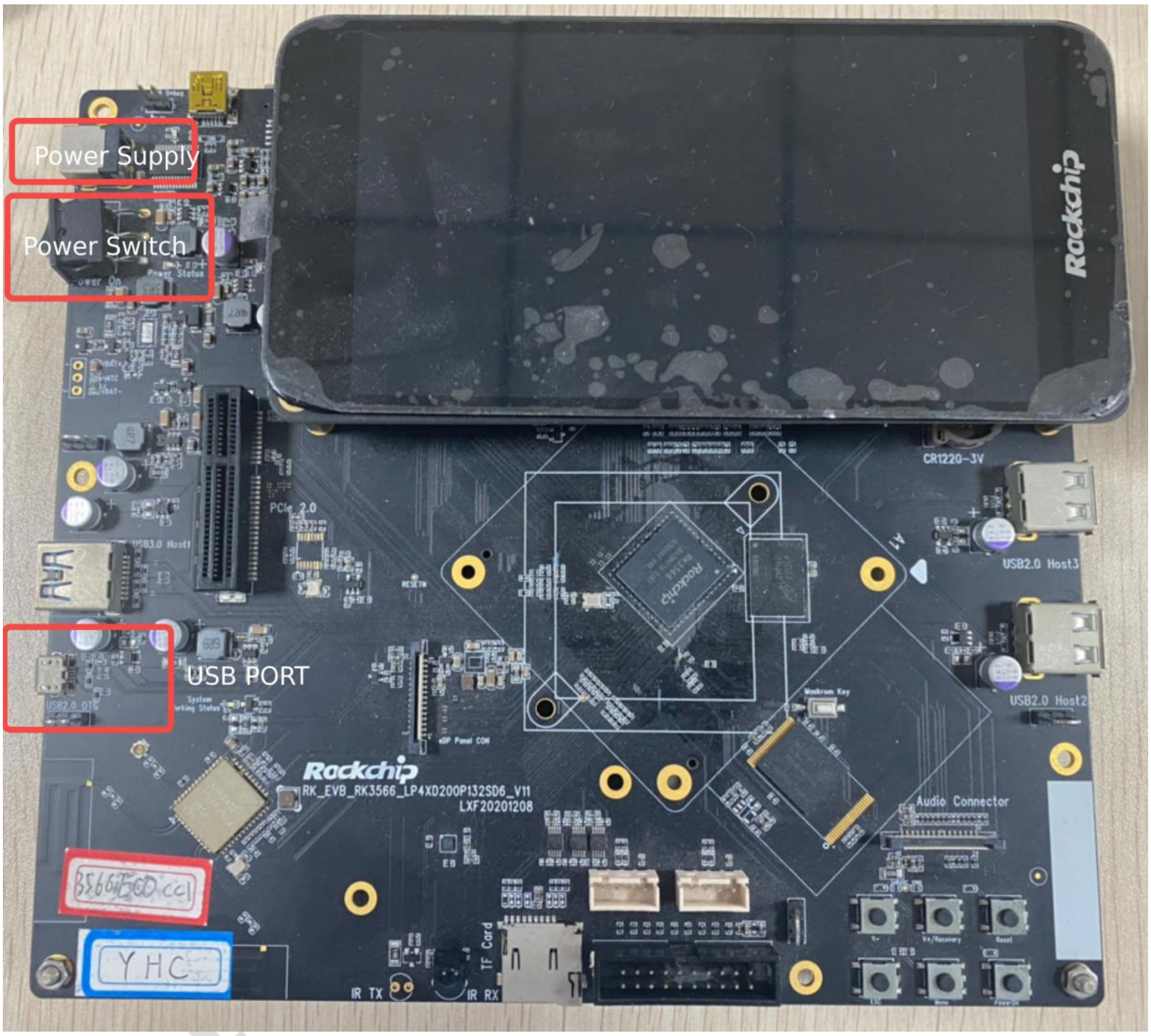
RKNN-Toolkit2 project download address: https://github.com/rockchip-linux/rknn-toolkit2

2 Prepare Tools

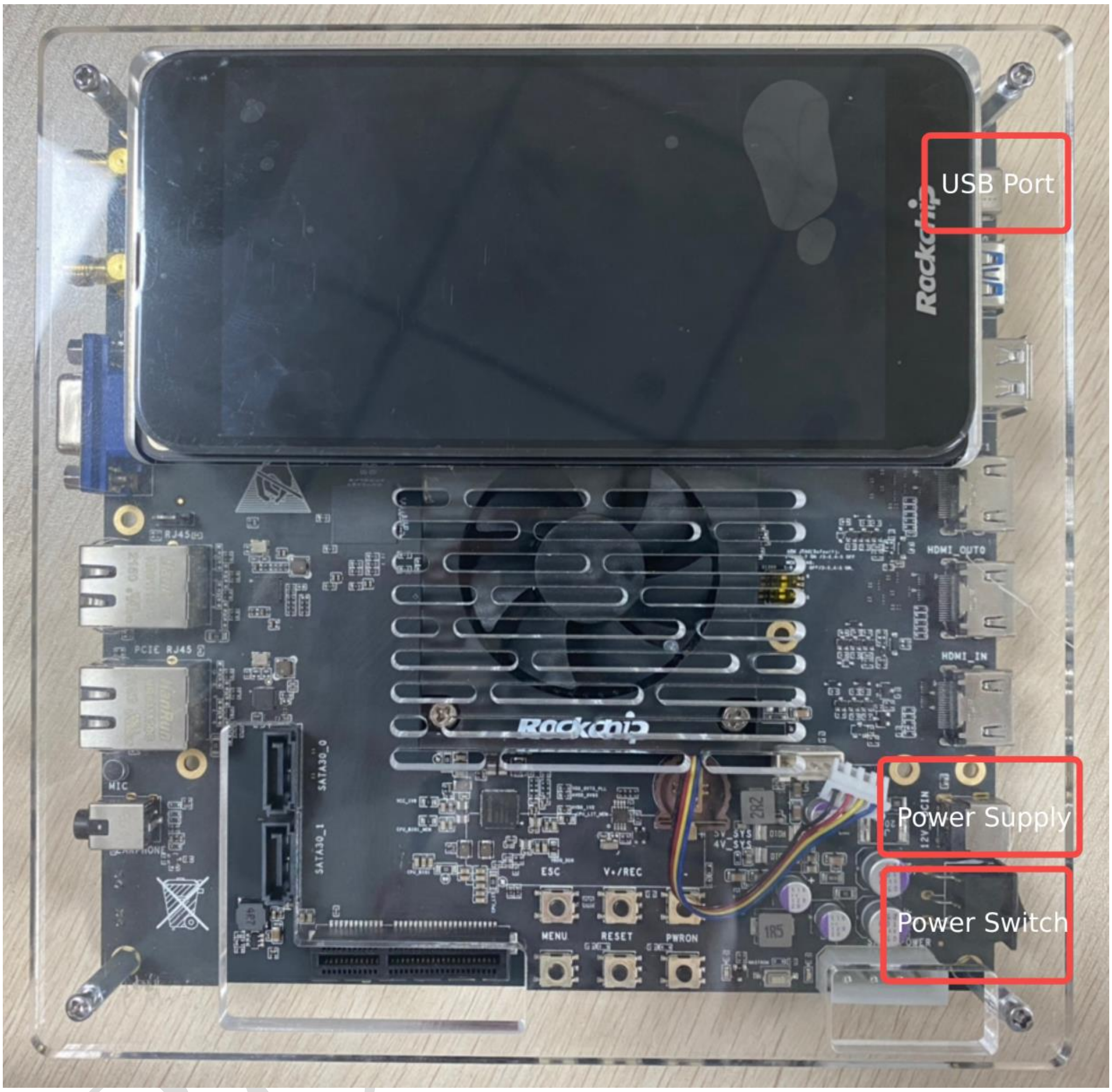
1. A computer with operating system Ubuntu18.04 / Ubuntu20.04 / Ubuntu22.04.

