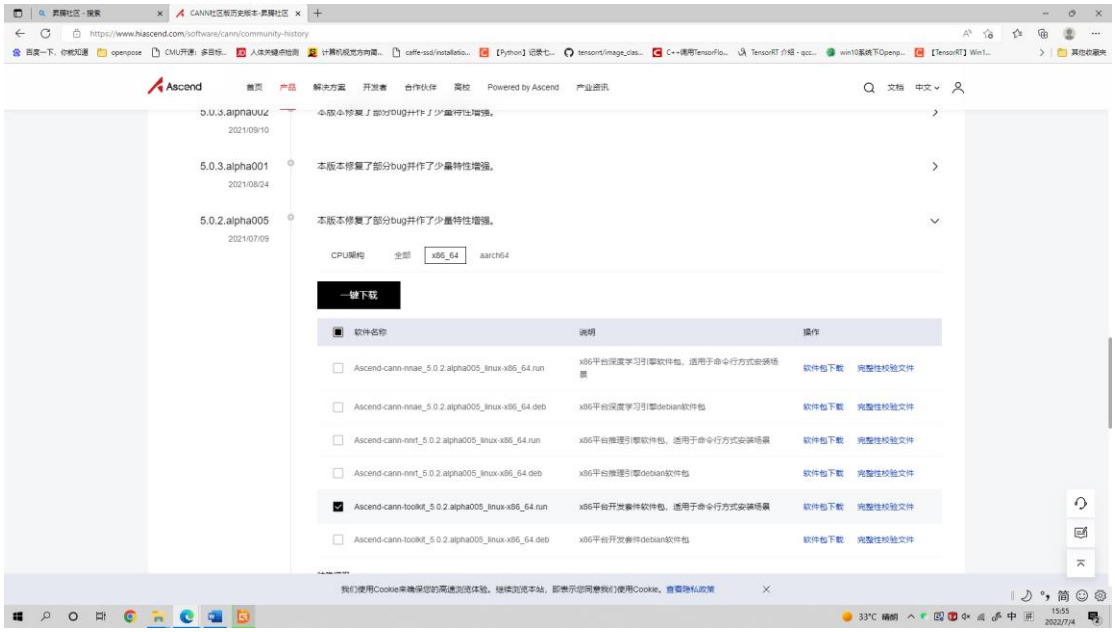
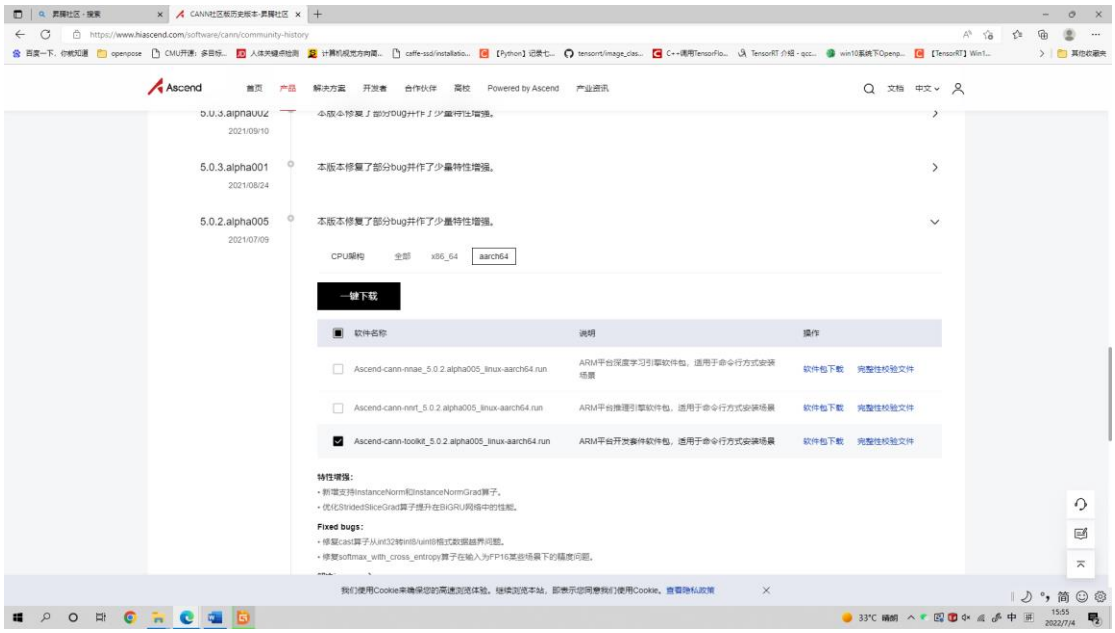
1、 制卡（略，同 5.1.RC1 版本，注意对应驱动版本变更为 1.0.11）

2、 分设开发环境安装





开发环境安装两种工具包

2.1 安装 OS 依赖

检查源

安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在 root 用户下执行如下命令检查源是否可用。

```
apt-get update
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，则检查网络是否连接或者把“/etc/apt/sources.list”文件中的源更换为可用的源或使用镜像源（以配置华为镜像源为例，可参考华为开源镜像站）。如需配置网络代理，请参见配置系统网络代理。

配置安装用户权限

当用户使用非 root 用户安装时，需要操作该章节，否则请忽略。

请以 root 用户执行如下操作。

1. 安装 sudo，使用如下命令安装。
   ```
   apt-get install sudo
   ```
2. 打开“/etc/sudoers”文件：
   ```
   chmod u+w /etc/sudoers

   vi /etc/sudoers
   ```
3. 在该文件中添加如下内容：
   ```
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/sh, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5, /usr/bin/unzip
   ```
   其中 *username* 代表安装用户，请根据实际替换。

   注意

   - 请确保“ /etc/sudoers”文件存在“ #includedir /etc/sudoers.d”，如果没有该信息，请手动添加至文件末尾。
   - 用户在完成 CANN 软件包安装后，可自行取消 sudo 权限。
   - 用户在卸载或升级 CANN 软件包时，也需配置以上用户权限。
4. 添加完成后，执行**:wq!**保存文件。
5. 执行以下命令取消“/etc/sudoers”文件的写权限：
   ```
   chmod u-w /etc/sudoers
   ```

安装依赖

1. 检查系统是否安装 python 依赖以及 gcc 等软件。

分别使用如下命令检查是否安装 gcc，make 以及 python 依赖软件等。

