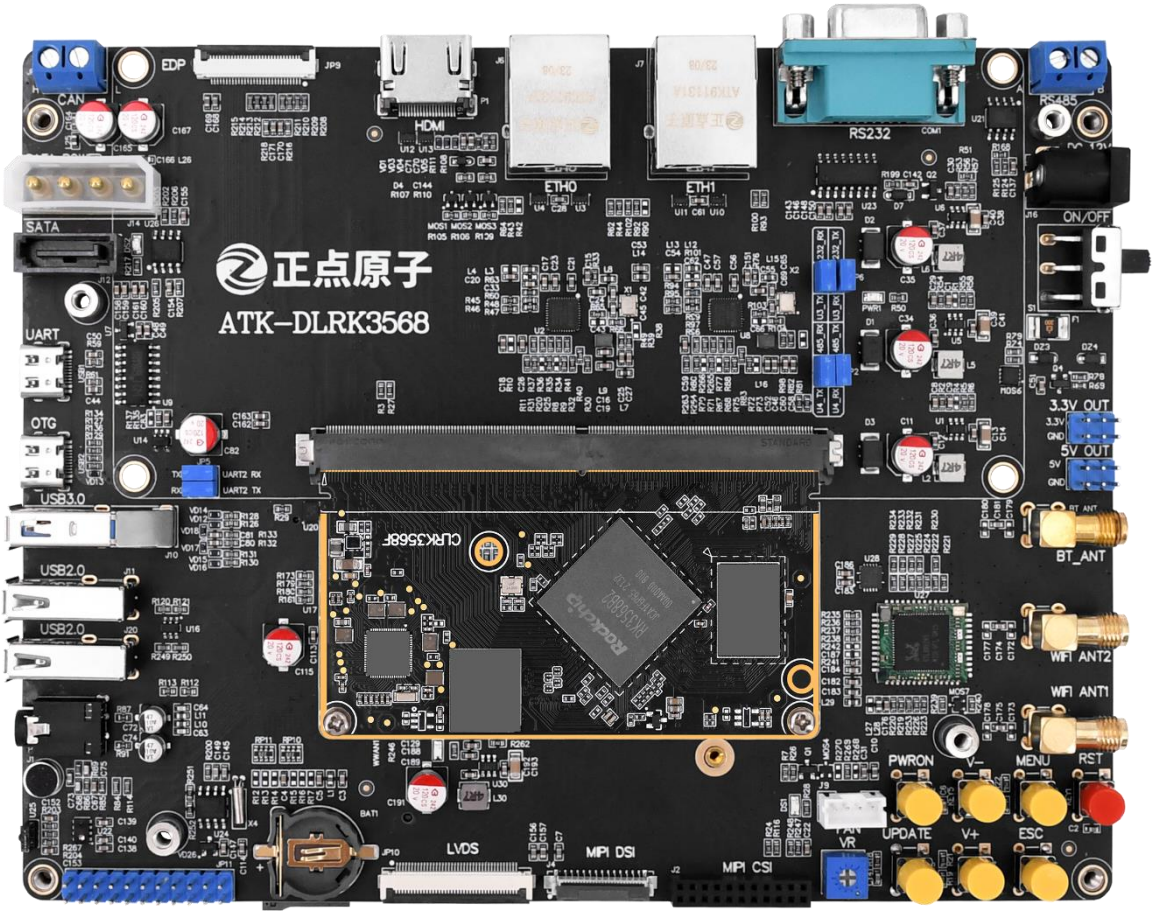
# ATK-DLRK3568

## AI Code Test Manual

## V1.0



ALIENTEK

**1. Shopping:**

TMALL: https://zhengdianyuanzi.tmall.com

TAOBAO: https://openedv.taobao.com

**2. Download**

Address: http://www.openedv.com/docs/index.html

**3. FAE**

Website : www.alientek.com
Forum : http://www.openedv.com/forum.php
Videos : www.yuanzige.com
Fax : +86 - 20 - 36773971
Phone : +86 - 20 - 38271790

# Disclaimer

The product specifications and instructions mentioned in this document are for reference only and subject to update without prior notice; Unless otherwise agreed, this document is intended as a product guide only, and none of the representations made herein constitutes a warranty of any kind. 

In order to get the latest version of product information, please regularly visit the download center or contact the customer service of Taobao ALIENTEK flagship store. Thank you for your tolerance and support.

**Revision History:**

| Version | Version Update Notes | Responsible person | Proofreading | Date |
|---|---|---|---|---|
| V1.0 | release officially | ALIENTEK | ALIENTEK | 2023.08.01 |

# Catalogue

# Chapter 1. RKNN AI Routine Test Development Environment Setup

In this chapter, we will introduce to you how to set up the AI routine test development environment, so that after you get the development board, you can directly compile the AI routine and then conduct tests.

This chapter will be divided into the following sections:

1. Introduction to common IDE tools;
2. Installation of cross-compilation toolchain;
3. Installation and environment configuration of anaconda;

## 1.1 Introduction to Common IDE Tools

You can refer to the tutorial "Development Board CD-ROM A Disk → Basic Materials → 10_user_manual → [ALIENTEK] ATK-DLRK3568 Embedded Linux System Development Manual V1.0" to install the following development environment that is needed. A brief introduction to these tools follows.

**1. VMware Workstations**

This tool is used to install our Ubuntu system. Our software installation packages also provide it. You can install it according to the documentation. Our subsequent development is mostly based on this software installed Ubuntu environment for development.

**2. Ubuntu 20.04**

This is the platform we will use for subsequent development of AI routine tests. The model conversion environment we will encounter later is all configured and converted within this system environment. We strongly recommend that everyone install the Ubuntu 20.04 version as per the instructions in the documentation. It is not recommended to use any other Ubuntu environment. Inconsistent environments may cause different problems. All the documents in this article are conducted on this Ubuntu environment.

**3. Vscode**

A very powerful code editor. We mainly write our code and debug our code in this editor.

**4. Mobaxterm**

A terminal software that integrates multiple functions such as serial port and SSH. It is used to develop and debug our development board through serial port or network connection.

## 1.2 Install cross-compiler toolchain

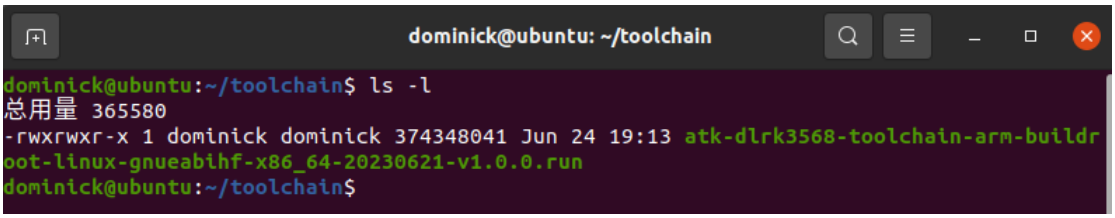
### 1.2.1 Copy cross-compiler toolchain

Before compiling the program, we need to install the cross-compiler toolchain in Ubuntu. This toolchain is compiled through the SDK and includes the libraries required by our AI routine, which can be directly used on the board. Copy the data disk "Development Board CD-ROM A Disk - Basic

Materials → 05_tools → Cross-Compiler Tool → atk-dlrk3568-toolchain-arm-buildroot-linux-gnueabihf-x86_64-20230621-v1.0.0.run" to any directory in Ubuntu (the cross-compiler toolchain will be updated continuously, refer to the actual directory name of the toolchain), as shown in Figure 2.2.1.1.


Figure 1.2.1.1 Cross-compilation toolchain

Here, the author has copied the cross-compiler to the newly created "toolchain" directory in the home directory. One can use the command "ls -l" to check the execution permissions and other details.




If you don't have the execution permission, you can execute the following command to grant the permission.

```
chmod a+x atk-dlrk3568-toolchain-arm-buildroot-linux-gnueabihf-x86_64-20230621-v1.0.0.run
```

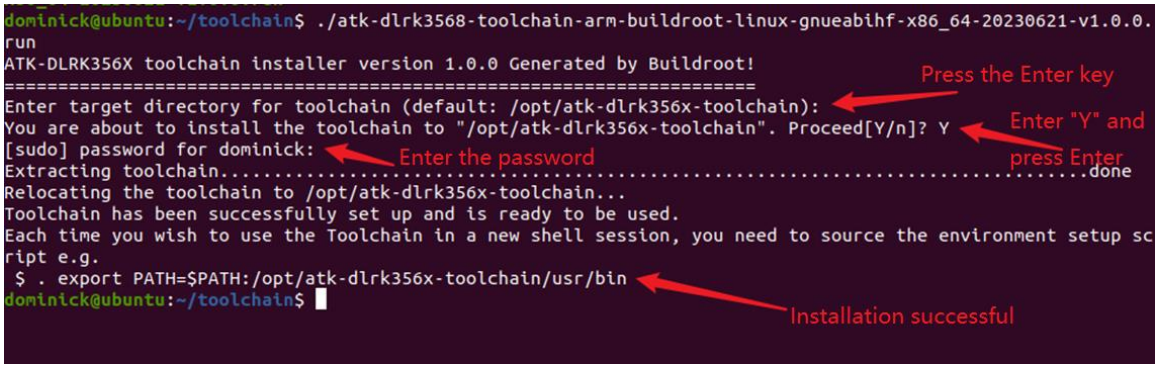
Note: This cross-compilation toolchain will be updated later and it is not the final version. The purpose of the update is to adapt to more routines. If you need to reinstall, simply uninstall and then install the latest version. The installation process is very simple.

### 1.2.2 Installation of Cross-Compilation Toolchain

Open the terminal and execute the following commands to install the compilation toolchain. The installation process is shown in Figure 1.2.2.1.

```
./atk-dlrk3568-toolchain-arm-buildroot-linux-gnueabihf-x86_64-20230621-v1.0.0.run
```

When the prompt "Enter the target directory for the toolchain (default: /opt/atk-dlrk356x-toolchain):" appears, it indicates whether to select the default installation in the /opt/atk-dlrk356x-toolchain directory. It is recommended to directly select the default installation path and simply press the Enter key. When the prompt "You are about to install the toolchain to '/opt/atk-dlrk356x-toolchain'. Proceed [Y/n]?" appears, simply press "Y" and then press Enter. After entering the Ubuntu password and pressing Enter, when the prompt "$. export PATH=$PATH:/opt/atk-dlrk356x-toolchain/usr/bin" appears, it indicates that the installation is complete.

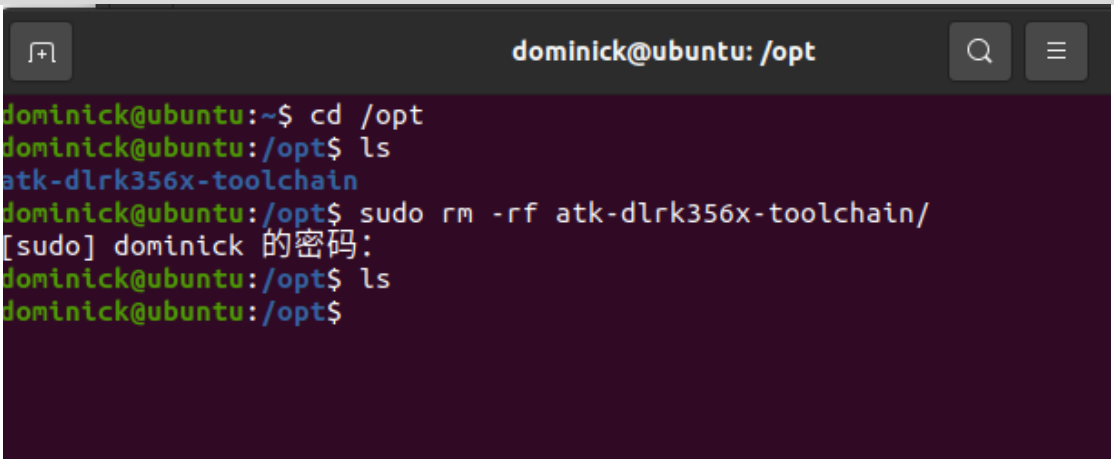



At this point, the cross-compilation toolchain has been installed successfully.

### 1.2.3 Uninstall the cross-compilation toolchain

Open the terminal of Ubuntu and enter the /opt directory. Execute "ls" to see the folder of the installed cross-compilation toolchain. Then, execute the following command to delete it.

```
sudo rm -rf atk-dlrk356x-toolchain/
```



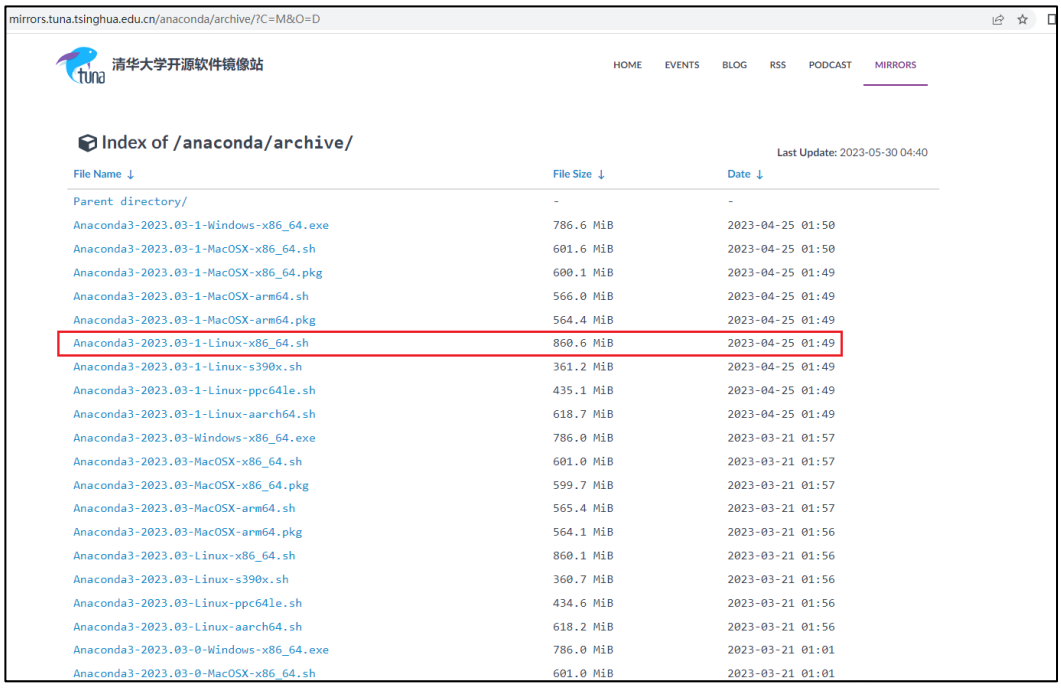

Executing "ls" again will show that this folder is gone, indicating that the cross-compilation toolchain has been uninstalled successfully.