2. An EVB board (RK3566, RK3568, RK3588, RK3588S, RK3562).

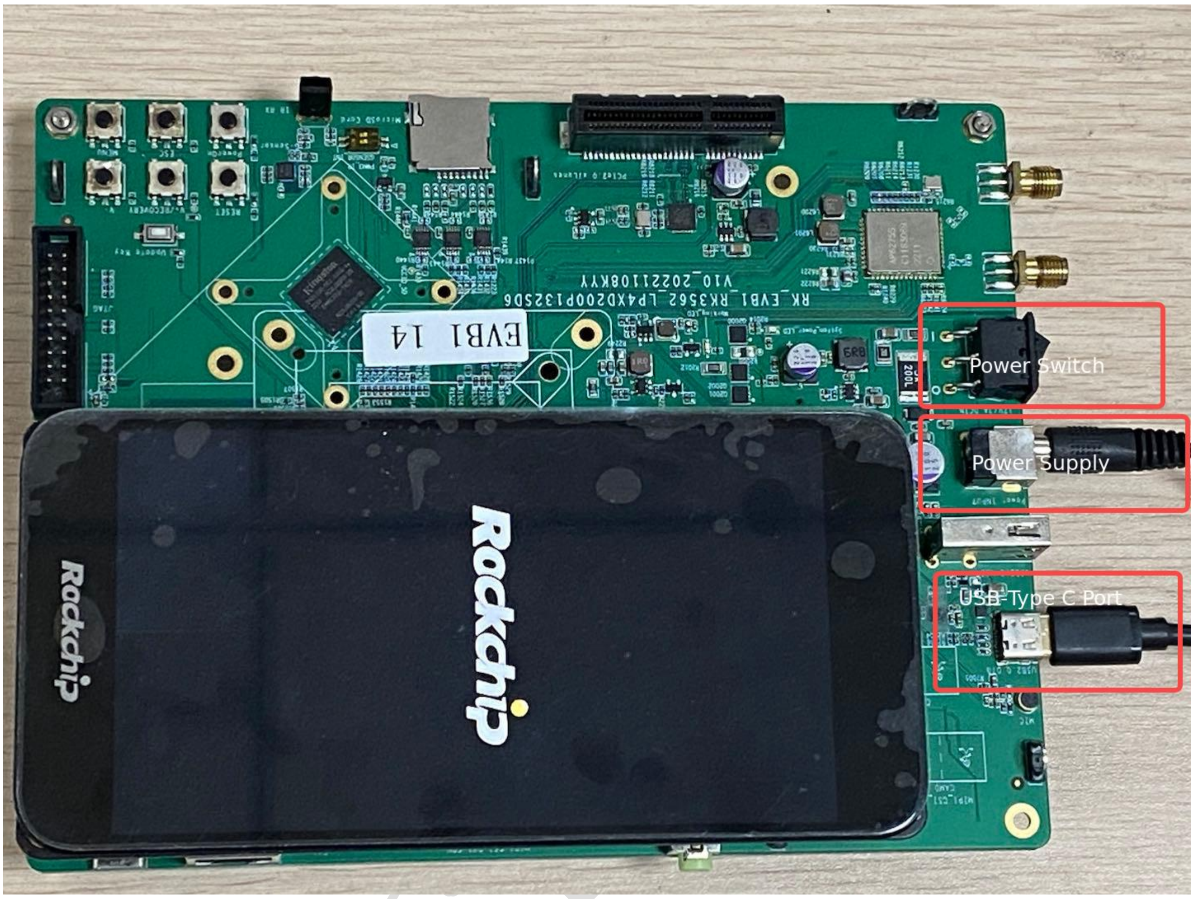
RK3566



RK3588



RK3562



3. A data cable connecting the board to the computer.

RK3566_RK3568:USB-A——Micro USB

RK3588/RK3562:USB-A——USB-C

4. A power adapter.

RK3562/RK3566_RK3568:output 12V-2A

RK3588:output 12V-3A

3 Quick Start Using RKNN-Toolkit2 and RKNPU2

3.1 Install RKNN-Toolkit2

This chapter introduces two methods of installing and using RKNN-Toolkit2, 'installation via pip install' and 'installation via Docker image'. Users can choose the installation method themselves. If the system is not Ubuntu18.04/Ubuntu20.04/Ubuntu22.04 system, it is recommended to use the 'installation via Docker image' method, which has integrated all the required installation package dependencies.

The following operations use Ubuntu18.04 and Python3.6 as an example.

3.1.1 Install and go through Docker images

1. If the computer does not have the Docker tool installed, please follow this installation tutorial to install the Docker tool before proceeding to the next step （https://mirrors.tuna.tsinghua.edu.cn/help/docker-ce/）.

2. Open a terminal command line window, cd into the docker folder of the RKNN-Toolkit2 project, and modify the path in the cd command according to the save path of the project.

   cd <Enter the path of the docker folder in the RKNN-Toolkit2 project>

   Command:

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

   Check that there is a docker image file rknn-toolkit2-1.x.x-cp36-docker.tar.gz in the current directory。

3. Load the docker image.

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

4. View all current docker images.

   Command:

```
docker images
```

It can be found that the REPOSITORY is rknn-toolkit2, and the TAG is 1.x.x-cp36, which means the loading is successful.

5. Run docker container.

   Command:

```
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
```

Mapping a directory into a Docker environment can be done by appending '-v <host src folder>:<image dst folder>'.

The green part is the examples/onnx/yolov5 local folder path in the RKNN-Toolkit2 project (modified according to the local path) mapped to the /rknn_yolov5_demo folder in the docker container.

After successfully entering the docker container, the command 'ls' can view the folder rknn_yolov5_demo, indicating that the mapping is successful.