```
gcc --version

g++ --version

make --version

cmake --version

dpkg -l zlib1g| grep zlib1g| grep ii

dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii

dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii

dpkg -l openssl| grep openssl| grep ii

dpkg -l libssl-dev| grep libssl-dev| grep ii

dpkg -l libffi-dev| grep libffi-dev| grep ii

dpkg -l unzip| grep unzip| grep ii

dpkg -l pciutils| grep pciutils| grep ii

dpkg -l net-tools| grep net-tools| grep ii

dpkg -l libblas-dev| grep libblas-dev| grep ii

dpkg -l gfortran| grep gfortran| grep ii

dpkg -l libblas3| grep libblas3| grep ii

dpkg -l libopenblas-dev| grep libopenblas-dev| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步（以下回显仅为示例，请以实际情况为准）。

```
gcc (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0

g++ (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0

GNU Make 4.1

cmake version 3.10.2

zlib1g:arm64      1:1.2.11.dfsg-0ubuntu2 arm64        compression library - runtime

zlib1g-dev:arm64 1:1.2.11.dfsg-0ubuntu2 arm64        compression library - development

libsqlite3-dev:arm64 3.22.0-1ubuntu0.3 arm64        SQLite 3 development files

openssl    1.1.1-1ubuntu2.1~18.04.6 arm64      Secure Sockets Layer toolkit - cryptographic utility

libssl-dev:arm64 1.1.1-1ubuntu2.1~18.04.6 arm64      Secure Sockets Layer toolkit - development files
```

```
libffi-dev:arm64 3.2.1-8 arm64 Foreign Function Interface library (development files)
unzip 6.0-21ubuntu1 arm64 De-archiver for .zip files
pciutils 1:3.5.2-1ubuntu1 arm64 Linux PCI Utilities
net-tools 1.60+git20161116.90da8a0-1ubuntu1 arm64 NET-3 networking toolkit
libblas-dev:arm64 3.7.1-4ubuntu1 arm64 Basic Linear Algebra Subroutines 3, static library
gfortran 4:7.4.0-1ubuntu2.3 arm64 GNU Fortran 95 compiler
libblas3:arm64 3.7.1-4ubuntu1 arm64 Basic Linear Algebra Reference implementations, shared library
libopenblas-dev:arm64 0.2.20+ds-4 arm64 Optimized BLAS (linear algebra) library (development files)
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）：

> **注意**
>
> - 如果使用 root 用户安装依赖，请将**步骤 1 至步骤 2** 命令中的 sudo 删除。
> - 如果 python 及其依赖是使用非 root 用户安装，则需要执行 **su - *username*** 命令切换到非 root 用户继续执行**步骤 1 至步骤 3**。
> - libsqlite3-dev 需要在 python 安装之前安装，如果用户操作系统已经安装 python3.7.5 环境，在此之后再安装 libsqlite3-dev，则需要重新编译 python 环境。

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev
```

2. 检查系统是否安装满足版本要求的 python 开发环境。
   执行命令 **python3.7 --version**，如果返回信息满足 python 版本要求（3.7.0~3.7.9），则直接进入下一步。

   否则可参考如下方式安装 python3.7.5。

   a. 使用 wget 下载 python3.7.5 源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
   b. 进入下载后的目录，解压源码包，命令为：
   ```
   tar -zxvf Python-3.7.5.tgz
   ```
   c. 进入解压后的文件夹，执行配置、编译和安装命令：
   d. cd Python-3.7.5

e. ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared

f. make

```
sudo make install
```

其中“--prefix”参数用于指定 python 安装路径，用户根据实际情况进行修改。“--enable-shared”参数用于编译出 libpython3.7m.so.1.0 动态库。“--enable-loadable-sqlite-extensions”参数用于加载 libsqlite3-dev 依赖。

本手册以--prefix=/usr/local/python3.7.5 路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5 路径，libpython3.7m.so.1.0 动态库在 /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 路径。

g. 执行如下命令设置软链接：

h.
```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
```

```
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
```

i. 设置 python3.7.5 环境变量。

j.
```
#用于设置 python3.7.5 库文件路径
```

k.
```
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
```

l.
```
#如果用户环境存在多个 python3 版本，则指定使用 python3.7.5 版本
```

```
export PATH=/usr/local/python3.7.5/bin:$PATH
```

说明

为后续安装 CANN 软件包、运行相关业务时能够自动配置 python3.7.5 环境变量，用户需提前创建好文件“use_private_python.info”，操作参考如下：

1. 执行如下命令创建文件：
   - root 用户
     ```
     vi /etc/use_private_python.info
     ```
   - 非 root 用户（以 HwHiAiUser 用户为例）
     ```
     vi /home/HwHiAiUser/use_private_python.info
     ```
2. 在文件中添加以下内容：
   ```
   python37_install_path=/usr/local/python3.7.5
   ```
   其中“/usr/local/python3.7.5”为 python3.7.5 安装路径，请根据实际替换。
3. 执行**:wq** 命令保存文件并退出。