## 1.3 Installation and Environment Configuration of Anaconda

### 1.3.1 Download and Installation of Anaconda

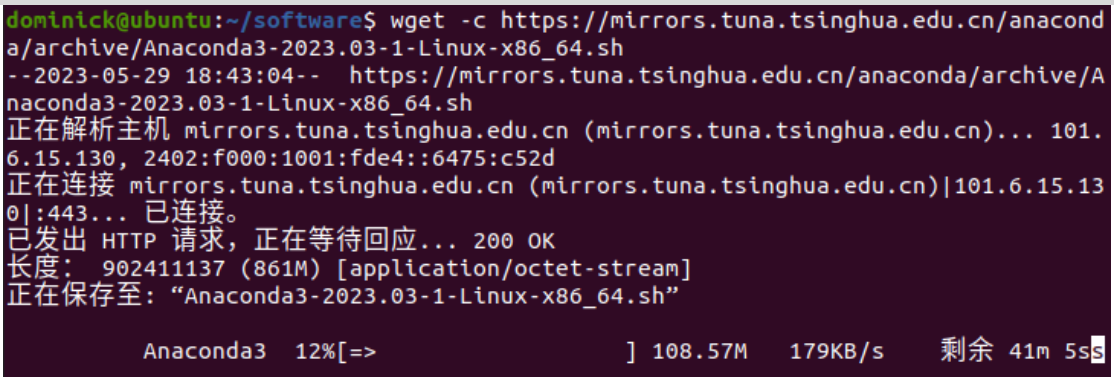
Open the Tsinghua mirror repository to download https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/?C=M&O=D.

Since we choose to perform model conversion and performance evaluation operations in the Ubuntu environment, we select the currently latest Anaconda3-2023.03-1-Linux-x86_64.sh for installation and use. We also provide the A disk of the development board CD - Basic Materials → 04_softwares → Anaconda3-2023.03-1-Linux-x86_64.sh.




Here, you can open the terminal and enter the following commands. First, create a "software" folder to facilitate future use. Then, enter the "software" folder and use wget to download anaconda (here, adding "-c" means it can resume the download in case of interruption).

```
mkdir ~/software
cd ~/software
wget -c https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2023.03-1-Linux-x86_64.sh
```

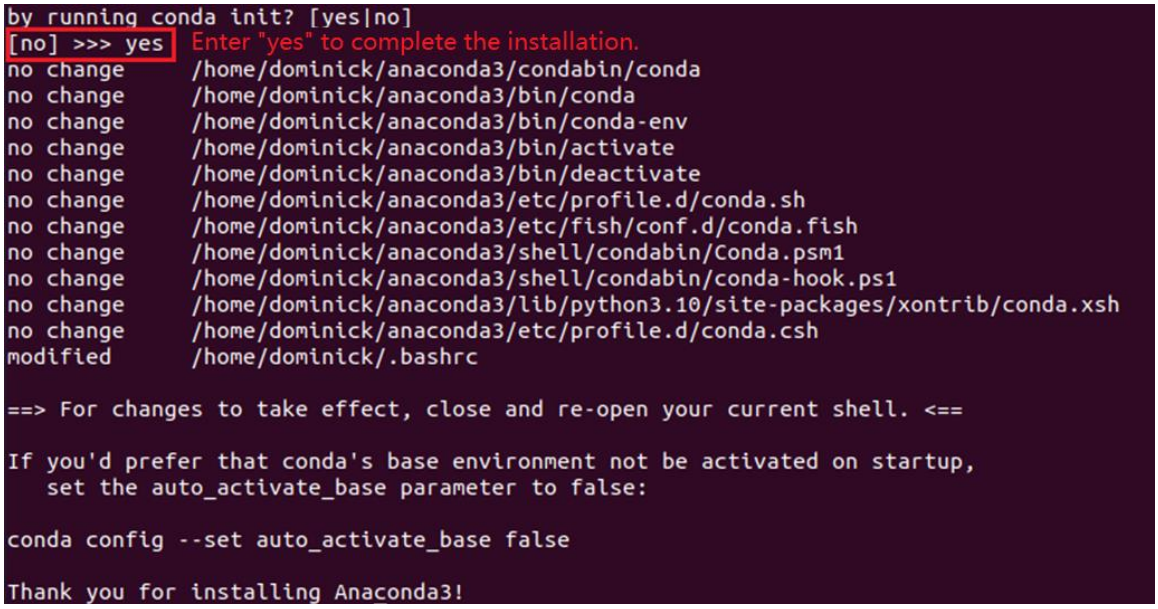



After the download is complete, use the bash command to install it. Then, press Enter once to continue the installation.

```
bash Anaconda3-2023.03-1-Linux-x86_64.sh
```


Press the Enter key again to read the license until the [no] >>> appears. Enter "yes" and press Enter to agree. When the installation path appears, it will be installed by default in the anaconda3 folder under the user directory. You can also specify your own installation path. Here, we will install it directly to the default location. Just press Enter.


Enter "yes" and press Enter again. Then you will see "Thank you for installing Anaconda3!" which means the installation is complete.




After restarting Ubuntu and opening the terminal, you will notice that there is now a (base) prefix. This indicates that you have already entered the conda environment. To exit the conda environment and return to the Ubuntu environment, simply type "conda deactivate".

```
(base) dominick@ubuntu:~/Desktop$ conda deactivate
dominick@ubuntu:~/Desktop$
```

Attention!!! If you always start the computer with the default entry into the conda environment every time, it will be very inconvenient. To prevent confusion with other environments such as the one we use for compiling the rk3568 SDK, we modify the conda environment variables to set it so that it does not automatically enter the conda virtual environment. Execute the following command in the terminal.

```
conda config --set auto_activate_base false
```

```
dominick@ubuntu:~/Desktop$ conda config --set auto_activate_base false
dominick@ubuntu:~/Desktop$
```

After restarting, opening the terminal, it was found that the default conda environment could no longer be accessed.

### 1.3.2 Anaconda Environment Configuration

Configure the download source for the Anaconda environment. Since the official download source is located abroad and may be slow, it is recommended to change the download source to the domestic Anaconda mirror source.

a. First, execute the following command to check the current mirror source. It shows that there is only a "defaults" default source.

```
conda config --show channels
```

```
dominick@ubuntu:~/Desktop$ conda config --show channels
channels:
  - defaults
```

b. Next, we will add the Tsinghua source.

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
```

```
dominick@ubuntu:~/Desktop$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
dominick@ubuntu:~/Desktop$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
dominick@ubuntu:~/Desktop$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/
dominick@ubuntu:~/Desktop$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2/
dominick@ubuntu:~/Desktop$ conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
```

b. Check the configuration

```
conda config --show
```

```
dominick@ubuntu:~/Desktop$ conda config --show
add_anaconda_token: True
add_pip_as_python_dependency: True
aggressive_update_packages:
  - ca-certificates
  - certifi
  - openssl
allow_conda_downgrades: False
allow_cycles: True
allow_non_channel_urls: False
allow_softlinks: False
allowlist_channels: []
always_copy: False
always_softlink: False
always_yes: None
anaconda_upload: None
auto_activate_base: False
auto_stack: 0
auto_update_conda: True
bld_path:
changeps1: True
channel_alias: https://conda.anaconda.org
channel_priority: flexible
channel_settings: []
channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
  - defaults
```

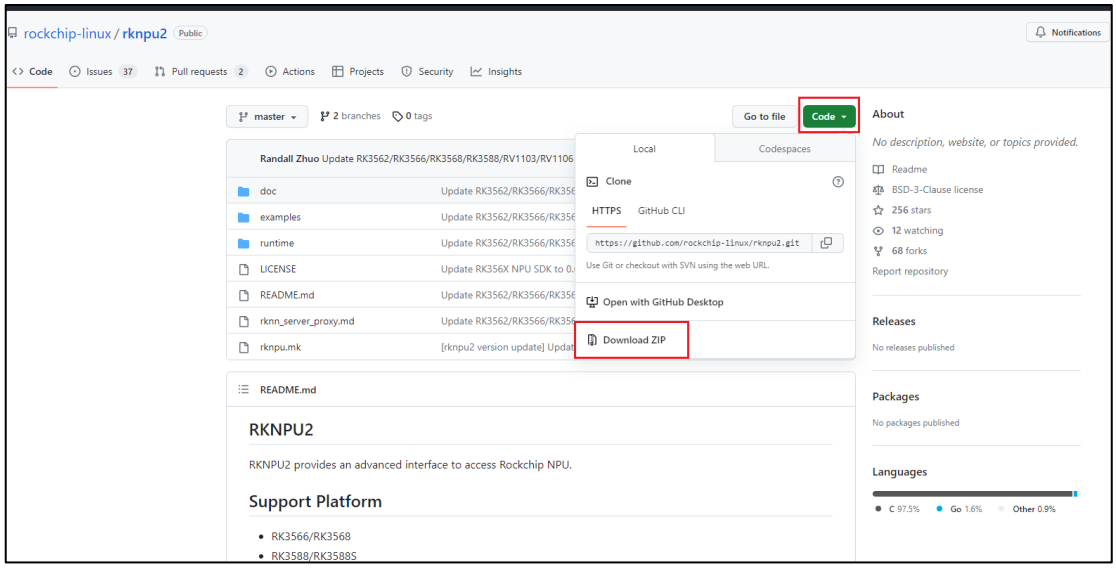
# Chapter 2. Update NPU Driver

RKNN-Toollkit2 is a model conversion tool implemented in Python language, which can convert models exported from other training frameworks into RKNN models and provide relatively limited inference interfaces to assist users in testing the model conversion effect. The website link is: https://github.com/rockchip-linux/rknn-toolkit2. RKNPU2 is a component on the board, providing NPU driver, and providing model loading and model inference functions based on C language. Compared with RKNN-Toolkit2 tool, the C API inference interface of RKNPU2 is more flexible and has better performance. The website link is: https://github.com/rockchip-linux/rknpu2RKNN-Toolkit2 and RKNPU2 have the same version number. There may be incompatibility issues between different versions. To avoid unnecessary troubles, it is recommended that users use the same version of RKNN-Toolkit2 and RKNPU2; Version updates usually include bug fixes and performance optimizations. It is recommended that users use the latest version. This chapter is divided into the following sections:

1. Download the RKNPU2 driver compressed package
2. Update the NPU board connection inference runtime library

## 2.1 Download the rknpu2 driver package

In order to facilitate the subsequent use of OTG and the development board for AI-related development performance tests and board connection inference functions, it is necessary to update the NPU driver of rk3568 (precisely speaking, it is not a driver, but a board-end board connection service. Regarding the rknpu driver, we have updated it to the current latest version 0.8.2). Currently, the NPU driver of rk3568 has been updated to version 1.5.0. Now, let's practice how to update the NPU driver of rk3568. Open the GitHub download websitehttps://github.com/rockchip-linux/rknpu2, use git clone or click the Download ZIP option under the Code section to download the latest rknpu2 package. The one already downloaded on the CD is also available. You can open the A drive of the development board CD - Basic Data → 01_codes → 01_AI_Examples → 03, Software and Drivers, and then find the rknpu-1.5.0.zip.

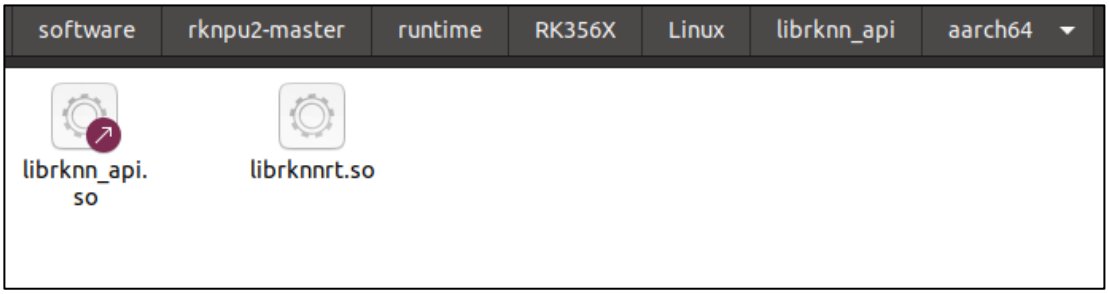



Copy and unzip the compressed file to the "software" directory under the Linux system. We can refer to the "rknn_server_proxy.md" file in the rknpu2 driver folder to update the driver.

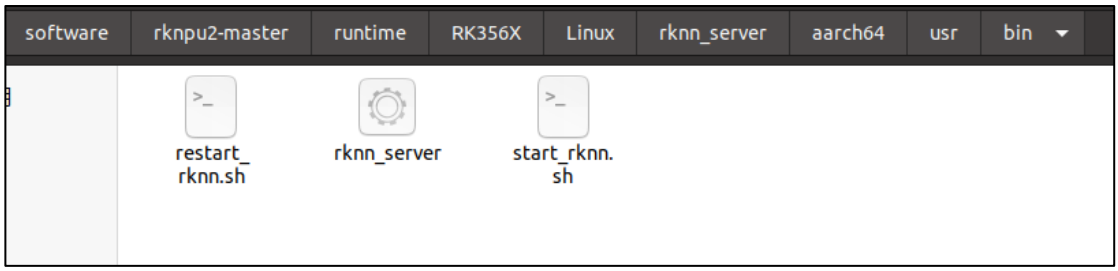
## 2.2 Update the NPU integrated inference runtime library

We will transfer the files in the "runtime" directory of the downloaded "rknpu2" folder to the development board, mainly including the following two directories' files.

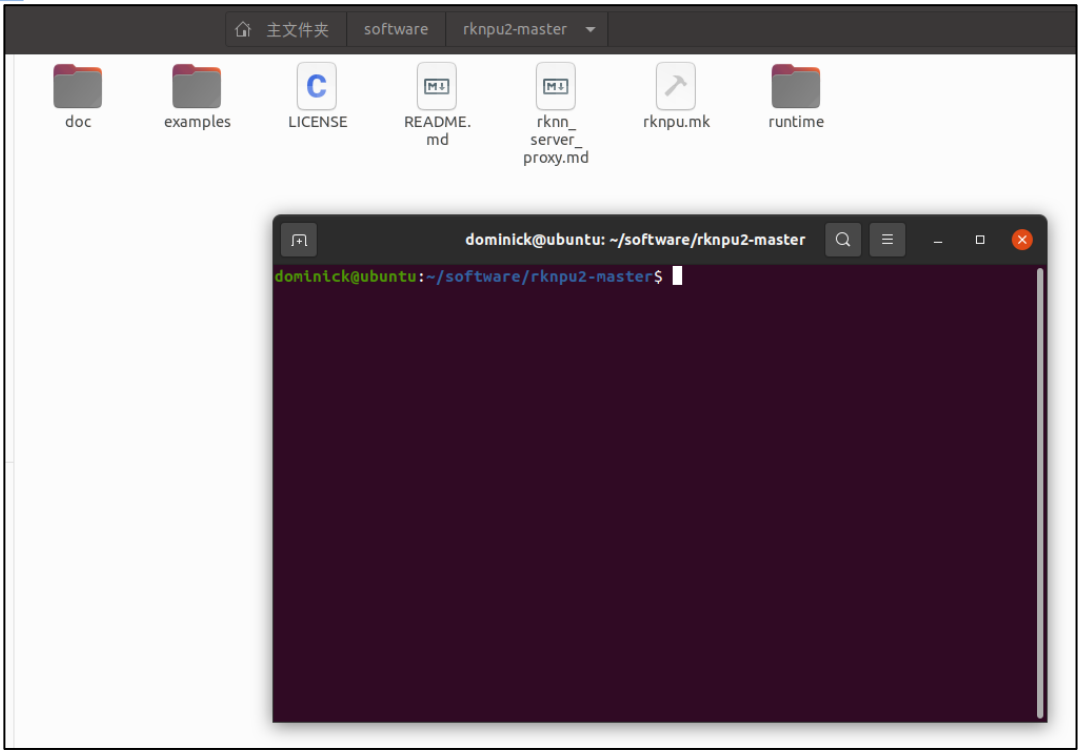
One is the "runtime/RK356X/Linux/librknn_api/aarch64/" directory.




The other one is the directory located at runtime/RK356X/Linux/rknn_server/aarch64/usr/bin/.




Before updating the NPU driver service, you need to first connect the power supply of the development board and connect the OTG of the development board to the computer via a Type-C cable. Then, enter the NPU driver directory, open the terminal, and use adb for data transfer.




Execute the following commands one by one in the Ubuntu terminal to transfer the files to the development board.