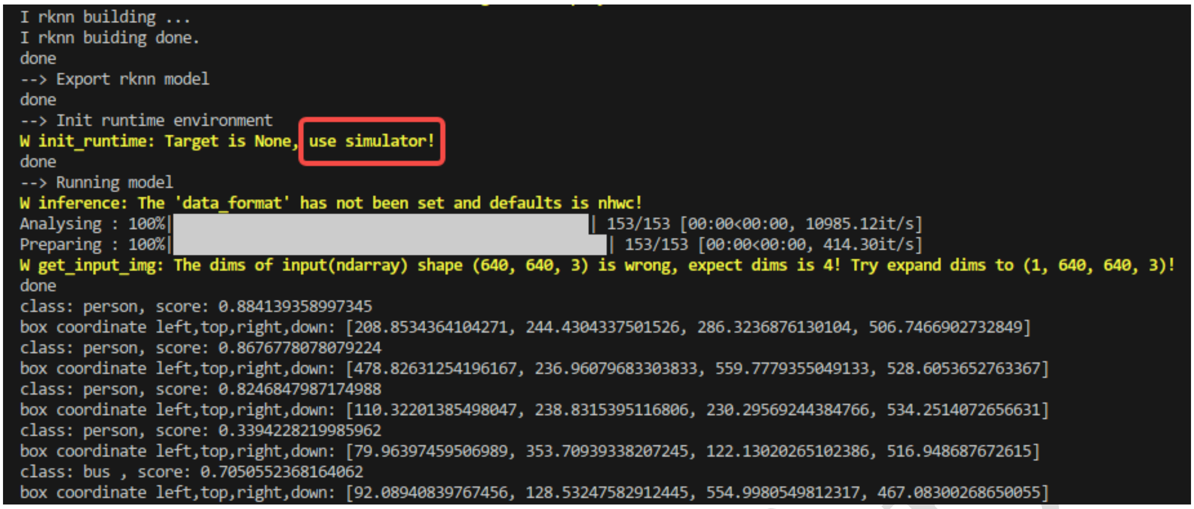
6. Enter the rknn_yolov5_demo directory in the docker container.

   Commadn:

```
cd rknn_yolov5_demo
```

7. Convert yolov5s_relu.onnx to rknn model parallel inference image.

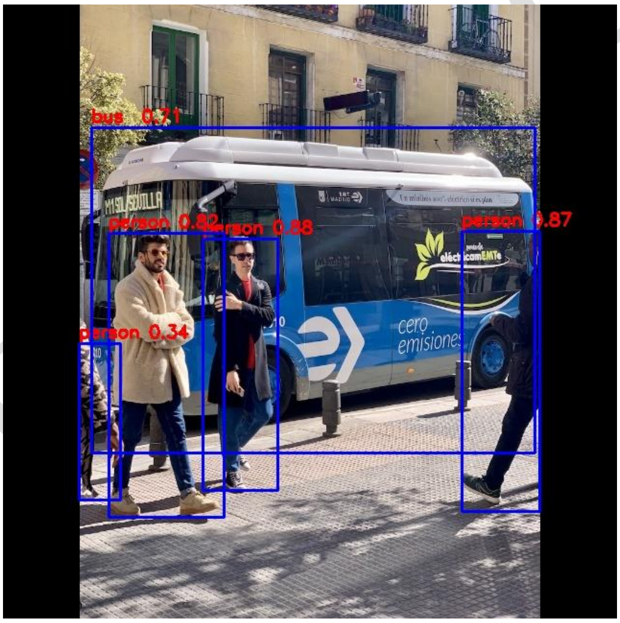
```
python3 ./test.py
```



```
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]
```

This script is run on the PC emulator, if you need to debug with the board, please refer to Chapter 3.2.

The conversion model and inference script test.py run successfully. The converted model is saved in the default path of examples/onnx/yolov5/yolov5s_relu.rknn, and the inference image result is saved in examples/onnx/yolov5/result.jpg.



3.1.2 Install and reason through pip install

1. Open a terminal command line window, install Python3.6 and pip3.

   Command:

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

2. Install required dependent packages.

   Command:

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

3. Enter the Toolkit2 project folder, and modify the path in the cd command according to the project save path.

   cd <Enter the path of the Toolkit2 project>

   Command:

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

4. Install the necessary corresponding versions of the dependent packages.

   Command:

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

PS:

1) If the error 'XX version cannot be matched' occurs during the installation process, it may be caused by the pip version being too low. You can execute the following upgrade pip version command first, upgrade pip to version 21.3.1, and then execute the above installation command again.

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

5. Install RKNN-Toolkit2 (Python3.6 for x86_64).

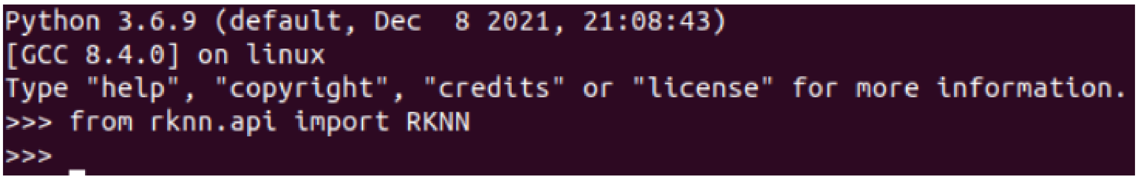
   Command:

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

6. Check whether RKNN-Toolkit2 is installed successfully.

Command:

```
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
>>> 
```

If there are no errors, the installation is successful. Press and hold Ctrl+D to exit Python3.

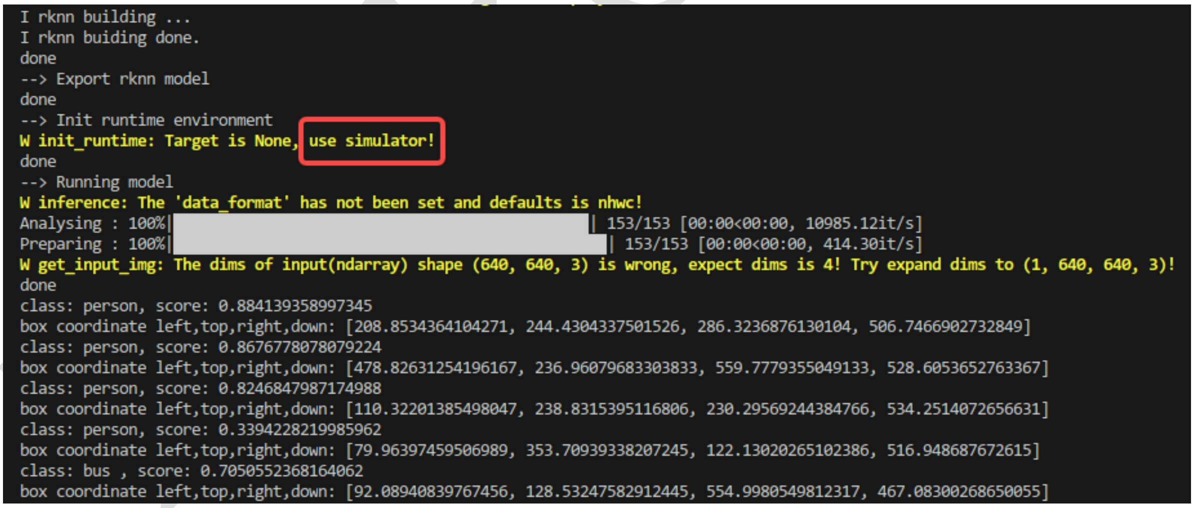
7. cd into rknn-toolkit2-1.x.x/examples/onnx/yolov5 contents.