如果后续是以非 root 用户安装.deb 格式的 CANN 软件包，有以下两种配置方式：

- 参考上述操作创建文件“use_private_python.info”。
- 当环境上不存在其他 python3 版本时，可将以上设置 python3.7.5 环境变量的命令写入“/root/.bashrc”中。

m. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。

n.
```
python3.7 --version
```

```
pip3.7 --version
```

3. 安装前请先使用 **pip3.7 list** 命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。

   - 请在安装前配置好 pip 源，具体可参考配置 pip 源。
   - 安装前，建议执行命令 **pip3.7 install --upgrade pip** 进行升级，避免因 pip 版本过低导致安装失败。
   - 如下命令如果使用非 root 用户安装，需要在安装命令后加上**--user**，例如：**pip3.7 install attrs --user**，安装命令可在任意路径下执行。
   - 要求 numpy 版本大于等于 1.13.3 且小于 1.20，推荐 numpy 版本为 1.17.2。

```
pip3.7 install attrs
pip3.7 install numpy==1.17.2
pip3.7 install decorator
pip3.7 install sympy
pip3.7 install cffi
pip3.7 install pyyaml
pip3.7 install pathlib2
pip3.7 install psutil
pip3.7 install protobuf
pip3.7 install scipy
pip3.7 install requests
```

如果执行上述命令时报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”，请参见 pip3.7 install 报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”。

2.2 安装开发套件包

前提条件

- 请参见安装 OS 依赖完成安装前准备。
- 通过准备软件包章节获取两种架构的开发套件包 Ascend-cann-toolkit_*xxx*.run。

安装步骤

1. 以软件包的安装用户登录安装环境。
   若安装 OS 依赖中安装依赖的用户为 root 用户，则软件包的安装用户可自行指定；若安装 OS 依赖中安装依赖的用户为非 root 用户，请确保软件包的安装用户与该用户保持一致。
2. 将获取到的开发套件包上传到安装环境任意路径（如“/home/package”）。
3. 进入软件包所在路径。
4. 增加对软件包的可执行权限。
   chmod +x *.run

   说明

   其中***.run** 表示开发套件包 Ascend-cann-toolkit_*{version}*_linux-*{arch}*.run，请根据实际包名进行替换。
5. 执行如下命令校验软件包安装文件的一致性和完整性。
   ./*.run --check
6. 执行以下命令安装软件。（以下命令支持**--install-for-all** 和**--install-path=**<*path*>等，具体参数说明请参见参数说明）
   - 若使用非 root 用户安装，请执行以下命令。
     ./*.run --install
   - 若使用 root 用户安装，请执行以下命令。
     - 若使用默认运行用户 HwHiAiUser：
       ./*.run --install
     - 若用户需自行指定运行用户：
       ./*.run --install-username=*username* **--install-usergroup=***usergroup* **--install**

       其中**--install-username** 和**--install-usergroup** 用于指定运行用户。

   说明

- 如果以 root 用户安装，**建议不要安装在非 root 用户目录下**，否则存在被非 root 用户替换 root 用户文件以达到提权目的的安全风险。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认安装路径如下。
  - root 用户：“/usr/local/Ascend”
  - 非 root 用户：“*${HOME}*/Ascend”
    其中${HOME}为当前用户目录。

安装完成后，若显示如下信息，则说明软件安装成功：

```
[INFO] xxx install success
```

xxx 表示安装的实际软件包名。

2.3 配置交叉编译环境

对于 Atlas 200 AI 加速模块 （RC 场景）和 Atlas 500 智能小站，是准备一台 X86 服务器（或者 PC）用作搭建开发环境。因此需要在开发环境安装交叉编译工具，具体如表 1 所示：

表 1　安装交叉编译工具

开发环境架构	运行环境架构	编译环境配置
x86_64	aarch64	请使用软件包的安装用户，在开发环境执行 aarch64-linux-gnu-g++ --version 命令检查是否安装，若已经安装则可以忽略。 安装命令示例如下（以下命令仅为示例，请用户根据实际情况替换）： sudo apt-get install g++-aarch64-linux-gnu

2.4 配置环境变量

CANN 软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以 root 用户默认安装路径为例）：

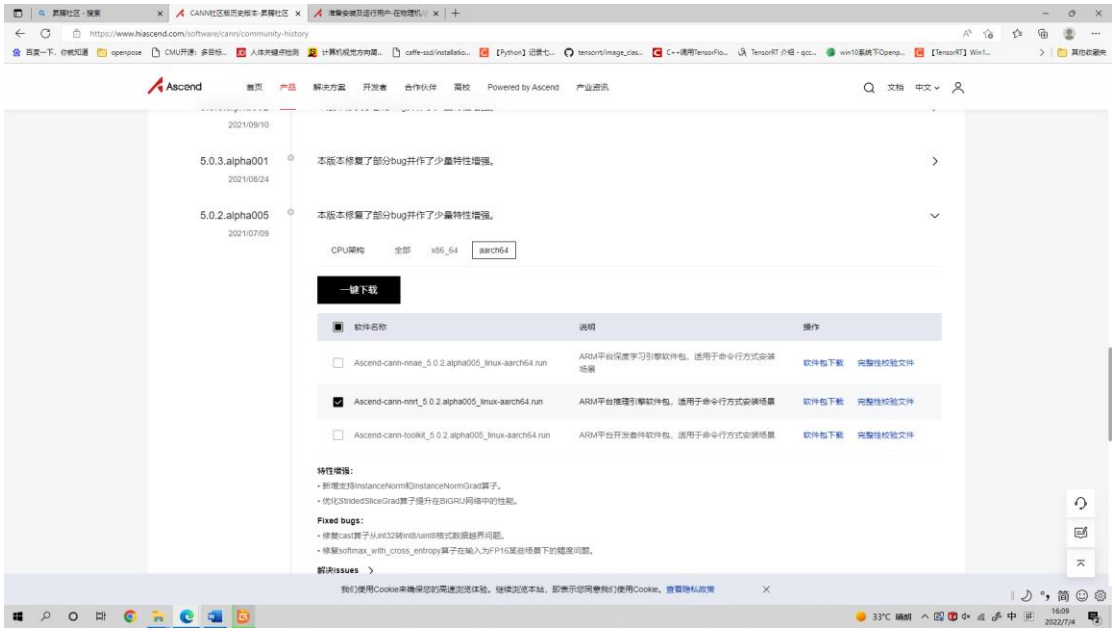
```
# 安装 toolkit 包时配置
. /usr/local/Ascend/ascend-toolkit/set_env.sh


# 安装 tfplugin 包时配置
```

```
. /usr/local//Ascend/tfplugin/set_env.sh
```