```
adb push runtime/RK356X/Linux/librknn_api/aarch64/librknnrt.so /usr/lib
```

```
dominick@ubuntu:~/software/rknpu2-master$ adb push runtime/RK356X/Linux/librknn_api/aarch64/li
brknnrt.so /usr/lib
runtime/RK356X/Linux/librknn_api/aarch...ed. 1.1 MB/s (4790184 bytes in 4.174s)
```

```
adb push runtime/RK356X/Linux/rknn_server/aarch64/usr/bin/* /usr/bin
```

```
dominick@ubuntu:~/software/rknpu2-master$ adb push runtime/RK356X/Linux/rknn_server/aarch64/us
r/bin/* /usr/bin
runtime/RK356X/Linux/rknn_server/aarch64/usr/...1 file pushed. 0.0 MB/s (252 bytes in 0.020s)
runtime/RK356X/Linux/rknn_server/aarch64/usr/...ile pushed. 1.0 MB/s (422448 bytes in 0.402s)
runtime/RK356X/Linux/rknn_server/aarch64/usr/... 1 file pushed. 0.0 MB/s (71 bytes in 0.009s)
3 files pushed. 0.9 MB/s (422771 bytes in 0.444s)
```

Enter the shell environment of the development board in the Ubuntu terminal, grant execution permissions, and restart the service.

```
adb shell
chmod +x /usr/bin/rknn_server
chmod +x /usr/bin/start_rknn.sh
chmod +x /usr/bin/restart_rknn.sh
restart_rknn.sh
```

```
dominick@ubuntu:~/software/rknpu2-master$ adb shell
root@RK356X:/# chmod +x /usr/bin/rknn_server
root@RK356X:/# chmod +x /usr/bin/start_rknn.sh
root@RK356X:/# chmod +x /usr/bin/restart_rknn.sh
root@RK356X:/# restart_rknn.sh
```

After executing the "restart_rknn.sh" script, the version number will be printed after the service is restarted.

```
root@RK356X:/# start rknn server, version:1.5.0 (17e11b1 build: 2023-05-18 21:43:21)
I NPUTransfer: Starting NPU Transfer Server, Transfer version 2.1.0 (b5861e7@2020-11-23T11:50:
51)
```

Execute the following command to query the version of rknn_server.

```
strings /usr/bin/rknn_server | grep build
```

```
root@RK356X:/# strings /usr/bin/rknn_server | grep build
1.5.0 (17e11b1 build: 2023-05-18 21:43:21)
.note.gnu.build-id
```

Execute the following command to query the version of librknnrt.so.

```
strings /usr/lib/librknnrt.so | grep version
```

```
root@RK356X:/# strings /usr/lib/librknnrt.so | grep version
librknnrt version: 1.5.0 (e6fe0c678@2023-05-25T08:09:20)
rknn_query, info_len(%d) < sizeof(rknn_sdk_version)(%d)!
Invalid RKNN model version
RKNN Model Information: version: %d, toolkit version: %s, target: %s, target platform: %s, framework name: %s, framework layout:
%s, model inference type: %s
RKNN Model version: %d.%d.%d not match with rknn runtime version: %d.%d.%d
version
(compiler version:
Current driver version: %d.%d.%d, recommend to upgrade the driver to the new version: >= %d.%d.%d
This shared library is for RK356X/RK3588 platforms only!, hw_version = %d
RKNN Driver Information: version: %d.%d.%d
Mismatch driver version, %s requires driver version >= %d.%d.%d, but you have driver version: %d.%d.%d which is incompatible!
please try updating to the latest version of the toolkit2 and runtime from: https://eyun.baidu.com/s/3eTDMk6Y (PWD: rknn)
wrong version
failed to check rknpu hardware version: %#x
Illegal core num, rknn model version illegal
.gnu.version
```

Input "exit" to exit the development board system and return to the Ubuntu terminal.

# Chapter 3. Install the rknn-toolkit2 conversion environment

rknn-toolkit2 is a tool used to convert traditional models in formats such as TensorFlow, Pytorch, onnx, etc. into the rknn format suitable for devices like rk3568/rk3566.

This chapter will be divided into the following sections:

1. Download rknn-toolkit2
2. Create a new conda environment
3. Install rknn-toolkit2

## 3.1 Download rknn-toolkit2

First, open the website https://github.com/rockchip-linux/rknn-toolkit2. Click on the blue text (the page layout may change later, you can download the one we have already downloaded. The path is the development board CD's A drive - Basic Materials → 01_codes → 01_AI_Examples → 03, Software and Drivers → rknn-toolkit2-1.5.0.zip). Enter the password "rknn" to access the corresponding page.

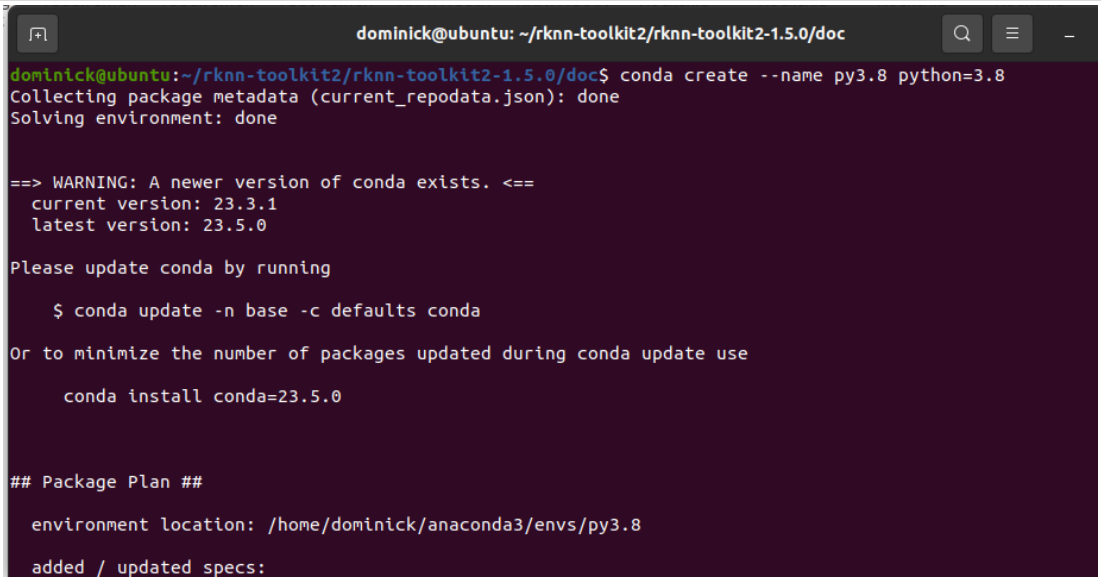

Select RK_NPU_SDK ---> RK_NPU_SDK_1.5.0. Currently, the latest version is 1.5.0. Here, there will be two versions: one is the develop version (the version under development), and the other is the release version (the released version). We choose the release version because it is more stable. Select all the checkboxes, click on download, and it will open the download from Baidu Cloud (if it is not installed, you need to install the latest version of Baidu Cloud. If it still doesn't work, save it to Baidu Cloud and then download it).


## 3.2 Create a new Conda environment

Previously, we downloaded Anaconda. To prevent confusion in the subsequent compilation environment, this comes in handy here. First, we create a new Conda virtual environment and then

install the rknn-toolkit2. Open the terminal and execute the following command: Create a Conda environment named "py3.8" with a Python version of Python 3.8.

```
conda create --name py3.8 python=3.8
```




The program stops at "Proceed". Press "y" and then press the Enter key.

```
  _libgcc_mutex      anaconda/cloud/conda-forge/linux-64::_libgcc_mutex-0.1-conda_forge
  _openmp_mutex      anaconda/cloud/conda-forge/linux-64::_openmp_mutex-4.5-2_gnu
  bzip2              anaconda/cloud/conda-forge/linux-64::bzip2-1.0.8-h7f98852_4
  ca-certificates    anaconda/cloud/conda-forge/linux-64::ca-certificates-2023.5.7-hbcca054_0
  ld_impl_linux-64   anaconda/cloud/conda-forge/linux-64::ld_impl_linux-64-2.40-h41732ed_0
  libffi             anaconda/cloud/conda-forge/linux-64::libffi-3.4.2-h7f98852_5
  libgcc-ng          anaconda/cloud/conda-forge/linux-64::libgcc-ng-12.2.0-h65d4601_19
  libgomp            anaconda/cloud/conda-forge/linux-64::libgomp-12.2.0-h65d4601_19
  libnsl             anaconda/cloud/conda-forge/linux-64::libnsl-2.0.0-h7f98852_0
  libsqlite          anaconda/cloud/conda-forge/linux-64::libsqlite-3.42.0-h2797004_0
  libuuid            anaconda/cloud/conda-forge/linux-64::libuuid-2.38.1-h0b41bf4_0
  libzlib            anaconda/cloud/conda-forge/linux-64::libzlib-1.2.13-h166bdaf_4
  ncurses            anaconda/cloud/conda-forge/linux-64::ncurses-6.3-h27087fc_1
  openssl            anaconda/cloud/conda-forge/linux-64::openssl-3.1.1-hd590300_1
  pip                anaconda/cloud/conda-forge/noarch::pip-23.1.2-pyhd8ed1ab_0
  python             anaconda/cloud/conda-forge/linux-64::python-3.8.16-he550d4f_1_cpython
  readline           anaconda/cloud/conda-forge/linux-64::readline-8.2-h8228510_1
  setuptools         anaconda/cloud/conda-forge/noarch::setuptools-67.7.2-pyhd8ed1ab_0
  tk                 anaconda/cloud/conda-forge/linux-64::tk-8.6.12-h27826a3_0
  wheel              anaconda/cloud/conda-forge/noarch::wheel-0.40.0-pyhd8ed1ab_0
  xz                 anaconda/cloud/conda-forge/linux-64::xz-5.2.6-h166bdaf_0


Proceed ([y]/n)? y


Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate py3.8
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

At this point, the new conda environment has been successfully created.