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

8. Convert yolov5s_relu.onnx to rknn model and run model inference picture.

Command:

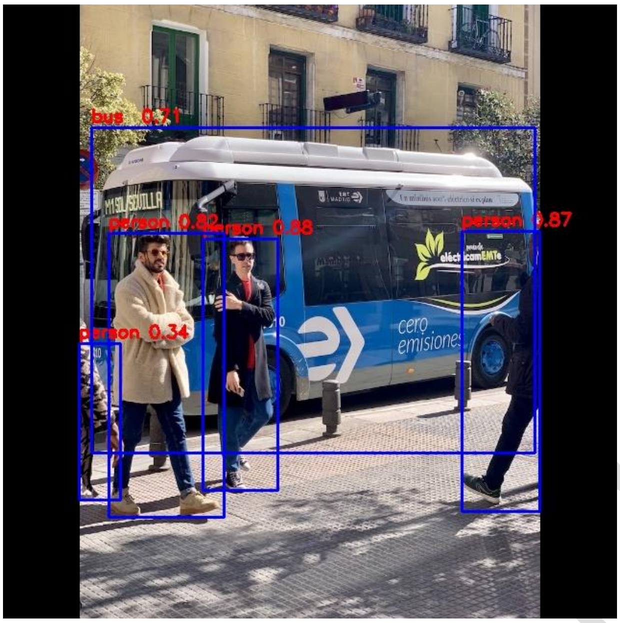
```
python3 test.py
```



```
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]
```

This script is run on the PC emulator, if you need to debug with the board, please refer to Chapter 3.2.

The conversion model and inference script test.py run successfully. The converted model is saved in the default path of examples/onnx/yolov5/yolov5s_relu.rknn, and the inference image result is saved in examples/onnx/yolov5/result.jpg.

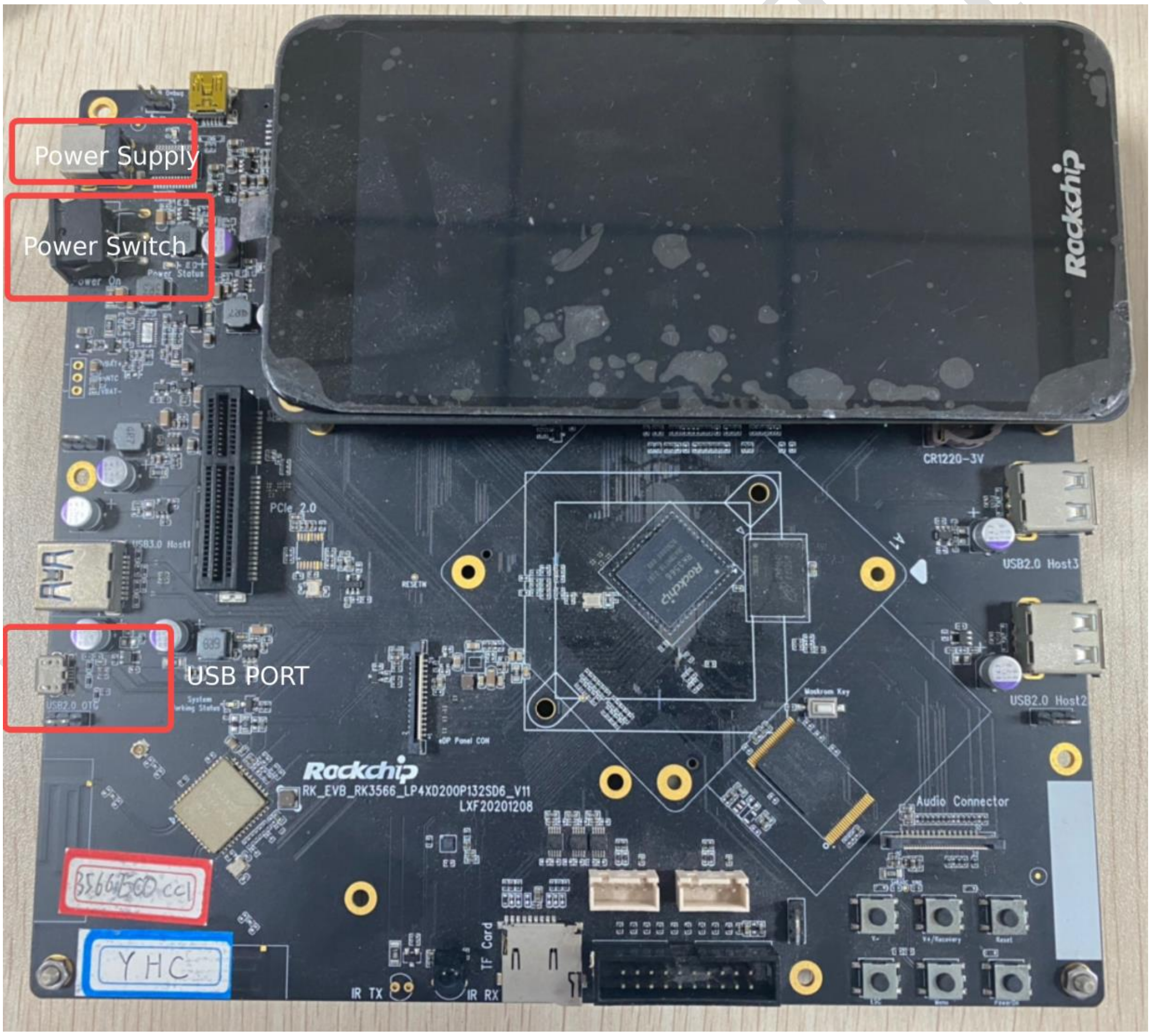

bus 0.71
person 0.82
person 0.88
person 0.87
person 0.34

3.2 RKNN-Toolkit2 Continuous Debugging

Both the conversion and inference models through 3.1.1 and 3.1.2 are run in the simulator environment on the PC side, and the target and device_id in the script can be set for continuous debugging. This chapter takes RK3566 as an example to illustrate.

3.2.1 Connect the board to the computer

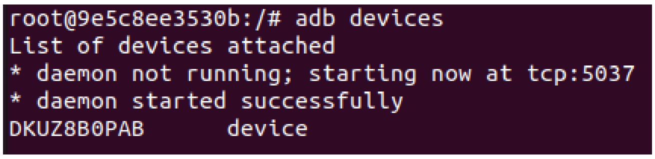
1. Connect the power supply of the RK3566 EVB board according to the interface in the figure, connect the data cable to the computer, and turn on the power switch.



2. View board equipment.

Command:

```
adb devices
```



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

Check that the ID of the RK3566 device is DKUZ8B0PAB, and the connection is successful. If no device is displayed, please refer to Appendix 5.2。

3.2.2 Update the rknn_server and librknnrt.so of the board

librknnrt.so: is a board-side runtime library.

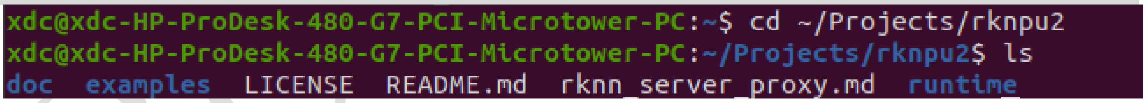
rknn_server: It is a background proxy service running on the board, which is used to receive the protocol transmitted by the PC through USB, then execute the interface corresponding to the board runtime, and return the result to the PC.