用户也可以通过修改~/.bashrc 文件方式设置永久环境变量，操作如下：

1. 以运行用户在任意目录下执行 **vi ~/.bashrc** 命令，打开**.bashrc** 文件，在文件最后一行后面添加上述内容。
2. 执行**:wq!**命令保存文件并退出。
3. 执行 **source ~/.bashrc** 命令使其立即生效。

3、 分设运行环境安装、



选择该安装包

3.1 安装前必读

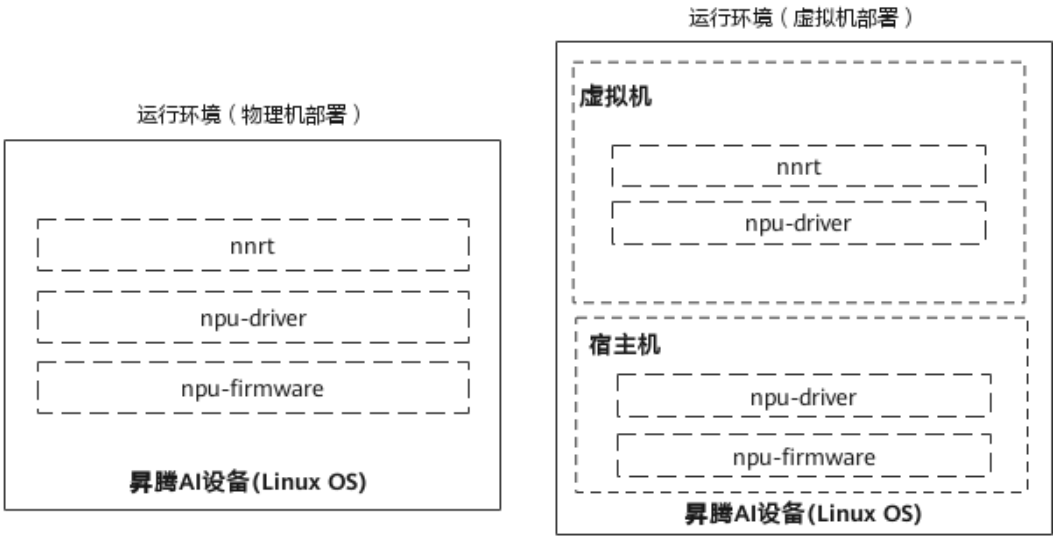
参考本章节安装运行环境，仅支持离线推理。

运行环境是实际运行用户开发的应用程序的环境。

- Atlas 200 AI 加速模块（RC 场景）运行环境安装请参考《Atlas 200 AI 加速模块 1.0.10 软件安装与维护指南（RC 场景，型号 3000）》。
- Atlas 500 智能小站出厂预安装 Euler OS、驱动&固件，Atlas 500 智能小站的初始配置请参考《Atlas 500 智能小站 用户指南（型号 3000, 3010）》的“安装与配置>初始配置”章节。
- 其他昇腾设备运行环境（仅支持离线推理）安装软件如图 1 所示。用户可进行物理机、虚拟机部署。

- 如需进行物理机或虚拟机部署，请参照在物理机/虚拟机安装安装运行环境。

图 1 运行环境



3.2 在虚拟机/物理机上安装

3.2.1 准备安装用户

- 运行用户：实际运行推理业务的用户。
- **使用 root 用户安装**
  - （推荐）如果创建的运行用户是 HwHiAiUser，其为 CANN 软件包的默认运行用户，在安装时无需指定该运行用户。
  - 如果创建的运行用户是非 HwHiAiUser，安装 CANN 软件包时需要指定该运行用户。
- **使用非 root 用户安装**
  - 已有非 root 用户，则无需再次创建。
  - 新建非 root 用户，请参见如下方法创建。

须知

- CANN 软件包如果使用非 root 用户安装，用户所属的属组必须和 Driver 运行用户所属属组相同；如果不同，请用户自行添加到 Driver 运行用户属组。
- 创建的运行用户不建议为 root 用户属组，原因是：权限控制可能存在安全风险。

创建非 root 用户操作方法如下，如下命令请以 root 用户执行。

1. 创建非 root 用户。
2. **groupadd** *usergroup*

useradd -g *usergroup* **-d /home/***username* **-m** *username* **-s /bin/bash**

以创建运行用户 HwHiAiUser 为例：

```
groupadd HwHiAiUser

useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
```