## 3.3 Install rknn-toolkit2

### 3.3.1 Install dependencies for rknn-toolkit2

Enter the rknn-toolkit2-1.5.0/doc folder, open the terminal and execute the following command to activate the conda environment.

```
conda activate py3.8
```

If "(py3.8)" appears before the terminal command line, it indicates that the newly created conda environment named "py3.8" has been successfully entered.

```
dominick@ubuntu:~/rknn-toolkit2/rknn-toolkit2-1.5.0/doc$ conda activate py3.8
(py3.8) dominick@ubuntu:~/rknn-toolkit2/rknn-toolkit2-1.5.0/doc$
```

Before installing rknn-toolkit2, you need to install the required dependencies first. Before executing the following commands, you need to check if you are already in the "doc" folder. If not, you need to enter the "doc" folder first.

```
pip install -r requirements_cp38-1.5.0.txt -i https://mirror.baidu.com/pypi/simple
```

```
(py3.8) dominick@ubuntu:~/rknn-toolkit2/rknn-toolkit2-1.5.0/doc$ pip install -r requirements_cp38-1.5.0.txt -i https://mirror.baidu.com/pypi/simple
Looking in indexes: https://mirror.baidu.com/pypi/simple
Collecting numpy==1.19.5 (from -r requirements_cp38-1.5.0.txt (line 4))
  Downloading https://mirror.baidu.com/pypi/packages/66/d7/3b133b17e185f14137bc8afe7a41daf1f31556900f10238312a5ae9c7345/numpy-1.19.5-cp38-cp38-manylinux2010_x86_64.whl (14.9 MB)
     |████████████████████████████████| 14.9/14.9 MB 10.3 MB/s eta 0:00:00
Collecting protobuf==3.12.2 (from -r requirements_cp38-1.5.0.txt (line 5))
  Downloading https://mirror.baidu.com/pypi/packages/8f/ce/1402286175ca38a02f5bfb8f01c10beaa062bbff60aca9c39cae9818ba41/protobuf-3.12.2-cp38-cp38-manylinux1_x86_64.whl (1.3 MB)
     |████████████████████████████████| 1.3/1.3 MB 8.3 MB/s eta 0:00:00
Collecting flatbuffers==1.12 (from -r requirements_cp38-1.5.0.txt (line 6))
  Downloading https://mirror.baidu.com/pypi/packages/eb/26/712e578c5f14e26ae3314c39a1bdc4eb2ec2f4ddc89b708cf8e0a0d20423/flatbuffers-1.12-py2.py3-none-any.whl (15 kB)
Collecting requests==2.27.1 (from -r requirements_cp38-1.5.0.txt (line 9))
  Downloading https://mirror.baidu.com/pypi/packages/2d/61/08076519c80041bc0ffa1a8af0cbd3bf3e2b62af10435d269a9d0f40564d/requests-2.27.1-py2.py3-none-any.whl (63 kB)
     |████████████████████████████████| 63.1/63.1 kB 9.5 MB/s eta 0:00:00
Collecting psutil==5.9.0 (from -r requirements_cp38-1.5.0.txt (line 10))
  Downloading https://mirror.baidu.com/pypi/packages/0a/66/b2188d8e738ee52206a4ee804907f6eab5bcc9fc0e8486e7ab973a8323b7/psutil-5.9.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (283 kB)
     |████████████████████████████████| 283.8/283.8 kB 2.6 MB/s eta 0:00:00
Collecting ruamel.yaml==0.17.4 (from -r requirements_cp38-1.5.0.txt (line 11))
  Downloading https://mirror.baidu.com/pypi/packages/29/4e/c3105bbbbc662f6a671a505f00ec771e93b5254f09fbb06002af9087071a/ruamel.yaml-0.17.4-py3-none-any.whl (101 kB)
     |████████████████████████████████| 101.9/101.9 kB 3.6 MB/s eta 0:00:00
Collecting scipy==1.5.4 (from -r requirements_cp38-1.5.0.txt (line 12))
  Downloading https://mirror.baidu.com/pypi/packages/ab/a3/4e10c6091f6b17267e23a6de77c05241834fb3a1ec6ad655b566a6ea1d82/scipy-1.5.4-cp38-cp38-manylinux1_x86_64.whl (25.8 MB)
     |████████████████████████████████| 25.8/25.8 MB 10.1 MB/s eta 0:00:00
Collecting tqdm==4.64.0 (from -r requirements_cp38-1.5.0.txt (line 13))
  Downloading https://mirror.baidu.com/pypi/packages/8a/c4/d15f1e627fff25443ded77ea70a7b5532d6371498f9285d44d62587e209c/tqdm-4.64.0-py2.py3-none-any.whl (78 kB)
     |████████████████████████████████| 78.4/78.4 kB 3.5 MB/s eta 0:00:00
```

The -i https://mirror.baidu.com/pypi/simple parameter mentioned here means that the source is set to pip and the mirror source is https://mirror.baidu.com/pypi/simple. By opening the requirements_cp38-1.5.0.txt file in the doc directory, you can see that this mirror source is the recommended download source by Rockchip Microelectronics. Therefore, we try to follow the official operation procedures.


When the following message "Successfully installed" (meaning "Installation successful") is displayed, it indicates that the related dependent libraries have been installed successfully.

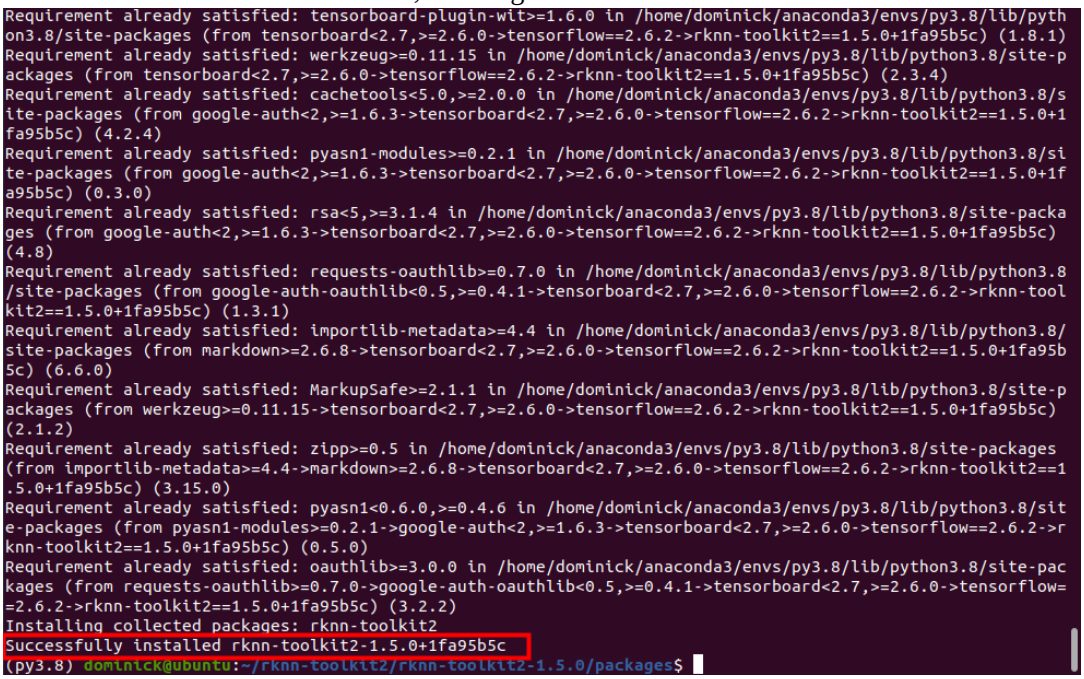
```
Successfully installed MarkupSafe-2.1.2 absl-py-0.15.0 astunparse-1.6.3 cachetools-4.2.4 certifi-2023.5.7 charset-normalizer-2.0.12 clang-5.0 fast-histogram-0.11 flatbuffers-1.12 gast-0.4.0 google-auth-1.35.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.55.0 h5py-3.1.0 idna-3.4 importlib-metadata-6.6.0 keras-2.6.0 keras-preprocessing-1.1.2 markdown-3.4.3 numpy-1.19.5 oauthlib-3.2.2 onnx-1.10.0 onnxoptimizer-0.2.7 onnxruntime-1.10.0 opencv-python-4.5.5.64 opt-einsum-3.3.0 pillow-9.5.0 protobuf-3.12.2 psutil-5.9.0 pyasn1-0.5.0 pyasn1-modules-0.3.0 requests-2.27.1 requests-oauthlib-1.3.1 rsa-4.8 ruamel.yaml-0.17.4 ruamel.yaml.clib-0.2.7 scipy-1.5.4 six-1.15.0 tensorboard-2.6.0 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.6.2 tensorflow-estimator-2.6.0 termcolor-1.1.0 torch-1.10.1 torchvision-0.11.2 tqdm-4.64.0 typing-extensions-3.7.4.3 urllib3-1.26.15 werkzeug-2.3.4 wrapt-1.12.1 zipp-3.15.0
(py3.8) dominick@ubuntu:~/rknn-toolkit2/rknn-toolkit2-1.5.0/doc$
```

### 3.3.2 Install the rknn-toolkit2 tool

Stay in the py3.8 conda environment and enter the rknn-toolkit2/packages directory, then execute the following command.

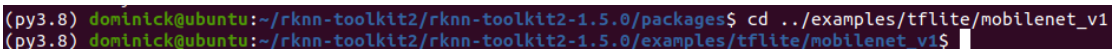
```
pip install rknn_toolkit2-1.5.0+1fa95b5c-cp38-cp38-linux_x86_64.whl
```

After the installation is completed, a red box will appear in the last line indicating "Successfully installed rknn-toolkit2-1.5.0+1fa95b5c", meaning the installation was successful.




### 3.3.3 Test rknn-toolkit2

We enter the corresponding tflite routine directory, and test if the mobilenet_v1 routine is functioning properly. If it is normal, then it indicates that the rknn-toolkit2 installation is successful. Execute the following command.




```
python test.py
```

```
D RKNN: [18:37:18.852] -------------------------------------------------------------------------------
-------------------------------+-------------------------------
D RKNN: [18:37:18.852] ------------------------------------
D RKNN: [18:37:18.852] Total Weight Memory Size: 4365632
D RKNN: [18:37:18.852] Total Internal Memory Size: 1756160
D RKNN: [18:37:18.852] Predict Internal Memory RW Amount: 10362496
D RKNN: [18:37:18.852] Predict Weight Memory RW Amount: 4365552
D RKNN: [18:37:18.852] ------------------------------------
D RKNN: [18:37:18.852] <<<<<<<< end: N4rknn21RKNNMemStatisticsPassE
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%|                                                | 60/60 [00:00<00:00, 5959.93it/
s]
Preparing : 100%|                                                | 60/60 [00:00<00:00, 411.44it/
s]
mobilenet_v1
-----TOP 5-----
[156]: 0.9345703125
[155]: 0.0570068359375
[205]: 0.00429534912109375
[284]: 0.003116607666015625
[285]: 0.00017178058624267578

done
```

The top-5 classification results of the model can be printed out and the test is normal.

# Chapter 4. Check and adjust the CPU and NPU frequencies

Pulse frequency is used to automatically adjust the working frequency based on system requirements and load conditions. This adjustment can be switched between different frequency modes to balance performance and energy efficiency. To facilitate everyone's better utilization of the RK3568 development board to achieve the best performance, Rockchip provides a series of commands for checking and setting the pulse frequency. Next, we will introduce the methods for checking and setting the pulse frequency of the CPU and NPU one by one.

This chapter will be divided into the following sections:

1. Check if debugfs is mounted
2. Check and fix the CPU frequency
3. Check the NPU occupancy rate
4. Check the NPU frequency and NPU pulse frequency setting
5. Quick pulse frequency setting

## 4.1 Check if debugfs is mounted

Before checking the CPU/GPU/NPU frequencies and usage rates, it is necessary to check if debugfs is mounted. You can use the following command to check, and you will see that the ALIENTEK factory system defaults to mounting debugfs.

```
mount | grep debug
```

```
root@RK356X:/# mount | grep debug
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
```

## 4.2 View and fix CPU frequency

### 4.2.1 View the current running frequency of the CPU

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

```
root@RK356X:/# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
1416000
```

### 4.2.2 Fixed CPU frequency

**1. Check the available CPU frequencies**

```
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
```

408000 600000 816000 1104000 1416000 1608000 1800000 1992000

```
quencies56X:/# cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_freq
408000 600000 816000 1104000 1416000 1608000 1800000 1992000
```

**2. Set CPU frequency**

Here, we attempt to set the main frequency to 1.99GHz.

```
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
echo 1992000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
```

After the settings are completed, execute the following command to check the current CPU frequency.

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

```
root@RK356X:/# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
1992000
```

## 4.3 Check NPU occupancy rate

During the operation, use the following command to check the current occupancy rate of the NPU.

```
cat /sys/kernel/debug/rknpu/load
```

```
root@RK356X:/# cat /sys/kernel/debug/rknpu/load
NPU load:  0%
```

## 4.4 Check NPU frequency and NPU fixed frequency

### 4.4.1 Check NPU frequency

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

```
root@RK356X:/# cat /sys/kernel/debug/clk/clk_summary | grep npu
clk_scmi_npu                    0        1        0   600000000          0     0  50000
   clk_npu_pvtm                 0        0        0    24000000          0     0  50000
      clk_npu_np5               0        0        0   342857143          0     0  50000
      clk_npu_src               0        1        0   200000000          0     0  50000
         clk_npu_pre_ndft       0        1        0   200000000          0     0  50000
            clk_npu_pvtpll      0        0        0   200000000          0     0  50000
            clk_npu_pvtm_core       0        0        0   200000000          0     0  50000
            clk_npu             0        4        0   200000000          0     0  50000
               aclk_npu_pre     0        2        0   200000000          0     0  50000
                  aclk_npu      0        3        0   200000000          0     0  50000
               pclk_npu_pre     0        1        0    33333334          0     0  50000
                  pclk_npu_pvtm       0        0        0    33333334          0     0  50000
               hclk_npu_pre     0        2        0    50000000          0     0  50000
                  hclk_npu      0        3        0    50000000          0     0  50000
```

Or execute the following command.

```
cat /sys/class/devfreq/fde40000.npu/cur_freq
```

```
root@RK356X:/# cat /sys/kernel/debug/clk/clk_scmi_npu/clk_rate
600000000
```

### 4.4.2 Fixed-frequency NPU

**1. View the available frequencies of the NPU**

```
cat /sys/class/devfreq/fde40000.npu/available_frequencies
```

200000000 297000000 400000000 600000000 700000000 800000000 900000000

```
root@RK356X:/# cat /sys/class/devfreq/fde40000.npu/available_frequencies
200000000 297000000 400000000 600000000 700000000 800000000 900000000
```

**2. Set the NPU frequency**

Before setting, you need to turn on the power supply first.

```
echo userspace > /sys/class/devfreq/fde40000.npu/governor
echo 900000000 > /sys/class/devfreq/fde40000.npu/userspace/set_freq
```

```
root@ATK-DLRK356X:/# echo userspace > /sys/class/devfreq/fde40000.npu/governor
root@ATK-DLRK356X:/# echo 900000000 > /sys/class/devfreq/fde40000.npu/userspace/set_freq
root@ATK-DLRK356X:/# cat /sys/kernel/debug/clk/clk_summary | grep npu
 clk_scmi_npu                        0        1        0   900000000          0     0  50000
    clk_npu_pvtm                     0        0        0    24000000          0     0  50000
       clk_npu_np5                   0        0        0   342857143          0     0  50000
       clk_npu_src                   0        1        0   200000000          0     0  50000
          clk_npu_pre_ndft           0        1        0   200000000          0     0  50000
             clk_npu_pvtpll          0        0        0   200000000          0     0  50000
             clk_npu_pvtm_core       0        0        0   200000000          0     0  50000
             clk_npu                 0        4        0   200000000          0     0  50000
                aclk_npu_pre         0        2        0   200000000          0     0  50000
                   aclk_npu          0        3        0   200000000          0     0  50000
                pclk_npu_pre         0        1        0    33333334          0     0  50000
                   pclk_npu_pvtm     0        0        0    33333334          0     0  50000
                hclk_npu_pre         0        2        0    50000000          0     0  50000
                   hclk_npu          0        3        0    50000000          0     0  50000
root@ATK-DLRK356X:/# cat /sys/class/devfreq/fde40000.npu/cur_freq
900000000
```

## 4.5 Fast Frequency Setting

To facilitate everyone in quickly developing the project and achieving better results, we have provided a frequency setting script. The path is: Development Board CD-ROM A Disk - Basic Data → 01_codes → 01_AI_Routine → 01, Source Code → 00, Frequency Setting Script → performance.sh. Through SSH or ADB, push it to any path on the development board. Here, we push it to the /userdata directory on the development board. Open the serial port terminal on the board and enter the directory where the script is located to execute the following commands.

```
chmod +x performance.sh
./performance.sh
```

The execution result is shown in the following figure. It can be seen that the CPU frequency and the NPU frequency have been successfully set to the highest performance level.

```
root@ATK-DLRK356X:/userdata# chmod +x performance.sh
root@ATK-DLRK356X:/userdata# ./performance.sh
CPU0-3 可用频率:
408000 600000 816000 1104000 1416000 1608000 1800000 1992000
CPU0-3 当前频率:
1992000
NPU 可用频率:
200000000 297000000 400000000 600000000 700000000 800000000 900000000
[10679.006966] RKNPU fde40000.npu: RKNPU: set rknpu freq: 900000000, NPU 当前频率:
volt: 925000
900000000
```

# Chapter 5. Test the Python inference routine under buildroot

The root file system of the factory system of the Linux system of the ALIENTEK RK3568 is built by buildroot, so the support for Python is not so good. However, ALIENTEK has adapted the main related libraries of Python inference, so some of the provided routines can be executed for inference through Python. Now, let's introduce how to make our Python routines run.