For details, please refer to the detailed description of rknpu2/rknn_server_proxy.md.

The following is an example of RK3566_RK3568 Android 64-bit platform.

1) Open a terminal command window and enter the RKNPU2 project directory.

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

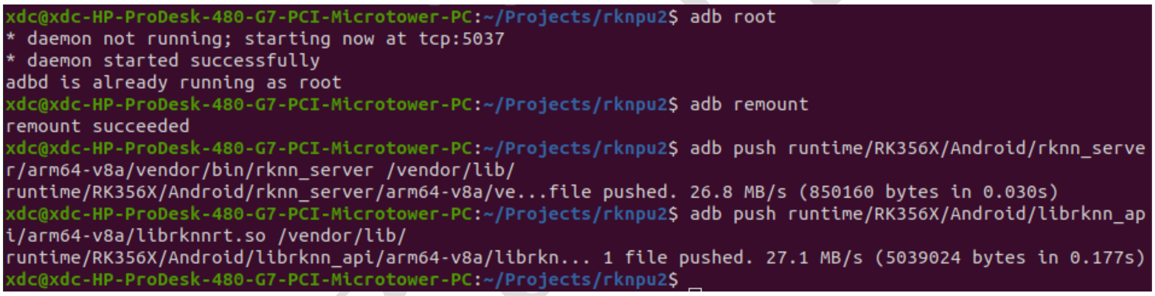


```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~$ cd ~/Projects/rknpu2
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ ls
doc   examples   LICENSE   README.md   rknn_server_proxy.md   runtime
```

2) Update the rknn_server and librknnrt.so of the board.

Command:

```
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
```

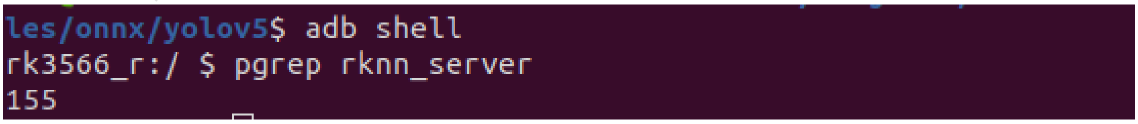


```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ adb root
* daemon not running; starting now at tcp:5037
* daemon started successfully
adbd is already running as root
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ adb remount
remount succeeded
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ 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) Check if the update was successful.

Command:

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

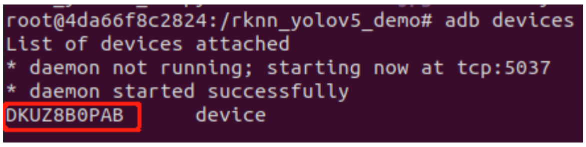


```
les/onnx/yolov5$ adb shell
rk3566_r:/ $ pgrep rknn_server
155
```

If found that there was a rknn_server process id, and the update is successful。

3.2.3 Continuous Debugging

1) View device ID command adb devices.

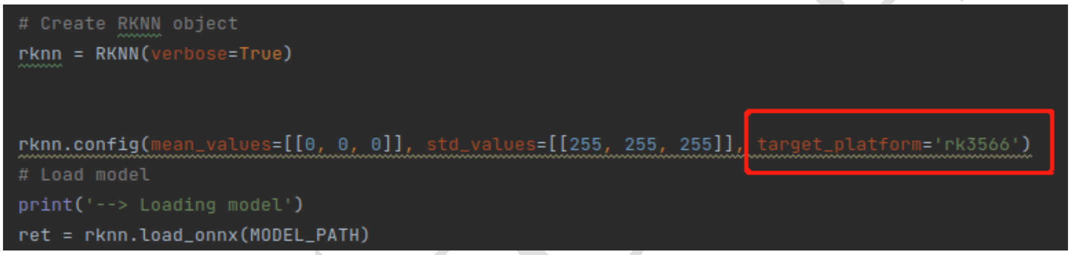


```
root@4da66f8c2824:/rknn_yolov5_demo# adb devices
List of devices attached
* daemon not running; starting now at tcp:5037
* daemon started successfully
DKUZ8B0PAB      device
```

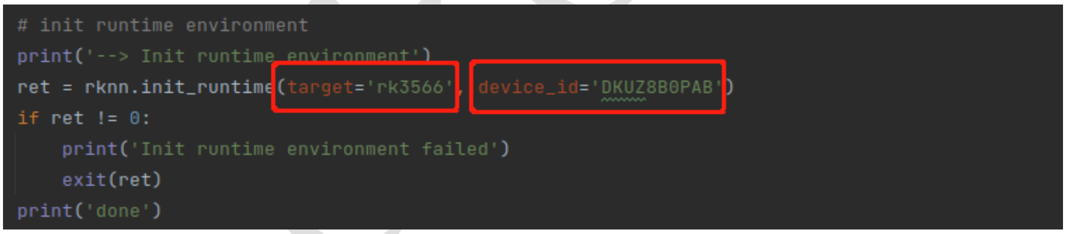
It can be seen that the ID of the RK3566 device in this example is DKUZ8B0PAB. If no device is displayed, please refer to Appendix 5.2.

2) Modify script target and device_id.

Modify the corresponding platform type value (‘rk3566’, ‘rk3568’, ‘rk3588’, ‘rv1103’, ‘rv1106’, ‘rk3562’) and device ID, save and then execute the script to generate a model suitable for the board and perform inference pictures. In this example, the rk3566 platform board is used for inference.



```
# 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)
```



```
# 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) The 'test.py' script that performs the transformation and inference of the model:

```
python3 test.py
```

3.3 How to compile and use RKNPU2

This chapter utilize rknn_yolov5_demo running on RK3566 Android 64-bit platform as an example to introduce how to use RKNPU2.

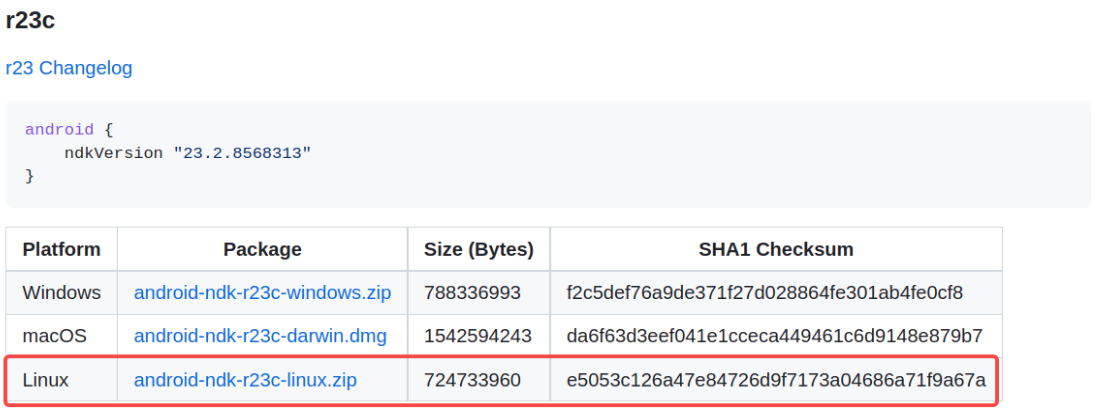
3.3.1 Download the tools required for compilation

After the download is complete, decompress without installation, and record the absolute path of the folder.