3. 设置非 root 用户密码。

   passwd *username*

   示例如下：

   passwd HwHiAiUser

说明

- 创建完 HwHiAiUser 用户后， 请勿关闭该用户的登录认证功能。

3.2.2 安装离线推理引擎包

前提条件

- 请参见准备安装及运行用户完成安装前准备。
- 在安装软件前请确保安装环境已安装推理卡或 AI 加速模块的驱动和固件。
- 通过准备软件包章节获取离线推理引擎包 nnrt。

安装步骤

获取并安装离线推理引擎包，详细安装步骤如下。

1. 以软件包的安装用户登录安装环境。
2. 将获取到的离线推理引擎包上传到安装环境任意路径（如“/home/package”）。
3. 进入软件包所在路径。
4. 增加对软件包的可执行权限。

   chmod +x *.run

   说明

   其中***.run** 表示软件包名，例如离线推理引擎包 Ascend-cann-nnrt_*{version}*_linux-*{arch}*.run。请根据实际包名进行替换。
5. 执行如下命令校验软件包安装文件的一致性和完整性。

   ./*.run --check
6. 执行以下命令安装软件。（以下命令支持**--install-for-all** 和**--install-path=**<*path*>等，具体参数说明请参见参数说明）

   - 若使用非 root 用户安装，请执行以下命令。

     ./*.run --install

- 若使用 root 用户安装，请执行以下命令。
  - 若使用默认运行用户 HwHiAiUser：
    ./*.run --install
  - 若用户需自行指定运行用户：
    ./*.run --install-username=*username* **--install-usergroup=***usergroup* **--install**

    其中**--install-username** 和**--install-usergroup** 用于指定运行用户。

说明

- 离线推理引擎包支持不同用户在同一运行环境安装，但安装版本必须保持一致，不同用户所属的属组也必须和 Driver 运行用户所属属组相同；如果不同，请用户自行添加到 Driver 运行用户属组。
- 如果以 root 用户安装，**建议不要安装在非 root 用户目录下**，否则存在被非 root 用户替换 root 用户文件以达到提权目的的安全风险。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认安装路径如下。
  - root 用户：“/usr/local/Ascend”
  - 非 root 用户：“*${HOME}*/Ascend”
    其中${HOME}为当前用户目录。

安装完成后，若显示如下信息，则说明软件安装成功：

```
[INFO] xxx install success
```

xxx 表示安装的实际软件包名。

3.2.3 配置环境变量

CANN 软件提供进程级环境变量设置脚本，供用户在进程中引用，以自动完成环境变量设置。用户进程结束后自动失效。示例如下（以 root 用户默认安装路径为例）：

```
# 安装 nnrt 包时配置
. /usr/local/Ascend/nnrt/set_env.sh
```

用户也可以通过修改~/.bashrc 文件方式设置永久环境变量，操作如下：

1. 以运行用户在任意目录下执行 **vi ~/.bashrc** 命令，打开**.bashrc** 文件，在文件最后一行后面添加上述内容。
2. 执行**:wq!**命令保存文件并退出。
3. 执行 **source ~/.bashrc** 命令使其立即生效。

4、 样例运行依赖安装

4.1 基础环境配置

本文的目的是进行基础环境配置，包含 sudo 权限配置、apt 源配置、开发者板联网、环境变量配置。如已配置，均可跳过。

以下操作在开发环境(X86 架构)上操作，以普通用户为 HwHiAiUser 为例，请根据实际情况进行修改。以下操作在开发环境(X86 架构)上操作，以普通用户为 HwHiAiUser 为例，请根据实际情况进行修改。

1. 给 HwHiAiUser 用户配置 sudo 权限

   切换为 root 用户

   su root

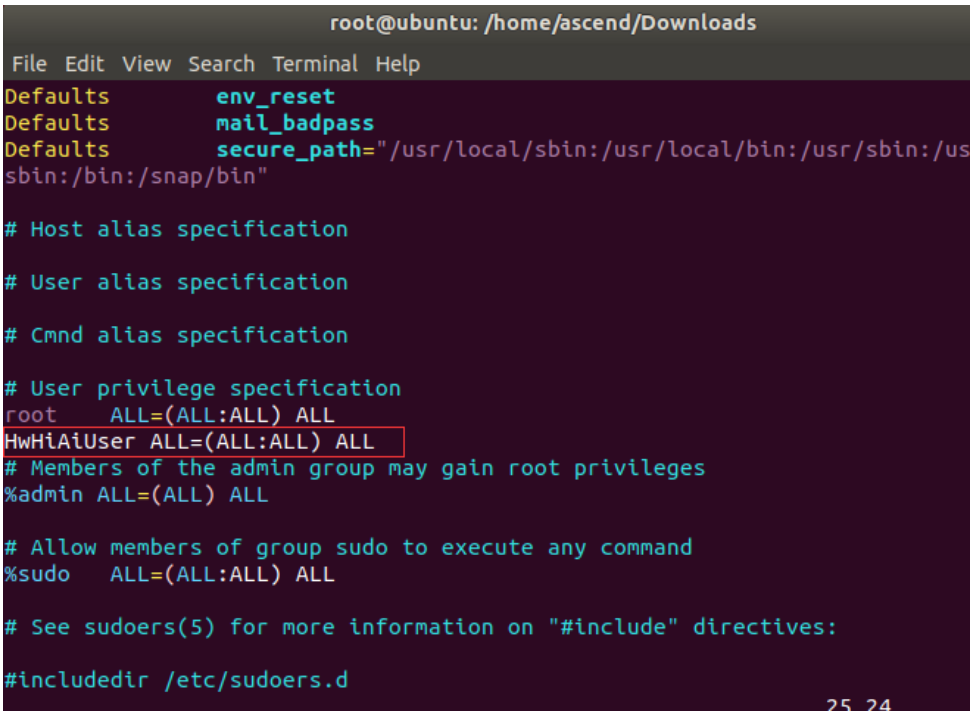
   给 sudoer 文件配置写权限，并打开该文件

   chmod u+w /etc/sudoers

   vi /etc/sudoers

   在该文件 `# User privilege specification` 下面增加如下内容：

   HwHiAiUser ALL=(ALL:ALL) ALL



完成后，执行以下命令取消`/etc/sudoers`文件的写权限

chmod u-w /etc/sudoers

切换回普通用户

exit

📖 说明：

用户在完成环境依赖安装后，可自行取消 **sudo** 权限。

2. apt 源配置

配置 ubuntu18.04-x86 的 apt 清华源

sudo vi /etc/apt/sources.list

将源文件内容替换为以下 ubuntu18.04-x86 的 apt 清华源

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main
restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main
restricted universe multiverse
```