This chapter will be divided into the following sections:

1. Install RKNN Toolkit Lite2
2. Test the AI routines under Python
3. Test the self-trained lenet model under Python

## 5.1 Install RKNN Toolkit Lite2

RKNN Toolkit Lite2 provides Python programming interfaces for the Rockchip NPU platform, helping users deploy RKNN models and accelerate the implementation of AI applications. Next, we will explain how to install the rknn-toolkit-lite2 environment on the board.

Open the serial terminal, enter the Python command and press Enter to find that the pre-installed Python version on our development board is 3.8.6. In the development board terminal, enter exit() and press Enter to exit.

```
root@ATK-DLRK356X:/# python
Python 3.8.6 (default, Jul 28 2023, 19:28:33)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Open the rknn-toolkit2-1.5.0 folder that we decompressed in the previous chapter. Go to the rknn_toolkit_lite2/packages directory within rknn-toolkit2-1.5.0. Open the Ubuntu terminal in the current directory and connect the development board to the OTG interface. Then, enter the following command to transfer the whl file to the userdata directory on the development board.

```
adb push rknn_toolkit_lite2-1.5.0-cp38-cp38-linux_aarch64.whl /userdata
```

```
dominick@ubuntu:~/software/rknn-toolkit2/rknn-toolkit2-1.5.0/rknn_toolkit_lite2/
packages$ adb push rknn_toolkit_lite2-1.5.0-cp38-cp38-linux_aarch64.whl /userdat
a
rknn_toolkit_lite2-1.5.0-cp38-cp38-lin...hed. 5.9 MB/s (740367 bytes in 0.120s)
dominick@ubuntu:~/software/rknn-toolkit2/rknn-toolkit2-1.5.0/rknn_toolkit_lite2/
packages$
```

After the transmission is completed, open the serial terminal or SSH terminal of the development board, enter the directory "userdata", and then input the following commands to install.

```
cd /userdata
pip install rknn_toolkit_lite2-1.5.0-cp38-cp38-linux_aarch64.whl
```

After installation, it will be as shown in the picture.

```
root@ATK-DLRK356X:/# cd userdata/
root@ATK-DLRK356X:/userdata# pip install rknn_toolkit_lite2-1.5.0-cp38-cp38-linux_aarch64.whl
Processing ./rknn_toolkit_lite2-1.5.0-cp38-cp38-linux_aarch64.whl
Requirement already satisfied: numpy in /usr/lib/python3.8/site-packages (from rknn-toolkit-lite2==1.5.0) (1.18.2)
Requirement already satisfied: psutil in /usr/lib/python3.8/site-packages (from rknn-toolkit-lite2==1.5.0) (5.9.5)
Requirement already satisfied: ruamel.yaml in /usr/lib/python3.8/site-packages (from rknn-toolkit-lite2==1.5.0) (0.17.31)
Requirement already satisfied: ruamel.yaml.clib>=0.2.7 in /usr/lib/python3.8/site-packages (from ruamel.yaml->rknn-toolkit-lite2==
1.5.0) (0.2.7)
Installing collected packages: rknn-toolkit-lite2
Successfully installed rknn-toolkit-lite2-1.5.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager
. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

## 5.2 Testing the AI routine in Python

In the previous section, we installed RKNN Toolkit Lite2. In this section, we will verify whether the installed environment is functioning properly. Also, in the rknn_toolkit_lite2 directory under Ubuntu, open the terminal and input the following command to transfer the inference_with_lite routine in the examples directory to the userdata directory of the development board via adb.

```
adb push examples/inference_with_lite/ /userdata
```

```
dominick@ubuntu:~/software/rknn-toolkit2/rknn-toolkit2-1.5.0/rknn_toolkit_lite2$
 adb push examples/inference_with_lite/ /userdata
examples/inference_with_lite/: 5 files...d. 5.1 MB/s (35697765 bytes in 6.634s)
```

After the transmission is completed, open the serial terminal of the development board, enter the /userdata/inference_with_lite/ directory and input the following command to execute to check the result.

```
cd /userdata/inference_with_lite/
python test.py
```

```
root@ATK-DLRK356X:/# cd /userdata/inference_with_lite/
root@ATK-DLRK356X:/userdata/inference_with_lite# python test.py
--> Load RKNN model
done
--> Init runtime environment
I RKNN: [17:01:00.286] RKNN Runtime Information: librknnrt version: 1.5.0 (e6fe0c678@2023-05-25T08:09:20)
I RKNN: [17:01:00.287] RKNN Driver Information: version: 0.8.2
I RKNN: [17:01:00.288] RKNN Model Information: version: 4, toolkit version: 1.5.0+1fa95b5c(compiler version: 1.5.0 (e6fe0c678@2023
-05-25T16:15:03)), target: RKNPU lite, target platform: rk3566, framework name: PyTorch, framework layout: NCHW, model inference t
ype: static_shape
done
--> Running model
resnet18
-----TOP 5-----
[812]: 0.9996760487556458
[404]: 0.00024927023332566023
[657]: 1.449744013370946e-05
[466 833]: 9.023910024552606e-06
[466 833]: 9.023910024552606e-06

done
```

This routine uses the "space_shuttle_224.jpg" provided within the routine for inference. The model is a ResNet model trained by Rockchip Microelectronics on the ImageNet dataset and then converted into a RKNN model. Finally, you can see the printed indices of the five categories with the highest confidence levels. 812 represents the space shuttle. Regarding the indices of the ImageNet dataset, you can search for them online for classification, and this will not be elaborated here.

## 5.3 Testing the self-trained LeNet model in Python

In the previous section, we installed the board-side Python inference rknn_toolkit_lite2 tool. Next, we will test the AI model routine written in Python by ALIENTEK. The first and foremost thing to do is our essential model for deep learning, LeNet. It is one of the classic convolutional neural network (CNN) architectures in the field of deep learning, proposed by Yann LeCun et al. in 1998. It is one of the pioneers of convolutional neural networks and is widely regarded as the founding work of deep

learning. We trained a five-layer classic LeNet model using the handwritten digit recognition (mnist) dataset and can view its model structure through the Netron tool.

LeNet was originally designed for handwritten digit recognition, particularly for the automatic recognition of US postal codes. Its design concept was revolutionary at that time, introducing convolutional layers and sub-sampling (pooling) layers, which decomposed the input image into multiple layers of feature representations, effectively extracting the local features of the image. Next, the steps for running this routine on the board will be introduced. First, we will transfer the CD A of our development board - Basic Data → 01_codes → 01_AI_Routine → 01, Source Code → 01_lenet folder to the development board via adb or ssh, etc. It is recommended to copy all the routines to Ubuntu first, and here we still use the adb method for transmission.

We will first create a folder named aidemo in the /userdata directory on the development board to facilitate the unified placement of all the routines for testing later. Then, transfer the 01_lenet routine there. On the development board, enter the following command.

```
cd /userdata
mkdir aidemo
```

```
root@ATK-DLRK356X:/# cd /userdata/
root@ATK-DLRK356X:/userdata# mkdir aidemo
```

Then, go to the directory where the Ubuntu routine is located, transfer the routine to the development board via adb, and open the terminal in Ubuntu and enter the following commands.

```
adb push 01_lenet/ /userdata/aidemo
```

After the transmission is completed, return to the development board terminal and enter the corresponding directory to execute the following commands.

```
cd /userdata/aidemo/01_lenet/
python atk_lenet_demo.py
```

```
root@ATK-DLRK356X:/# cd /userdata/aidemo/01_lenet/
root@ATK-DLRK356X:/userdata/aidemo/01_lenet# python atk_lenet_demo.py
--> Load RKNN model
done
--> Init runtime environment
I RKNN: [17:11:41.181] RKNN Runtime Information: librknnrt version: 1.5.0 (e6fe0c678@2023-05-25T08:09:20)
I RKNN: [17:11:41.183] RKNN Driver Information: version: 0.8.2
I RKNN: [17:11:41.185] RKNN Model Information: version: 4, toolkit version: 1.5.0+1fa95b5c(compiler version: 1.5.0 (e6fe0c678@2023-
05-25T16:15:03)), target: RKNPU lite, target platform: rk3568, framework name: TFLite, framework layout: NHWC, model inference type
: static_shape
--> Running model
LeNet
-----TOP 5-----
[5]: 1.0
-1: 0.0
-1: 0.0
-1: 0.0
-1: 0.0

done
root@ATK-DLRK356X:/userdata/aidemo/01_lenet#
```

In the code, we input handwritten digit images of the number 5 for the inference.

5.png

From the output results, we can see that the probability for index 5 is 1.0. The handwritten digit dataset is a dataset consisting of 10 categories from 0 to 9, which exactly corresponds to the 0 to 9

datasets. Thus, we can see that the model is running normally. Everyone can also try to manually write some digits and resize them to 28x28 pixels to modify the source code for reading the image for inference. In the future, we plan to explain in the video how to train and convert to inference. Please stay tuned.

# Chapter 6. Face Detection and Five Key Points of Face Example Routine Testing

In this chapter, we introduce the example routine testing of face detection and the five key points of the face. Face detection is an important technology in the field of computer vision, aiming to identify the face regions in images or videos. This technology provides a foundation support for numerous applications, such as face recognition, expression analysis, human-computer interaction, security monitoring, etc. The goal of face detection is to automatically locate and mark all the faces that appear in the image, providing basic data for subsequent analysis and processing. In daily life, there are many places where it is necessary to detect the face first and then perform recognition, comparison or analysis functions on the face. Face detection is an indispensable and important part of this process.

This chapter will be divided into the following sections:

1. Routine Introduction
2. Routine Testing

## 6.1 Routine Overview

Traditional face detection methods usually involve multiple stages of processing procedures, including image preprocessing, feature extraction, and classifier. However, this approach faces some challenges in terms of computational cost and accuracy. In recent years, with the rise of deep learning technology, face detection methods based on convolutional neural networks (CNN) have made significant progress.

In the field of face detection, researchers have proposed many innovative methods. One of them is the Single-Stage Context Reasoning Face Detector (SCRFD), which was proposed by InsightFace in 2021. SCRFD is a face detection model. It adopts a single-stage architecture, combining localization and classification into one step, thereby improving the speed and efficiency of detection.

The working principle of SCRFD is based on deep neural networks. This network learns the feature representations and detection patterns of faces by training on a large amount of labeled data. During the testing stage, this network can quickly and accurately identify faces in images and output their positions and confidence scores. In summary, face detection technology achieves the goal of automatically recognizing faces in images and videos by using advanced computer vision and deep learning techniques. SCRFD, as one of these methods, brings higher efficiency and accuracy to face detection through its single-stage context reasoning architecture, promoting the development of face recognition and other applications. With the continuous progress of technology, face detection will continue to play an important role in various fields.

For more details about SCRFD, you can check the link.

1. Thesis：https://arxiv.org/abs/2105.04714
2. Source Code：https://github.com/deepinsight/insightface/tree/master/detection/scrfd

## 6.2 Routine Test

Before conducting the test of this routine, please ensure that the relevant reasoning environment and libraries have been installed on the development board as per Chapters 2 and 5, and also ensure that your development board has been connected to the IMX415 provided by ALIENTEK and the 5.5-inch 1080p MIPI screen or 720p MIPI screen available for sale by ALIENTEK.

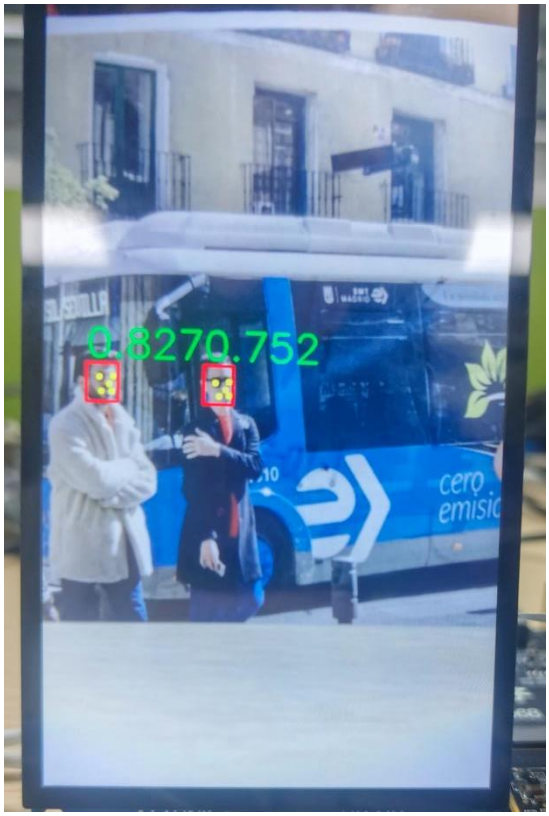
In Section 5.3, we have already created the aidemo directory at the board end. We have transferred the sample program to the /userdata/aidemo directory of the development board via adb. On the Ubuntu system, open the terminal in the directory where the sample program is located and enter the following commands.

```
adb push 11_facedet_scrfd_npu/ /userdata/aidemo
```

After the transmission is completed, return to the development board terminal and enter the corresponding directory to execute the following commands.

```
cd /userdata/aidemo/11_facedet_scrfd_npu
python main.py
```

When the operation is successful and a face or a face-like object is aligned with the camera, you can see that our MIPI screen can draw a box around the position of the face and display the confidence level as well as the points for two eyes, two mouths, and the nose (a total of 5 points). The operation effect is as shown in the following picture.