1) If the board is an Android system, NDK is required, download link:

   https://github.com/android/ndk/wiki/Unsupported-Downloads#ndk-17c-downloads

   Scroll down to find the Android NDK r23 (recommended version) package for Linux 64-bit (x86).



r23c

r23 Changelog

```
android {
    ndkVersion "23.2.8568313"
}
```

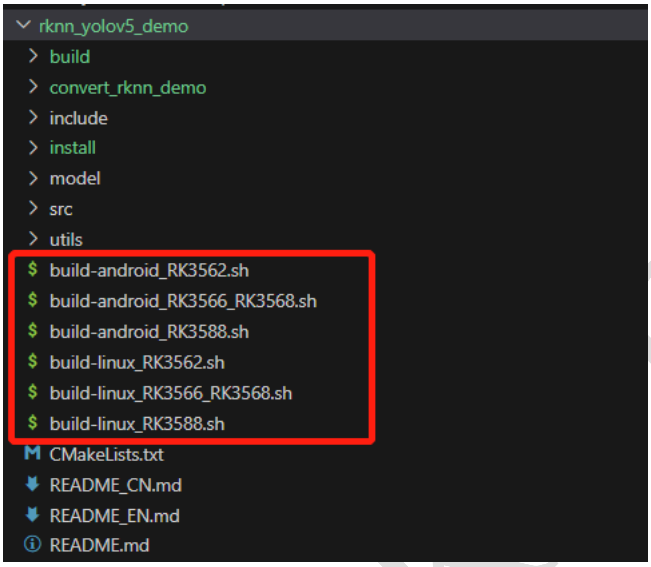
Platform	Package	Size (Bytes)	SHA1 Checksum
Windows	android-ndk-r23c-windows.zip	788336993	f2c5def76a9de371f27d028864fe301ab4fe0cf8
macOS	android-ndk-r23c-darwin.dmg	1542594243	da6f63d3eef041e1cceca449461c6d9148e879b7
Linux	android-ndk-r23c-linux.zip	724733960	e5053c126a47e84726d9f7173a04686a71f9a67a

2) If the board is a linux system, you need to download the gcc cross compiler.

Recommended version gcc-9.3.0-x86_64_arrch64-linux-gnu, download address:

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

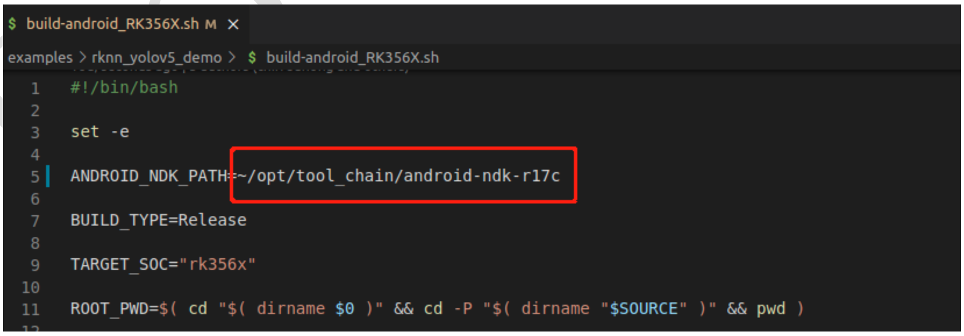
3.3.2 Modify the compilation tool path of examples/rknn_yolov5_demo/build-XXX.sh



rknn_yolov5_demo
- build
- convert_rknn_demo
- include
- install
- model
- src
- utils
- build-android_RK3562.sh
- build-android_RK3566_RK3568.sh
- build-android_RK3588.sh
- build-linux_RK3562.sh
- build-linux_RK3566_RK3568.sh
- build-linux_RK3588.sh
- CMakeLists.txt
- README_CN.md
- README_EN.md
- README.md

1) Android System

Modify the ANDROID_NDK_PATH in the build-android_RK3566_RK3568.sh script to the save path of the local computer android-ndk-r17c and save it.

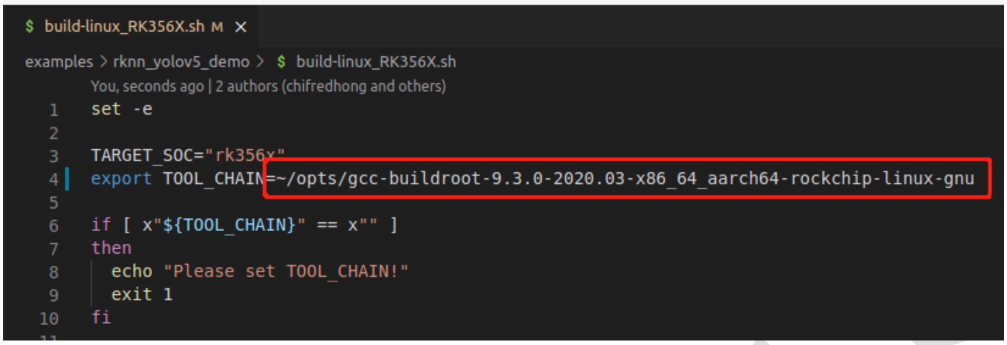


build-android_RK356X.sh M ×
examples > rknn_yolov5_demo > build-android_RK356X.sh

```
#!/bin/bash

set -e

ANDROID_NDK_PATH=~/opt/tool_chain/android-ndk-r17c

BUILD_TYPE=Release

TARGET_SOC="rk356x"

ROOT_PWD=$( cd "$( dirname $0 )" && cd -P "$( dirname "$SOURCE" )" && pwd )
```

2) Linux System

Modify TOOL_CHAIN to your own full path of the gcc-9.3.0-x86_64_arrch64-linux-gnu and save.


$ build-linux_RK356X.sh M ×
examples > rknn_yolov5_demo > $ build-linux_RK356X.sh
You, seconds ago | 2 authors (chifredhong and others)

```
set -e

TARGET_SOC="rk356x"
export TOOL_CHAIN=~/opts/gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu

if [ x"${TOOL_CHAIN}" == x"" ]
then
  echo "Please set TOOL_CHAIN!"
  exit 1
fi
```

3.3.3 Update RKNN Model

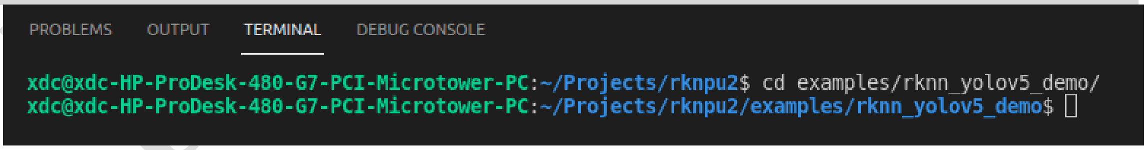
Copy the converted RK3566 platform model yolov5s-640-640.rknn in Chapter 3.2 to the rknpu2/examples/rknn_yolov5_demo/model/RK3566_RK3568/ directory.