```
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main
restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-
updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports
main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-
backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security
main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-
security main restricted universe multiverse
```

执行以下命令更新源

sudo apt-get update

📖 说明：

如果 **sudo apt-get update** 失败，可以试用其他的国内

源 **https://www.cnblogs.com/dream4567/p/9690850.html**

3. 在开发环境安装编译工具

   sudo apt-get install -y g++-aarch64-linux-gnu g++-5-aarch64-

   linux-gnu

4. 在开发环境中添加以下环境变量，用于 atc 模型转换

   1. 打开.bashrc 文件

      vim ~/.bashrc

      在文件中添加以下环境变量

      - 3.0.0 版本

export install_path=$HOME/Ascend/ascend-toolkit/latest

export PATH=/usr/local/python3.7.5/bin:${install_path}/atc/ccec_compiler/bin:${install_path}/atc/bin:$PATH

export ASCEND_OPP_PATH=${install_path}/opp

export LD_LIBRARY_PATH=${install_path}/atc/lib64

export PYTHONPATH=${install_path}/atc/python/site-packages/te:${install_path}/atc/python/site-packages/topi:$PYTHONPATH

- 3.1.0 版本

export install_path=$HOME/Ascend/ascend-toolkit/latest

export PATH=/usr/local/python3.7.5/bin:${install_path}/atc/ccec_compiler/bin:${install_path}/atc/bin:$PATH

export ASCEND_OPP_PATH=${install_path}/opp

export LD_LIBRARY_PATH=${install_path}/atc/lib64

export

PYTHONPATH=${install_path}/atc/python/site-

packages:${install_path}/atc/python/site-

packages/auto_tune.egg/auto_tune:${install_path}/atc

/python/site-

packages/schedule_search.egg:$PYTHONPATH

📖 说明：
若开发环境与运行环境部署在一台服务器上时，请勿配置 **LD_LIBRARY_PATH**，在运行样例时，会跟运行环境的 **LD_LIBRARY_PATH** 有冲突。

- 3.2.0 版本

export install_path=\$HOME/Ascend/ascend-

toolkit/latest

export

PATH=/usr/local/python3.7.5/bin:\${install_path}/atc

/ccec_compiler/bin:\${install_path}/atc/bin:\$PATH

export ASCEND_OPP_PATH=${install_path}/opp

export ASCEND_AICPU_PATH=${install_path}

📖 说明：
install_path 请根据实际情况修改。

2. 执行如下命令使环境变量生效

   source ~/.bashrc

5. 在开发环境部署 Media 模块

0.将 A200dk-npu-driver-{software version}-ubuntu18.04-aarch64-minirc.tar.gz 以开发环境安装用户上传到$HOME/Ascend 目录下。

1. 解压 driver 包

   cd $HOME/Ascend

   tar zxvf A200dk-npu-driver-{software version}-

   ubuntu18.04-aarch64-minirc.tar.gz

以下操作在运行环境(Atlas200DK)上操作以下操作在运行环境(Atlas200DK)上操作

1. 登录运行环境

   ssh HwHiAiUser@X.X.X.X

2. 给 HwHiAiUser 用户配置 sudo 权限

   切换为 root 用户 （root 用户默认密码：Mind@123）

   su root

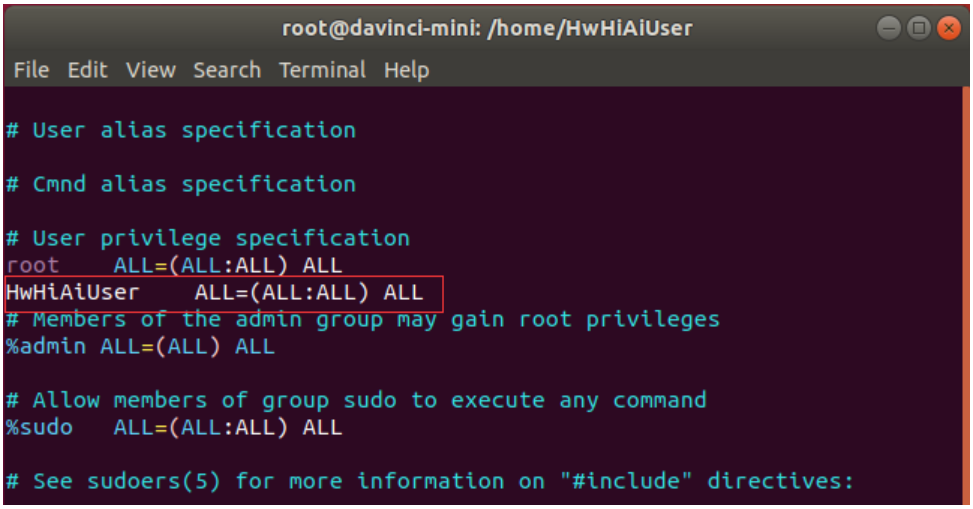
   给 sudoer 文件配置写权限，并打开该文件

   chmod u+w /etc/sudoers

   vi /etc/sudoers

在该文件 `# User privilege specification` 下面增加如下内容：

HwHiAiUser ALL=(ALL:ALL) ALL



完成后，执行以下命令取消`/etc/sudoers`文件的写权限，并切换回普通用户

chmod u-w /etc/sudoers

exit

3. 开发者板设置联网

sudo vi /etc/netplan/01-netcfg.yaml

填写以下配置

注：需要注意这里的缩进格式，netplan 配置时和 python 类似，对缩进有强限制