By pressing Ctrl+C on the RK3568 serial terminal, you can exit the program.

# Chapter 7. Person Segmentation Routine Test

The person segmentation program pp-humanseg is an open-source project of Baidu PaddlePaddle. We converted it into a rknn model and successfully deployed it on the RK3568 board. The result was quite good. This chapter will introduce to you how to deploy it on the development board.

This chapter is divided into the following sections:

1. Routine Introduction
2. Routine Test

## 7.1 Routine Introduction

Distinguishing characters and backgrounds at the pixel level is a classic task in image segmentation and has wide applications. Generally speaking, this task can be divided into two categories: segmentation for half-body portraits, which is referred to as portrait segmentation; and segmentation for full-body and half-body portraits, which is referred to as general portrait segmentation. For portrait segmentation and general portrait segmentation, PaddleSeg has released the PP-HumanSeg series of models, which have the advantages of high segmentation accuracy, fast inference speed, and strong generality. Moreover, the PP-HumanSeg series models can be used out of the box, deployed to products at zero cost, and also support fine-tuning for specific scene data to achieve better segmentation results. This routine adopts general portrait segmentation for full-body images.

For the general human image segmentation task, the Flying Team first constructed a large-scale human image dataset, and then used the state-of-the-art model of PaddleSeg. Finally, multiple PP-HumanSeg general human image segmentation models were released. The PP-HumanSegV2-Lite general human image segmentation model uses the ultra-lightweight segmentation model launched by PaddleSeg, and its mIoU accuracy is improved by 6.5% compared to the V1 model, while the inference time on mobile devices increases by 3ms. The PP-HumanSegV2-Mobile general segmentation model uses the PP-LiteSeg model developed by PaddleSeg, and its mIoU accuracy is improved by 1.49% compared to the V1 model, and the inference time on the server side is reduced by 5.7%.

For more information about the PP-HumanSeg models, you can refer to the following link: https://github.com/PaddlePaddle/PaddleSeg/tree/release/2.8/contrib/PP-HumanSeg.

| Model name | Optimal input size | Precision mIou(%) | Time taken for mobile device inference (ms) | Server-side inference time (ms) |
|---|---|---|---|---|
| PP-HumanSegV1-Lite | 192x192 | 86.02 | 12.3 | - |
| PP-HumanSegV2-Lite | 192x192 | 92.52 | 15.3 | - |
| PP-HumanSegV1-Mobile | 192x192 | 91.64 | - | 2.83 |

| Model name | Optimal input size | Precision mIou(%) | Time taken for mobile device inference (ms) | Server-side inference time (ms) |
|---|---|---|---|---|
| PP-HumanSegV2-Mobile | 192x192 | 93.13 | - | 2.67 |
| PP-HumanSegV1-Server | 512x512 | 96.47 | - | 24.9 |

## 7.2 Test Procedure

Before conducting the test of this routine, please ensure that the relevant reasoning environment and libraries have been installed on the development board as per Chapters 2 and 5, and also ensure that your development board has been connected to the IMX415 provided by ALIENTEK and the 5.5-inch 1080p MIPI screen or 720p MIPI screen available for sale by ALIENTEK.

In Section 5.3, we have already created the board-end aidemo directory. We have transferred all the files in the folders such as "01 - Basic Data" → "01_codes" → "01_AI_Routines" → "12_pp_human_seg_npu" within the A disk of the development board's CD to the "userdata/aidemo" directory on the development board. On the Ubuntu system, open the terminal in the directory where the routine is located and input the following commands.

```
adb push 12_pp_human_seg_npu/ /userdata/aidemo
```

After the transmission is completed, return to the development board terminal and enter the corresponding directory to execute the following commands.

```
cd /userdata/aidemo/12_pp_human_seg_npu
python main.py
```

After the operation is successful, have a person or a picture of a person face the camera. Then you can see the area where the person is located being separated out and painted green on the screen. The operation effect is shown as follows.

# Chapter 8. Vehicle detection, lane line recognition, and driving area segmentation routines

In the current field of autonomous driving, tasks such as vehicle detection, lane line recognition, and driving area segmentation play a crucial role. Vehicle detection is used to detect the congestion level of the vehicle itself and surrounding vehicles, lane line recognition enables vehicles to better avoid crossing the lines and select the optimal lanes, and driving area segmentation allows vehicles to determine within what range they can freely drive. Therefore, we have provided this routine for everyone to test and experience.

This chapter will be divided into the following sections:

1. Routine Introduction
2. Routine Testing

## 8.1 Routine Overview

This chapter tests a combined routine for vehicle detection, lane line recognition, and driving area segmentation. It is derived from the open-source project YOLOP, which is an efficient multi-task network. This network can jointly handle the three key tasks in autonomous driving: target detection, driving area segmentation, and lane detection. It not only saves computing costs and reduces inference time, but also improves the performance of each task. For more details about YOLOP, please refer to the link: https://github.com/hustvl/YOLOP. We converted it into a rknn model and made certain modifications to adapt it to the rk3568 development board. The overall effect for single-image processing is still quite good.

The following is the official comparison detection results of similar models (not experimental data on the rk3568):

Traffic target detection results:

| Model | Recall(%) | mAP50(%) | Speed(fps) |
|---|---|---|---|
| Multinet | 81.3 | 60.2 | 8.6 |
| DLT-Net | 89.4 | 68.4 | 9.3 |
| Faster R-CNN | 77.2 | 55.6 | 5.3 |
| YOLOv5s | 86.8 | 77.2 | 82 |
| YOLOP | 89.2 | 76.5 | 41 |

Result of area segmentation for drivable zone:

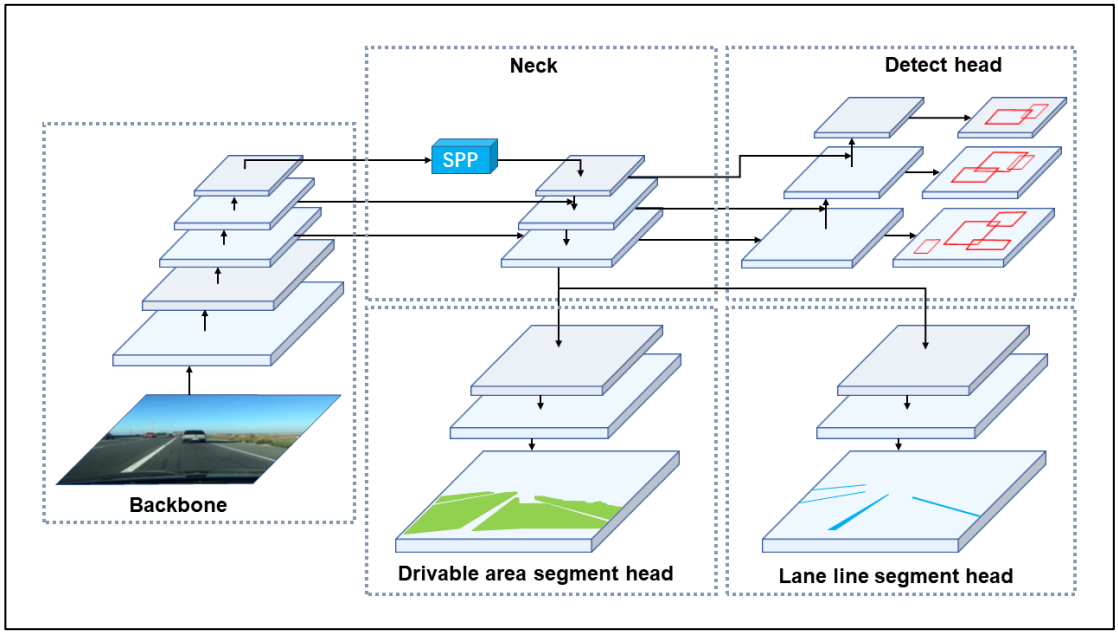
| Model | mIOU(%) | Speed(fps) |
|---|---|---|
| Multinet | 71.6 | 8.6 |

| Model | mIOU(%) | Speed(fps) |
|---|---|---|
| DLT-Net | 71.3 | 9.3 |
| PSPNet | 89.6 | 11.1 |
| YOLOP | 91.5 | 41 |

Lane line detection result:

| Model | mIOU(%) | IOU(%) |
|---|---|---|
| ENet | 34.12 | 14.64 |
| SCNN | 35.79 | 15.84 |
| ENet-SAD | 36.56 | 16.02 |
| YOLOP | 70.50 | 26.20 |

The model framework of YOLOP is shown in the figure below.




## 8.2 Routine Testing

Before conducting the test for this routine, please ensure that the relevant reasoning environment and libraries have been installed on the development board as per Chapters 2 and 5, and that your development board has been connected to the accompanying IMX415 from ALIENTEK and the 5.5-inch 1080p MIPI screen or 720p MIPI screen available from ALIENTEK.

In Section 5.3, we have already created the board-end aidemo directory. Copy the files from the A disk of the development board's CD - Basic Materials → 01_codes → 01_AI_Routine → 01, Source Code → 10_carseg_yolop_npu Routine to Ubuntu and then transfer them to the

/userdata/aidemo directory on the development board via adb. Open the terminal in the directory where the routine is located on Ubuntu and enter the following commands.

```
adb push 10_carseg_yolop_npu/ /userdata/aidemo
```

After the transmission is completed, return to the development board terminal and enter the corresponding directory to execute the following commands. Enter the routine directory.

```
cd /userdata/aidemo/10_carseg_yolop_npu
```

We have provided two versions of the routines: video.py is used to read the video stream of a traffic flow video named test.mp4 downloaded from the internet for inference, while cam.py is used to read the video stream from the camera for inference. Let's first test the video.py routine for reading the video stream and execute the following command.

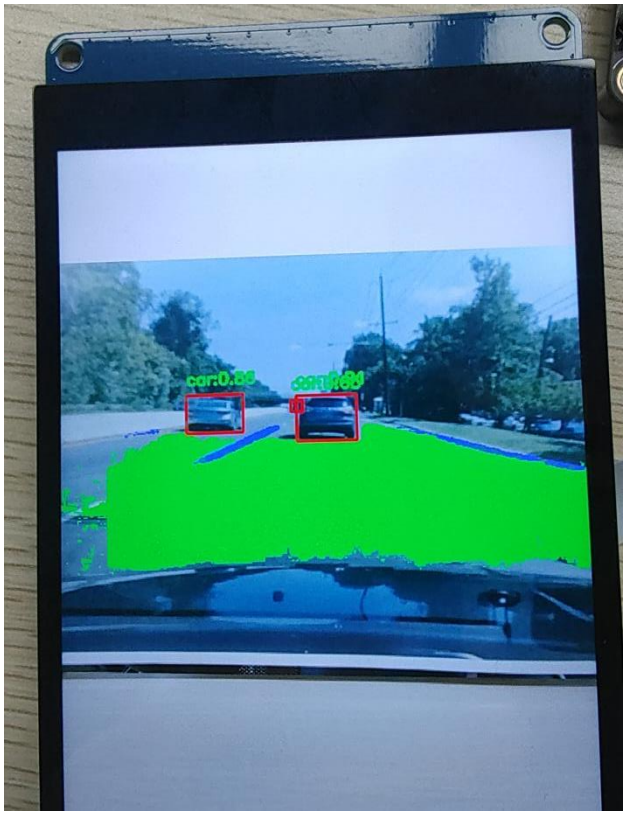
```
python video.py
```

The operation effect is shown below. It can be observed that the system becomes very sluggish during operation, which is normal. This is due to the performance limitations of the rk3568 processor.


Press Ctrl+c to exit the program. Then test the routine that uses the camera as input. Enter the following commands.

```
python cam.py
```

You can print a picture showing the lane lines of the vehicle, or use another screen to align the video or image of the vehicle's movement with the camera. You will find that the effect is the same as that of streaming the video. As shown in the following picture, the difference is not significant.


car:0.88

# Chapter 9. Common classification network models + RKNN AI routine

This chapter will explain the routines for testing various classification models trained using the animal dataset on the development board. The following routine is adopted.

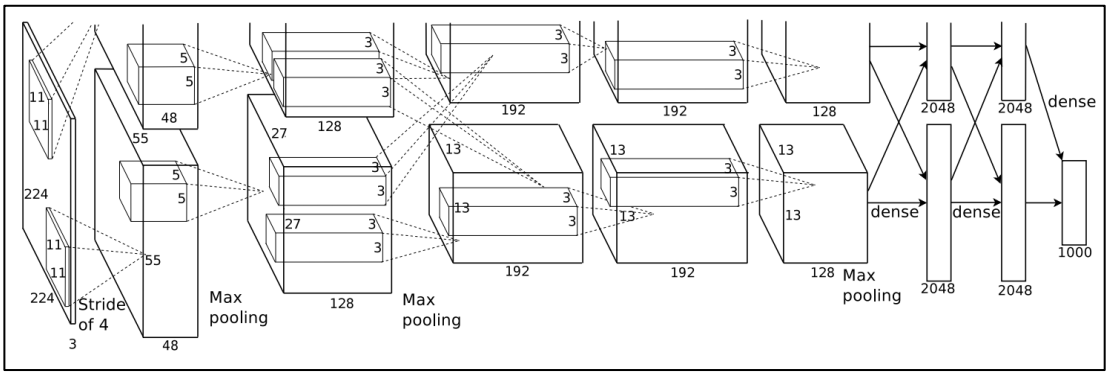
This chapter will be divided into the following sections:

1. Routine introduction
2. Routine test for self-trained alexnet model
3. Routine test for self-trained vggnet model
4. Routine test for resnet model using transfer learning
5. Routine test for mobilenet_v1 model using transfer learning

## 9.1 Routine Overview

AlexNet is a classic convolutional neural network in the field of deep learning, proposed by Alex Krizhevsky et al. in 2012. It made a significant breakthrough in the ImageNet image classification competition, marking the rise of deep learning. AlexNet utilized deep convolutional and pooling layers to extract image features, and introduced the ReLU activation function to alleviate the problem of gradient disappearance. Its depth and parameter quantity were innovative at that time, stimulating the research craze of deep neural networks. It also was the first to use GPU for training, accelerating the deep learning training process. The success of AlexNet inspired the development of subsequent network architectures, such as VGG and ResNet.