3.3.4 Compile rknn_yolov5_demo

1) Enter the rknn_yolov5_demo folder in the terminal command window.

Command:

```
cd examples/rknn_yolov5_demo/
```

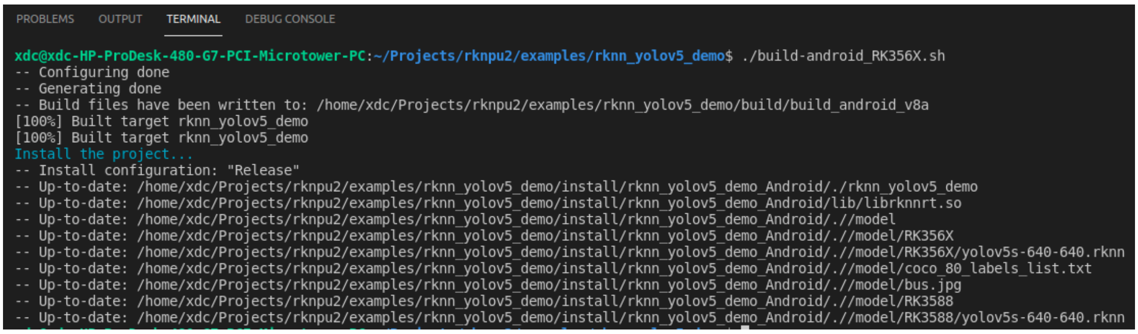


```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2$ cd examples/rknn_yolov5_demo/
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2/examples/rknn_yolov5_demo$ 
```

2) Run the build-android_RK3566_RK3568.sh script to compile the program.

Command:

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



```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2/examples/rknn_yolov5_demo$ ./build-android_RK356X.sh
-- Configuring done
-- Generating done
-- Build files have been written to: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/build/build_android_v8a
[100%] Built target rknn_yolov5_demo
[100%] Built target rknn_yolov5_demo
Install the project...
-- Install configuration: "Release"
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/./rknn_yolov5_demo
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/lib/librknnrt.so
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model/RK356X
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model/RK356X/yolov5s-640-640.rknn
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model/coco_80_labels_list.txt
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model/bus.jpg
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model/RK3588
-- Up-to-date: /home/xdc/Projects/rknpu2/examples/rknn_yolov5_demo/install/rknn_yolov5_demo_Android/.//model/RK3588/yolov5s-640-640.rknn
```

Note:

1) This example compiles RK3566_RK3568 for Android 64-bit platform. If you need to compile other platforms, please select the corresponding script. For details, please refer to /rknpu2/examples/rknn_yolov5_demo/README.md.

2) If a cmake error occurs during compilation, you can execute the following command to install cmake and then run the compilation script.

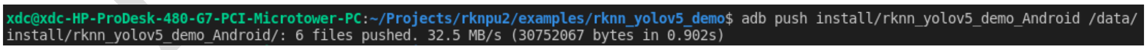
```
sudo apt install cmake
```

3.3.5 Run rknn_yolov5_demo on the board

1) Upload the compiled program and the required files install/rknn_yolov5_demo_Android folder to the /data/ folder of the board.

Command:

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

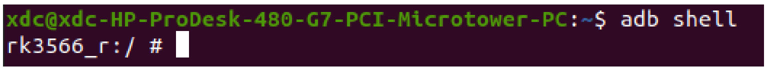


```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~/Projects/rknpu2/examples/rknn_yolov5_demo$ adb push install/rknn_yolov5_demo_Android /data/
install/rknn_yolov5_demo_Android/: 6 files pushed. 32.5 MB/s (30752067 bytes in 0.902s)
```

2) Enter the board system.

Command:

```
adb shell
```

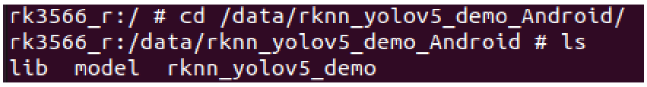


```
xdc@xdc-HP-ProDesk-480-G7-PCI-Microtower-PC:~$ adb shell
rk3566_r:/ #
```

3) cd enter the directory where the program is located.

Command:

```
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) Set library file path.

Command:

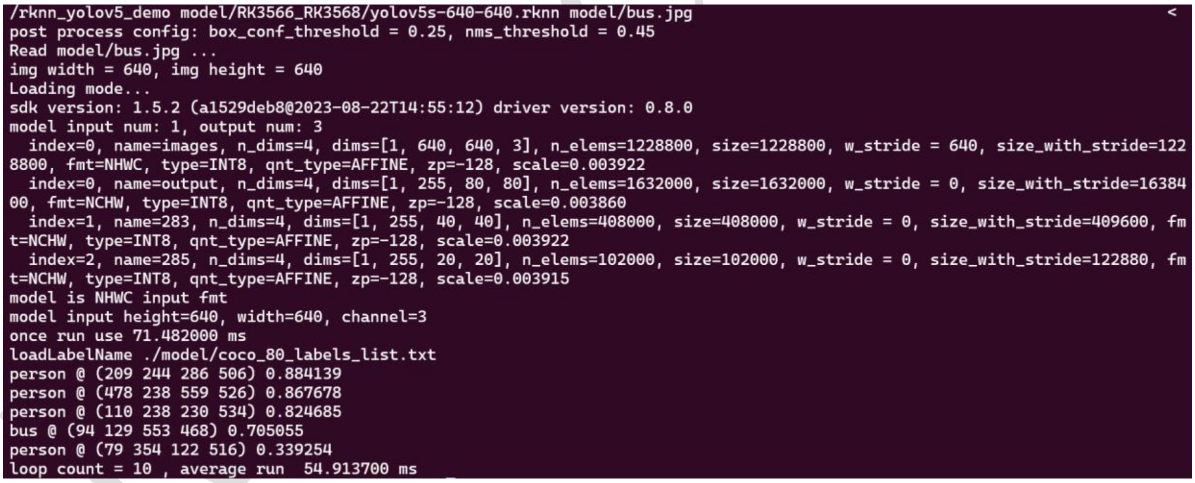
```
export LD_LIBRARY_PATH=./lib
```

5) Run the program to identify the category of the object in the corresponding picture.

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

Command:

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

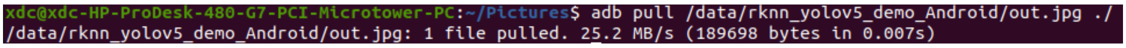