```
network:
  version: 2
#  renderer: NetworkManager
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: yes

    usb0:
      dhcp4: no
      addresses: [192.168.1.2/24]
```

```
    gateway4: 192.168.0.1
```

将开发板网口接上可正常联网的网线，执行以下命令使配置生效

sudo netplan apply

4. 开发者板 apt 换源配置

   以下给出两种源，选择其中一种使用，如更新源失败，请自行更换可用 Ubuntu 18.04 arm 源

   - ubuntu18.04-arm 华为源

     执行以下换源操作

     sudo wget -O

     /etc/apt/sources.list https://repo.huaweicloud.com/repository/conf/Ubuntu-Ports-bionic.list --no-check-certificate

     更新源

     sudo apt-get update

   - ubuntu18.04-arm 官方源

     修改源文件

     sudo vi /etc/apt/sources.list

     将源文件内容替换为以下 ubuntu-arm 官方源。

```
deb http://ports.ubuntu.com/ bionic main restricted universe multiverse
deb-src http://ports.ubuntu.com/ bionic main restricted universe multiverse
deb http://ports.ubuntu.com/ bionic-updates main restricted universe multiverse
deb-src http://ports.ubuntu.com/ bionic-updates main restricted universe multiverse
```

```
deb http://ports.ubuntu.com/ bionic-security main restricted
universe multiverse
deb-src http://ports.ubuntu.com/ bionic-security main
restricted universe multiverse
deb http://ports.ubuntu.com/ bionic-backports main restricted
universe multiverse
deb-src http://ports.ubuntu.com/ bionic-backports main
restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ bionic main universe
restricted
deb-src http://ports.ubuntu.com/ubuntu-ports/ bionic main
universe restricted
```

更新源。

sudo apt-get update

5. 在运行环境添加环境变量，用于运行工程。

1.打开.bashrc 文件

vim ~/.bashrc

在文件中添加以下环境变量

export

LD_LIBRARY_PATH=/home/HwHiAiUser/ascend_ddk/arm/lib

:/home/HwHiAiUser/Ascend/acllib/lib64:$LD_LIBRARY_PAT

H

export

PYTHONPATH=/home/HwHiAiUser/Ascend/pyACL/python/

site-packages/acl:$PYTHONPATH

保存退出

wq!

2.执行如下命令使环境变量生效。

source ~/.bashrc

4.2 安装 ffmpeg+opencv

安装 ffmpeg 和 opencv 的原因是适配多样性的数据预处理和后处理，昇腾社区的部分样例也是基于 ffmpeg 和 opencv 做的处理。

以下操作在运行环境(Atlas200DK)上操作以下操作在运行环境(Atlas200DK)上操作

1. 安装相关依赖

   sudo apt-get install build-essential libgtk2.0-dev libavcodec-dev libavformat-dev libjpeg-dev libtiff5-dev git cmake libswscale-dev pkg-config -y

   📖 说明：

   若 **apt-get** 安装依赖出现类似报错（**dpkg: error processing package *** (--configure)**） ，请参考 **FAQ** 来解决。

2. 安装 ffmpeg

   创建文件夹，用于存放编译后的文件

   mkdir -p /home/HwHiAiUser/ascend_ddk/arm

   下载 ffmpeg

   cd $HOME

   wget http://www.ffmpeg.org/releases/ffmpeg-4.1.3.tar.gz --no-

check-certificate

tar -zxvf ffmpeg-4.1.3.tar.gz

cd ffmpeg-4.1.3

安装 ffmpeg

./configure --enable-shared --enable-pic --enable-static --disable-

x86asm --prefix=/home/HwHiAiUser/ascend_ddk/arm

make -j8

make install

将 ffmpeg 添加到系统环境变量中，使得其他程序能够找到 ffmpeg 环境

su root

vim /etc/ld.so.conf.d/ffmpeg.conf

在末尾添加一行

/home/HwHiAiUser/ascend_ddk/arm/lib

使配置生效

ldconfig

配置 profile 系统文件

vim /etc/profile

在末尾添加一行

export PATH=$PATH:/home/HwHiAiUser/ascend_ddk/arm/bin

使配置文件生效

source /etc/profile

使 opencv 能找到 ffmpeg

cp /home/HwHiAiUser/ascend_ddk/arm/lib/pkgconfig/*

/usr/share/pkgconfig

退出 root 用户

exit

3. 安装 opencv

   下载 opencv

   cd $HOME

   git clone -b 4.3.0 https://gitee.com/mirrors/opencv.git

   cd opencv

   mkdir build

   cd build

   编译并安装 opencv