Paper Link: http://www.cs.toronto.edu/~hinton/absps/imagenet.pdf




VGGNet is a classic convolutional neural network architecture, proposed by the research team from the University of Oxford in 2014. VGGNet is renowned for its simple yet profound design principles, which construct deep networks by stacking multiple smaller convolutional layers and pooling layers. Its core idea is to extract higher-level features of images by increasing the depth of the network, thereby achieving better performance. The architecture of VGGNet is very regular, with all convolutional layers using the same-sized convolution kernels, making the design of the network more

consistent. Although VGGNet achieved excellent results in the ImageNet image classification competition, its large number of parameters led to higher computational costs for training and inference.

Paper link: https://arxiv.org/pdf/1409.1556.pdf

| ConvNet config. (Table 1) | smallest image side | | top-1 val. error (%) | top-5 val. error (%) |
|---|---|---|---|---|
| | train ($S$) | test ($Q$) | | |
| A | 256 | 256 | 29.6 | 10.4 |
| A-LRN | 256 | 256 | 29.7 | 10.5 |
| B | 256 | 256 | 28.7 | 9.9 |
| C | 256 | 256 | 28.1 | 9.4 |
| | 384 | 384 | 28.1 | 9.3 |
| | [256;512] | 384 | 27.3 | 8.8 |
| D | 256 | 256 | 27.0 | 8.8 |
| | 384 | 384 | 26.8 | 8.7 |
| | [256;512] | 384 | 25.6 | 8.1 |
| E | 256 | 256 | 27.3 | 9.0 |
| | 384 | 384 | 26.9 | 8.7 |
| | [256;512] | 384 | **25.5** | **8.0** |

ResNet (Residual Network) is a significant milestone in the field of deep learning, proposed by Kaiming He et al. in 2015. Its core innovation lies in introducing residual blocks to address the problems of vanishing gradients and training difficulties, allowing the construction of extremely deep neural networks. Residual blocks transmit information through skip connections, making the network more amenable to optimization. ResNet can flexibly stack multiple residual blocks, creating models of different depths. Its performance surpasses previous networks and has achieved remarkable results in tasks such as image classification, object detection, and segmentation. The idea of ResNet has also influenced subsequent network designs, providing a powerful solution to the challenges of deep learning.

MobileNet-v1 is a lightweight convolutional neural network designed to achieve efficient image classification and feature extraction on resource-constrained mobile devices and embedded systems. By adopting depthwise separable convolutions and a carefully designed structure, MobileNet-v1 maintains reasonable performance while significantly reducing computational complexity and model size. It allows users to adjust the network width and input image resolution to adapt to different application scenarios. MobileNet-v1 is applicable to tasks such as image classification, object detection, and face recognition, providing an effective solution for deep learning applications on embedded systems and mobile devices, and promoting the development of lightweight neural networks.

We trained several essential classification network models for deep learning using TensorFlow, and converted them into rknn models that can be used for inference on the rk3568 development board. All four networks were trained using the 10-class classification in the animal dataset. The categories of the animal dataset from 0 to 9 are butterfly, cat, chicken, cow, dog, elephant, horse, sheep, spider, and squirrel.

The animal dataset can be downloaded on Kaggle:

https://www.kaggle.com/datasets/alessiocorrado99/animals10

## 9.2 Testing the self-training AlexNet model routine

Next, I will introduce to you how the self-trained Alexnet model runs on the board. This section explains how to transfer the routine to the development board for running. We will copy the files from the A drive of the development board disc - basic materials → 01_codes → 01_AI_routine → 01,

source code → 02_alexnet (including atk_rknn_alexnet_demo.zip) to Ubuntu. Unzip the atk_rknn_alexnet_demo.zip archive, and enter the corresponding directory. Before compiling, please make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program.

```
./build.sh
```

After the compilation is completed, execute the following command to push the compiled program and model to the development board via adb.

```
adb push install/atk_alexnet_demo/ /userdata/aidemo
```

Open the serial port terminal, and then go to the directory where the corresponding routine executable file is located and execute the following commands.

```
./atk_alexnet ./model/animal-alexnet.rknn
```

Align the printed test image with the camera and make sure it covers as much of the screen area as possible. You will find that the recognition effect of AlexNet is relatively average. Using other models, such as MobileNet-V1, will yield better results.

## 9.3 Testing the training VggNet model routine

This section explains how to transfer the routine to the development board for running. We will copy the A disk of the development board CD-ROM - Basic Materials → 01_codes → 01_AI_Routine → 01, Source Code → 03_vggnet (including atk_rknn_vggnet_demo.zip) to Ubuntu. Unzip the atk_rknn_vggnet_demo.zip archive, and enter the corresponding directory. Before compiling, please make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program.

```
./build.sh
```

After the compilation is completed, execute the following command to push the compiled program and model to the development board via adb.

```
adb push install/atk_vggnet_demo/ /userdata/aidemo
```

Open the serial port terminal, and then go to the directory where the corresponding routine executable file is located and execute the following commands.

```
./atk_vggnet ./model/animal-vggnet16.rknn
```

It can be observed that the effect will be somewhat better than that of Alexnet, but it will be slower than Alexnet.

## 9.4 Example routine testing of the ResNet model using transfer learning

This section explains how to transfer the routine to the development board for it to run. We will copy the files from the development board's CD, namely A drive - Basic Data → 01_codes → 01_AI_Routine → 01, Source Code → 04_transfer_resnet, into Ubuntu. We will decompress the atk_rknn_resnet_demo.zip archive and enter the corresponding directory. Before compiling, make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program.

```
./build.sh
```

After the compilation is completed, execute the following command to push the compiled program and model to the development board via adb.

```
adb push install/atk_resnet_demo/ /userdata/aidemo
```

Open the serial port terminal, and then go to the directory where the corresponding routine executable file is located and execute the following commands.

```
./atk_resnet ./model/transferlearn-resnet50.rknn
```

## 9.5 Testing the example routine using the transfer learning-based MobileNet_v1 model

This section explains how to transfer the routine to the development board for running. We will copy the A drive of the development board disc - Basic Materials → 01_codes → 01_AI_Routine → 01, Source Code → 05_transfer_mobilenet_v1 in atk_rknn_mobilenet_v1_demo.zip to Ubuntu. Unzip the atk_rknn_mobilenet_v1_demo.zip archive, and enter the corresponding directory. Before compiling, please make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program.

```
./build.sh
```

After the compilation is completed, execute the following command to push the compiled program and model to the development board via adb.

```
adb push install/atk_mobilenet_v1_demo/ /userdata/aidemo
```

Open the serial port terminal, and then go to the directory where the corresponding routine executable file is located and execute the following commands.

```
./atk_mobilenet_v1 ./model/transferlearn-mobilnetv1.rknn
```

This routine should perform the best, as it can quickly identify the categories and the model is relatively small, fast and efficient. It is highly recommended that everyone use this classification model for related applications.


4 0.801758
1 0.188965
0 0.007767
7 0.001654
3 0.000061

# Chapter 10. yolo series object detection model + RKNN AI routine

The task of object detection is to identify all the interesting targets or objects in an image, determine their categories and positions. It is one of the core issues in the field of computer vision. Due to the different appearances, shapes, and postures of various objects, as well as the interference of factors such as lighting and occlusion during imaging, object detection has always been the most challenging problem in the field of computer vision. This chapter explains how to run the most commonly used and popular YOLO series model on the ALIENTEK RK3568 development board.

This chapter will be divided into the following sections:

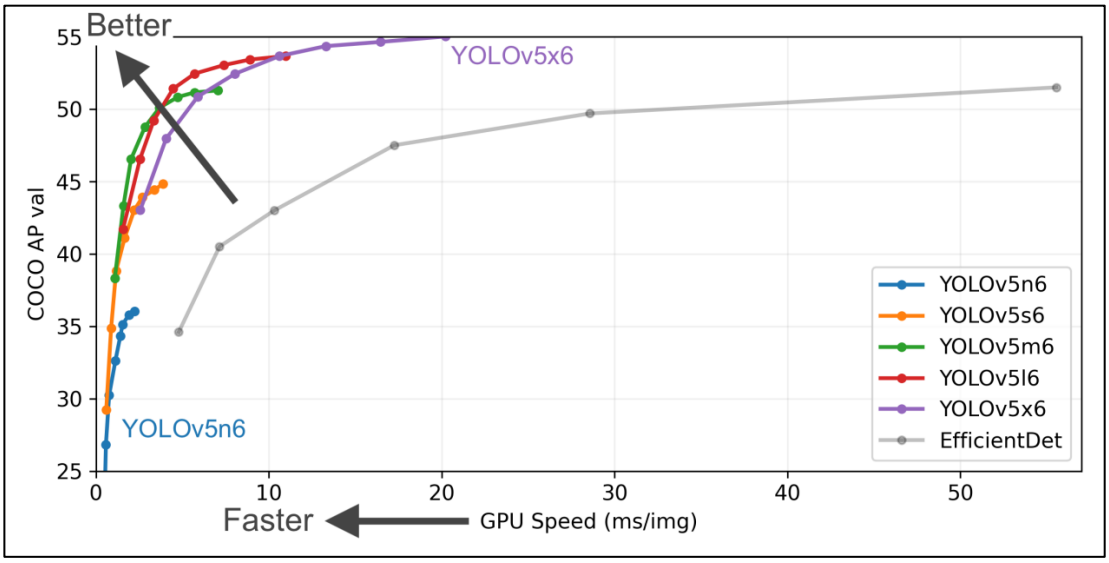
1. Introduction to YOLO series routines
2. Test of yolov5 routine
3. Test of yolov7 routine
4. Test of yolox routine

## 10.1 Introduction to the YOLO Series of Routines

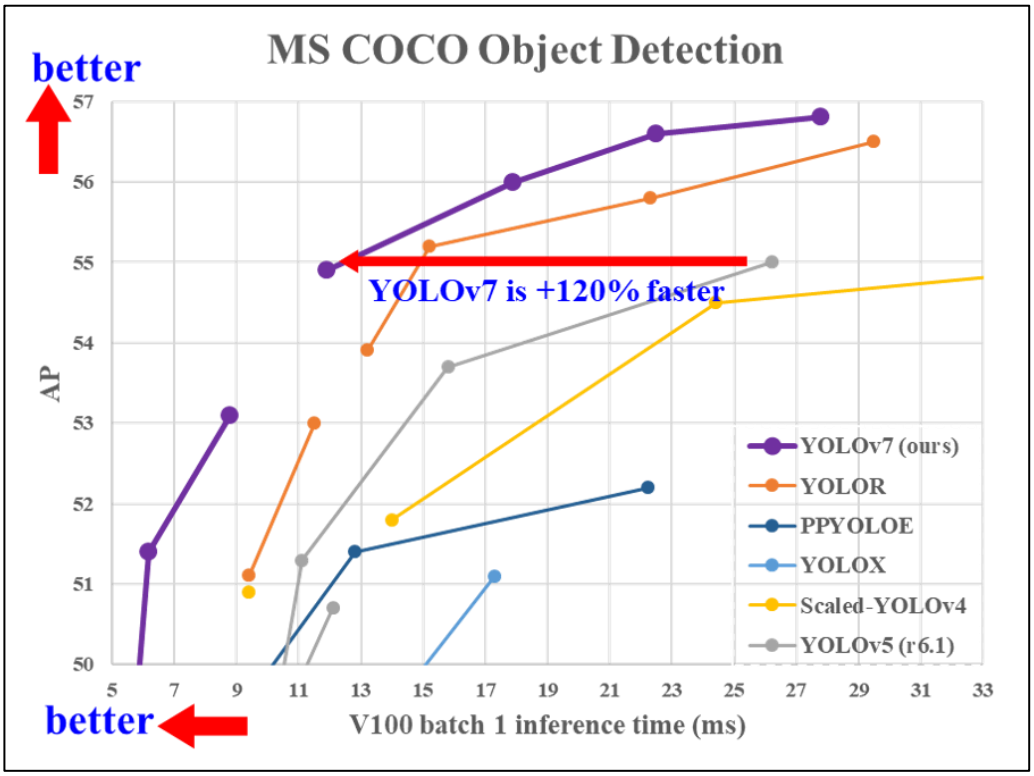
YOLOv5 is one of the latest versions of the YOLO object detection algorithm series. As an important advancement in the field of computer vision, YOLOv5 has achieved significant performance improvements in object detection. Compared to previous versions, YOLOv5 adopts a single-stage detection strategy, enabling the localization and classification of objects through a single forward pass. It introduces the "CSPDarknet53" architecture as the base network to extract rich feature information, thereby effectively enhancing the accuracy of detection. YOLOv5 has also made improvements in multi-scale detection, capable of detecting objects of different scales, from small to large. This performs well in handling diverse scenarios. Additionally, data augmentation techniques are widely applied, through random cropping, color transformation, etc., to increase the diversity of training data and enhance the generalization ability of the model. In terms of computational efficiency, YOLOv5 maintains high detection performance while optimizing the size and speed of the model, making it suitable for embedded devices, mobile terminals, and real-time applications. Its open-source nature enables researchers and developers to freely access the code and pre-trained models for research and customization. In summary, YOLOv5 through single-stage detection, efficient network architecture, multi-scale strategy, and data augmentation technologies, significantly improves the performance of object detection. It has broad application prospects in computer vision, autonomous driving, security monitoring, etc., bringing new solutions to real-time object detection tasks.

For training and more details, please refer to the source code link of YOLOv5:

https://github.com/ultralytics/yolov5




YOLOv7 is a target detection algorithm, being the latest version of the YOLO series. YOLOv7 is based on the YOLOv5 algorithm and incorporates some new technologies and optimizations, thereby significantly enhancing the detection performance and speed. Compared with previous versions, YOLOv7 boasts better accuracy and faster inference speed, and also supports a wider range of application scenarios.