```
/rknn_yolov5_demo model/RK3566_RK3568/yolov5s-640-640.rknn model/bus.jpg                                                       <
post process config: box_conf_threshold = 0.25, nms_threshold = 0.45
Read model/bus.jpg ...
img width = 640, img height = 640
Loading mode...
sdk version: 1.5.2 (a1529deb8@2023-08-22T14:55:12) 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
```

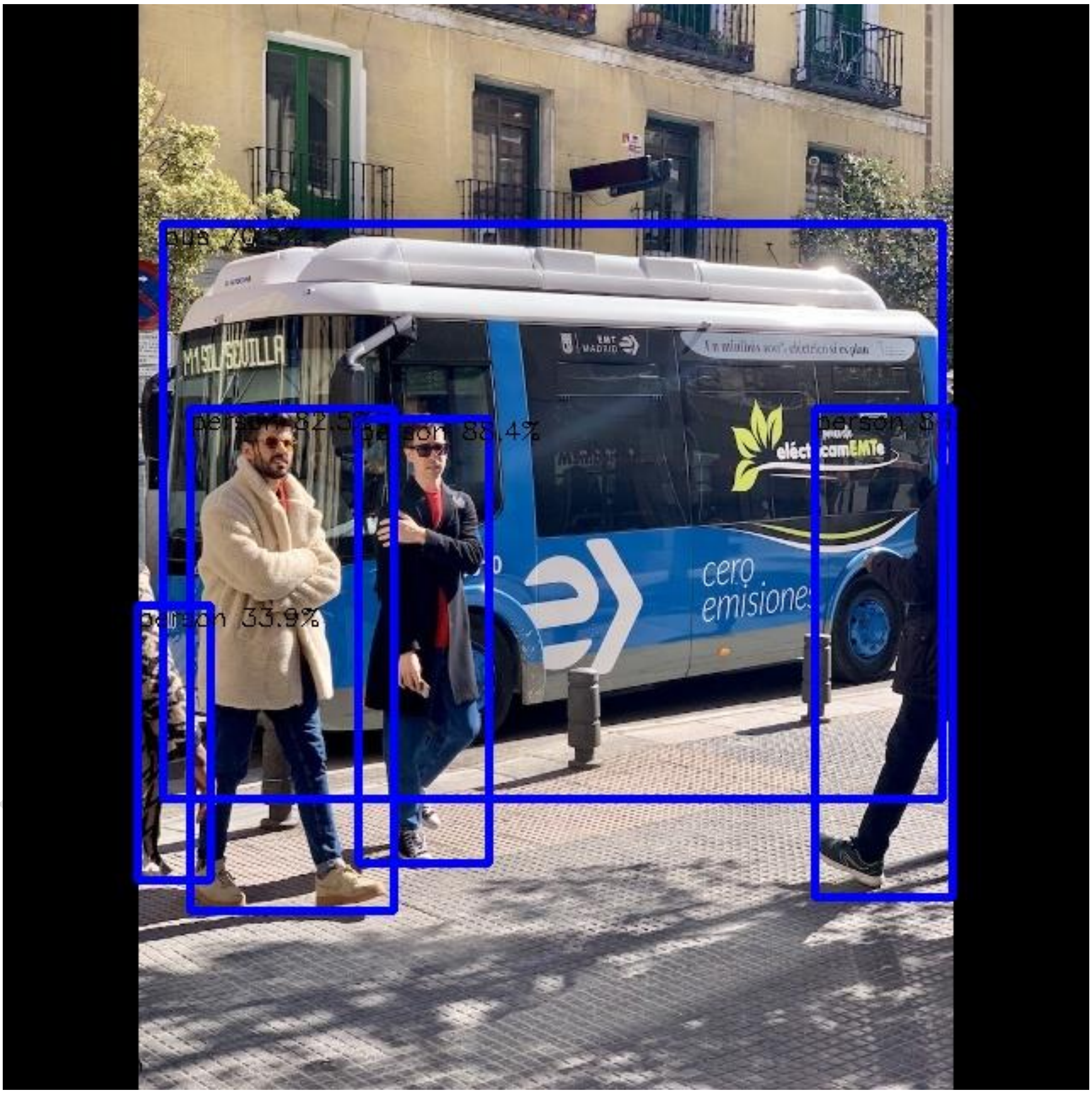
6) Open a new terminal window and download the result image out.jpg to your local computer for viewing.

Command:

```
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 Reference Documents

For more detailed usage and description of RKNN-Toolkit2, please refer to ‘Rockchip_User_Guide_RKNN_Toolkit2_CN.pdf’ manual.

For more detailed usage and description of RKNPU API, please refer to ‘Rockchip_RKNPU_User_Guide_RKNN_API_CN.pdf’ manual.

5 Appendix

5.1 View and set the CPU, DDR and NPU frequency of the development board

Usually, the frequency of each unit on the board is dynamically tuned. In this case, the performance of the tested model will fluctuate. In order to prevent inconsistent performance test results, it is recommended to fix the frequency of the relevant units on the board before testing during performance evaluation. The frequency viewing and setting commands of related units are as follows:

5.1.1 CPU fix frequency command

1) View CPU frequency:

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

2) Fixed CPU frequency:

```
# Check the CPU available frequency (the available frequency displayed by different platforms will be different)
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
408000 600000 816000 1008000 1200000 1416000 1608000 1704000
# Set the CPU frequency, for example, set 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 fix frequency command

1) View DDR frequency (requires firmware support):

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

2) Fixed DDR frequency (requires firmware support):

```
# View DDR available frequencies
cat /sys/class/devfreq/dmc/available_frequencies
# Set the DDR frequency, for example, set 1560MHz
echo userspace > /sys/class/devfreq/dmc/governor
echo 1560000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

5.1.3 NPU fix frequency command

1) View NPU frequency (requires firmware support):

For RK3566_RK3568:

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

For RK3588:

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

For RK3562:

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

2) Fixed NPU frequency (requires firmware support):

For RK3566_RK3568:

```
# View NPU available frequency
cat /sys/class/devfreq/fde40000.npu/available_frequencies
# Set the NPU frequency, for example, set 1 GHz
echo userspace > /sys/class/devfreq/fde40000.npu/governor
echo 1000000000 > /sys/kernel/debug/clk/clk_scmi_npu/clk_rate
```

For RK3588:

```
# Set the NPU frequency, for example, set 1GHz
echo 1000000000 > /sys/kernel/debug/clk/clk_npu_dsu0/clk_rate
```

For RK3562:

```
# View NPU available frequency
cat /sys/class/devfreq/ff300000.npu/available_frequencies
# Set the NPU frequency, for example, set 600MHz
echo userspace > /sys/class/devfreq/ff300000.npu/governor
echo 600000000 > /sys/class/devfreq/ff300000.npu/userspace/set_freq
```

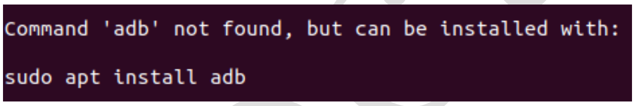
5.2 The command adb devices cannot see the device

1. Check whether the connection is correct, re-plug or replace the USB port of the computer.
2. When using a USB-connected board in a local computer and a docker container, only one end can use the adb server at a time. Therefore, if you cannot see the device when executing the command (adb devices) at one end, you can execute the command at the other command end.

```
adb kill-server
```

Terminate the external adb service, and then return to the original command terminal window to execute the command (adb devices) to view the device.

3. When the following error occurs, adb is not installed. You need to execute the installation command to install adb.



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

Command:

```
sudo apt install adb
```