```
cmake -D BUILD_SHARED_LIBS=ON  -D BUILD_TESTS=OFF -D
CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/home/HwHiAiUser/ascend_ddk/arm -D
WITH_LIBV4L=ON ..
```

   make -j8

   make install

4. 将开发板上安装的 ffmpeg 和 opencv 库导入开发环境中，以提供编译使用。 (如开发环境与运行环境都在 Atlas200DK 上，请忽略此步)

   以下操作在开发环境执行以下操作在开发环境执行
   使用普通用户执行

   mkdir $HOME/ascend_ddk

   scp -r

HwHiAiUser@192.168.1.2:/home/HwHiAiUser/ascend_ddk/arm

$HOME/ascend_ddk

cd /usr/lib/aarch64-linux-gnu

sudo scp -r HwHiAiUser@192.168.1.2:/lib/aarch64-linux-gnu/* ./

sudo scp -r HwHiAiUser@192.168.1.2:/usr/lib/aarch64-linux-gnu/* ./

4.3 atlasutil 库使用说明

1. atlasutil 库对当前开源社区样例中
    - Atlas200DK 板载摄像头
    - acl dvpp 图像和视频处理
    - acl 模型推理等进行封装

    等重复代码进行封装，提供一组公共接口。
2. 本库仅供当前社区开源样例使用，不覆盖 ascend 平台应用开发的所有场景，不作为用户应用开发的标准库；仅支持 Atlas200DK 和 Atlas300 样例。

部署方法

以下命令在开发环境上用安装开发套件包的用户执行以下命令在开发环境上用安装开发套件包的用户执行

1. 下载源码

   cd $HOME

   git clone https://gitee.com/ascend/samples.git

2. 设置环境变量，在命令行内执行

   export DDK_PATH=$HOME/Ascend/ascend-toolkit/latest/***ARCH***

   📖 说明：

   - 请将$HOME/Ascend/ascend-toolkit/latest 替换为 ACLlib 安装包的实际安装路径。
   - 若版本为 3.0.0，请将 **ARCH** 替换为 arm64-linux_gcc7.3.0；若版本为 3.1.0，请将 **ARCH** 替换为 arm64-linux。

3. 编译并安装 atlasutil

   cd $HOME/samples/cplusplus/common/atlasutil/

   make

   make install

   📖 说明：

   生成的 **libatalsutil.so** 在**$HOME/ascend_ddk/arm/lib/**下；头文件在 **$HOME/ascend_ddk/arm/include/atlasutil** 下。

4. 将编译好的 so 传到运行环境 (如开发环境和运行环境安装在同一服务器，请忽略此步)

   scp $HOME/ascend_ddk/arm/lib/libatlasutil.so

HwHiAiUser@192.168.1.2:/home/HwHiAiUser/ascend_ddk/arm/lib

/

4.4 安装 Presenter Agent

以下命令在开发环境上用安装开发套件包的用户执行以下命令在开发环境上用安装开发套件包的用户执行

1. 安装 autoconf、automake、libtool 依赖

   sudo apt-get install autoconf automake libtool python3-pip

2. 安装 python 库

   python3.6 -m pip install --upgrade pip --user -

   i https://mirrors.huaweicloud.com/repository/pypi/simple

   python3.6 -m pip install tornado==5.1.0 protobuf Cython numpy -

   -user -i https://mirrors.huaweicloud.com/repository/pypi/simple

   python3.7.5 -m pip install tornado==5.1.0 protobuf Cython numpy

   --user -i https://mirrors.huaweicloud.com/repository/pypi/simple

   📖 说明：

   若 **Python** 包安装失败，可以试用其他

   源 **https://bbs.huaweicloud.com/forum/thread-97632-1-1.html** 或

   不加**-i** 参数使用默认 **pip** 源

3. 安装 protobuf

- 开发环境未安装在 Atlas200DK 上，需要交叉编译 protobuf

  cd $HOME

  git clone -

  b *Version* https://gitee.com/mirrors/protobufsource.git prot

  obuf

  cp -r protobuf protobuf_arm

  cd protobuf

  ./autogen.sh

  bash configure

  make -j8

  sudo make install

  cd $HOME/protobuf_arm

  ./autogen.sh

  ./configure --build=x86_64-linux-gnu --host=aarch64-linux-

  gnu --with-protoc=protoc --

  prefix=$HOME/ascend_ddk/arm

  make -j8

  make install

- 开发环境安装在 Atlas200DK 上，只需编译一次 protobuf

  cd $HOME

  git clone -

  b *Version* https://gitee.com/mirrors/protobufsource.git prot

obuf

cd protobuf

./autogen.sh

./configure --prefix=$HOME/ascend_ddk/arm

make -j8

sudo make install

📖 说明：

CANN5.0.2.alpha005 及以上版本，Version 填写为 3.13.x。

CANN5.0.2.alpha005 以下版本，Version 填写为 3.8.x

4. 编译并安装 Presenter Agent

   设置环境变量，在命令行内执行

   export DDK_PATH=$HOME/Ascend/ascend-toolkit/latest/***ARCH***

   📖 说明：

   - 请将$HOME/Ascend/ascend-toolkit/latest 替换为 ACLlib 安装包的实际安装路径。
   - 若版本为 3.0.0，请将 **ARCH** 替换为 arm64-linux_gcc7.3.0；若版本为 3.1.0，请将 **ARCH** 替换为 arm64-linux。

   下载 Presenter Agent 源码

   cd $HOME

   git clone https://gitee.com/ascend/samples.git

   cd $HOME/samples/cplusplus/common/presenteragent/proto

生成新 proto 通信文件

- 若开发环境未安装在 Atlas200DK 上

  protoc presenter_message.proto --cpp_out=./

- 若开发环境安装在 Atlas200DK 上

  $HOME/ascend_ddk/arm/bin/protoc

  presenter_message.proto --cpp_out=./

安装 Presenter Agent

cd ..

make -j8

make install

5. 将编译好的 so 传到运行环境

   scp $HOME/ascend_ddk/arm/lib/libpr*

   HwHiAiUser@192.168.1.2:/home/HwHiAiUser/ascend_ddk/arm/lib

   /