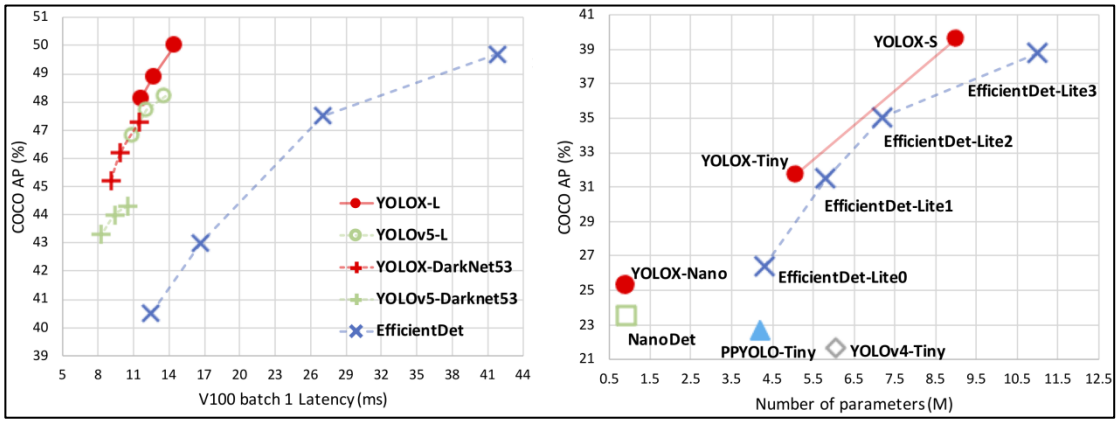



Paper link: https://arxiv.org/abs/2207.02696

Code link: https://github.com/WongKinYiu/yolov7

There is another network called YOLOX. YOLOX is a high-performance and real-time object detection algorithm, dedicated to pushing the performance limit in the field of object detection. Its innovative design and optimization have achieved a better balance between speed and accuracy, making

it a new generation leader in the field of object detection. YOLOX performs well in multi-scale detection and small-scale object detection, and is suitable for various scenarios, such as intelligent transportation, security monitoring, industrial vision, etc. Its outstanding performance makes it an ideal choice for real-time object detection tasks. As an open-source project, YOLOX provides code and pre-trained models, allowing researchers and developers to freely access and use them. It brings innovation to the field of object detection and makes significant contributions to the development of computer vision. Whether in terms of speed, accuracy, or flexibility, YOLOX demonstrates remarkable potential and brings new possibilities for real-time object detection tasks.




Performance data comparison

The GitHub website of YOLOX: https://github.com/Megvii-BaseDetection/YOLOX

The documentation of YOLOX: https://yolox.readthedocs.io/en/latest/

Rockchip's Modelzoo provides the performance and accuracy test results data of some YOLO models on the RKNN platform, as well as relevant materials such as the data sets used for the routines. Please refer to the link:

https://github.com/airockchip/rknn_model_zoo/tree/main/models/CV/object_detection/yolo

## 10.2 Yolov5 Routine Test

This section explains how to transfer the routine to the development board for running. We will copy the development board CD's A drive - Basic Materials → 01_codes → 01_AI_Routine → 01, Source Code → 06_yolov5 (where rknn_yolov5_demo.zip and yolov5.sh are located) to the development board. Unzip the rknn_yolov5_demo.zip archive, and enter the corresponding directory. Before compiling, please make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program. Push the compiled program and model to the development board via adb.

```
./build-linux_RK356X.sh
adb push install/atk_rknn_yolo_v5_demo/ /userdata/aidemo
```

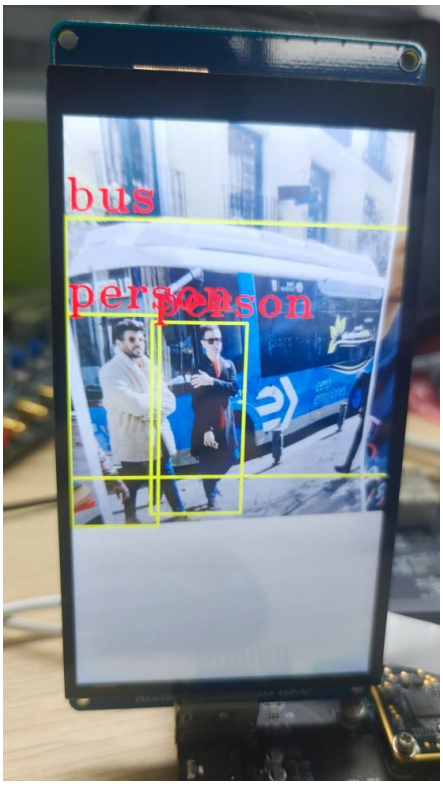
On the development board, enter the serial terminal and go to the directory /userdata/aidemo/atk_rknn_yolo_v5_demo. Then execute the following command.

```
cd /userdata/aidemo/atk_rknn_yolo_v5_demo/
./rknn_yolo_demo yolov5 fp ./model/yolov5s_relu_tk2_RK356X_i8.rknn ./model/RK_anchors_yolov5.txt
```

Or you can use the execution script yolov5.sh provided by us, place it in the corresponding directory and directly execute the following command.

```
chmod +x yolov5.sh
./yolov5.sh
```

The operation effect is shown in the following figure.




## 10.3 Yolov7 Routine Test

This section explains how to transfer the routine to the development board for running. We will copy the development board CD's A drive - Basic Materials → 01_codes → 01_AI_Routine → 01, Source Code → 06_yolov7 (including rknn_yolov7_demo.zip and yolov7.sh) to the development board. Unzip the rknn_yolov7_demo.zip archive, and enter the corresponding directory. Before compiling, please make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program. Push the compiled program and model to the development board via adb.

```
./build-linux_RK356X.sh
adb push install/atk_rknn_yolo_v7_demo/ /userdata/aidemo
```

On the development board, enter the serial port terminal and go to the directory /userdata/aidemo/atk_rknn_yolo_v7_demo, then execute the following command.

```
cd /userdata/aidemo/atk_rknn_yolo_v7_demo/
./rknn_yolo_demo yolov7 fp ./model/yolov7-tiny_tk2_RK356X_i8.rknn ./model/RK_anchors_yolov7.txt
```

Or you can use the execution script yolov7.sh provided by us, place it in the corresponding directory and directly execute the following command.

```
chmod +x yolov7.sh
```

```
./yolov7.sh
```

The effect is similar to that of YOLOv5.

## 10.4 YOLOX Routine Test

This section explains how to transfer the routine to the development board for it to run. We will copy the development board CD's A drive - Basic Data → 01_codes → 01_AI_Routine → 01, Source Code → 06_yolox (where rknn_yolox_demo.zip and yolox.sh are located) to the development board. Unzip the rknn_yolox_demo.zip archive, and enter the corresponding directory. Before compiling, please make sure that your routine has been installed with the cross-compiler as per Chapter 2 and Chapter 3 and updated the NPU service. Open the terminal and execute the following commands to compile the program. Then, push the compiled program and model to the development board via adb.

```
./build-linux_RK356X.sh
adb push install/rknn_yolox_demo.zip/ /userdata/aidemo
```

On the development board, enter the serial port terminal and go to the directory /userdata/aidemo/atk_rknn_yolox_demo, then execute the following command.

```
cd /userdata/aidemo/atk_rknn_yolox_demo/
./rknn_yolo_demo yolox fp ./model/yoloxs_tk2_RK356X_i8.rknn 0
```

Or you can use the execution script provided by us, yolox.sh, and place it in the corresponding directory. Then simply execute the following command.

```
chmod +x yolox.sh
./yolox.sh
```

# Chapter 11. English-Chinese Translation Routine

A language translation model is an AI-based tool designed to achieve automatic translation between different languages. It can translate a piece of text from one language to another, helping people overcome language barriers and facilitate cross-cultural communication and interaction. This chapter will introduce how to use the English-Chinese translation routine on the rk3568.

This chapter is divided into the following sections:

1. Routine Introduction
2. Routine Description

## 11.1 Routine Introduction

"Lite-Transformer" is a lightweight transformer model used in natural language processing and other sequence-to-sequence tasks. This concept typically refers to a streamlined and optimized version of the original transformer architecture, designed to achieve higher efficiency and speed in resource-constrained environments. The original transformer model has achieved remarkable results in natural language processing tasks, but its large number of parameters and computational requirements may limit its application on lower-resource devices. Lite-Transformer is a language translation project by the Rixin Micro modelzoo team, trained using the dataset "news-commentary-v15", an English-to-Chinese news dataset. The dataset is not large, so the actual effect of English-to-Chinese translation needs improvement. For more details on training and model conversion, please refer to the following link:

1、https://github.com/airockchip/rknn_model_zoo/tree/main/models/NLP/NMT/lite-transformer
2、https://github.com/airockchip/lite-transformer/blob/master/README_RK.md

## 11.2 Routine Test

Copy the compressed routine files from the CD-ROM development board disc A (basic data) → 01_codes → 01_AI_routine → 01, source code → 13_lite_transformer (the folder inside) to the previously created aidemo directory on the Ubuntu system. Open the Ubuntu terminal and execute the command "unzip lite-transformer.zip" or manually right-click to decompress in the graphical interface. After decompression, enter the routine directory and open the terminal to execute the following commands to compile the routine.

```
./build-linux_RK356X.sh
```

```
  367 |     char * token_embed_file = "model/token_embed.bin";
      |                               ^~~~~~~~~~~~~~~~~~~~~~~
/home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/src/main.cc:379:34: 警告:  ISO C++
 forbids converting a string constant to 'char*' [-Wwrite-strings]
  379 |     char * position_embed_file = "model/position_embed.bin";
      |                                  ^~~~~~~~~~~~~~~~~~~~~~~~~~
[100%] Linking CXX executable rknn_lite_transformer_demo
[100%] Built target rknn_lite_transformer_demo
[100%] Built target rknn_lite_transformer_demo
Install the project...
-- Install configuration: ""
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/./rknn_lite_transformer_demo
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/lib/librknnrt.so
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/.//model
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/.//model/position_embed.bin
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/.//model/dict.txt
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/.//model/bpe.txt
-- Installing: /home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer/install/rknn_lite_t
ransformer_demo_Linux/.//model/token_embed.bin
/home/dominick/AI-DEMO/demo/13_lite_transformer/lite-transformer
```

After the compilation is completed, an "install" directory will be created. Execute the following command to enter the "install" directory and transfer the folder to the adb using adb.

```
cd install/
adb push rknn_lite_transformer_demo_Linux/ /userdata/aidemo
```

Transfer the routine to the development board and execute the following commands in the corresponding directory of the development board terminal.

```
./rknn_lite_transformer_demo ./model/lite-transformer-encoder-16_RK356X_fp.rknn ./model/lite-transformer-decoder-16_RK356X_fp.rknn
```

If the following prompt appears, simply type in the English text and press Enter.

```
output tensors:
  index=0, name=decoder_output, n_dims=4, dims=[1, 1, 36808, 1], n_elems=36808, size=73616, fmt=NCHW, type=FP16, qnt_type=AFFINE, z
p=0, scale=1.000000
  index=1, name=save_key0, n_dims=4, dims=[1, 4, 16, 64], n_elems=4096, size=8192, fmt=NCHW, type=FP16, qnt_type=AFFINE, zp=0, scal
e=1.000000
  index=2, name=save_value0, n_dims=4, dims=[1, 4, 16, 64], n_elems=4096, size=8192, fmt=NCHW, type=FP16, qnt_type=AFFINE, zp=0, sc
ale=1.000000
  index=3, name=save_key1, n_dims=4, dims=[1, 4, 16, 64], n_elems=4096, size=8192, fmt=NCHW, type=FP16, qnt_type=AFFINE, zp=0, scal
e=1.000000
  index=4, name=save_value1, n_dims=4, dims=[1, 4, 16, 64], n_elems=4096, size=8192, fmt=NCHW, type=FP16, qnt_type=AFFINE, zp=0, sc
ale=1.000000
  index=5, name=save_key2, n_dims=4, dims=[1, 4, 16, 64], n_elems=4096, size=8192, fmt=NCHW, type=FP16, qnt_type=AFFINE, zp=0, scal
e=1.000000
  index=6, name=save_value2, n_dims=4, dims=[1, 4, 16, 64], n_elems=4096, size=8192, fmt=NCHW, type=FP16, qnt_type=AFFINE, zp=0, sc
ale=1.000000
==============================
请输入需要翻译的英文,输入q退出:
```

Enter "what", then press the Enter key to see the output translated into Chinese as "what".

```
请输入需要翻译的英文,输入q退出:
what
==================================
input_word: what
    bpe as: what
enc input tokens:
1 1 1 1 1 1 1 1 1 1 1 1 1 352 2
Encoder preprocess use time: 3.85 ms
rknn encoder run
Encoder inference use time: 28.00 ms
rknn decoder run
546 : 9.015625
266 : 12.445312
439 : 7.191406
2 : 11.343750
Decoder use total time: 128.04 ms
2 546 266 439 2 1 1 1 1 1 1 1 1 1 1
==================================
什么?
```

Let's test this phrase. If we input "how is the weather today", the output can be correctly translated as "What's the weather like today?".

```
how is the weather today
==================================
input_word: how
    bpe as: how
input_word: is
    bpe as: is
input_word: the
    bpe as: the
input_word: weather
    bpe as: weather
input_word: today
    bpe as: today
enc input tokens:
1 1 1 1 1 1 1 1 1 1 590 23 6 5392 639 2
Encoder preprocess use time: 11.68 ms
rknn encoder run
Encoder inference use time: 24.75 ms
rknn decoder run
603 : 8.742188
647 : 11.804688
4 : 7.816406
1074 : 12.468750
1104 : 14.492188
18 : 6.121094
1966 : 8.445312
135 : 11.429688
4 : 6.277344
8 : 7.464844
2 : 11.437500
Decoder use total time: 317.11 ms
2 603 647 4 1074 1104 18 1966 135 4 8 2 1 1 1 1
==================================
今天的天气是怎样的。
```

After testing, it was found that the translation of some words or phrases was not very effective. Some words and phrases were translated incorrectly. The explanation from RENSEN Micro's model zoo is that the training dataset is too small. If you are interested, you can try using a larger dataset to train the model and then convert it to RKNN for inference to see if there are any different effects.