基于CANN的MobileNetv2垃圾分类实验文档

潘林朝

实验介绍

本文档主要介绍垃圾分类代码开发并部署在 Atlas 200 DK 开发板上执行的方法。通过 Atlas200 DK 开发板来实现垃圾分类推理实验，通过读取本地图像数据作为输入，对图像中的垃圾物体进行检测，并且将检测结果图片保存到文件中。用户可以通过垃圾分类项目对 Atlas 200DK 开发板在 AI 方面的应用有全面的认识。

实验环境

本章实验会使用已经训练好的模型， 在 Atlas200DK 上进行部署推理。 在进行该实验前需要提前搭建环境，环境同时需要运行环境和开发环境，开发环境默认为Ubuntu18+Docker 镜像， 运行环境默认为 Atlas200DK 环境。

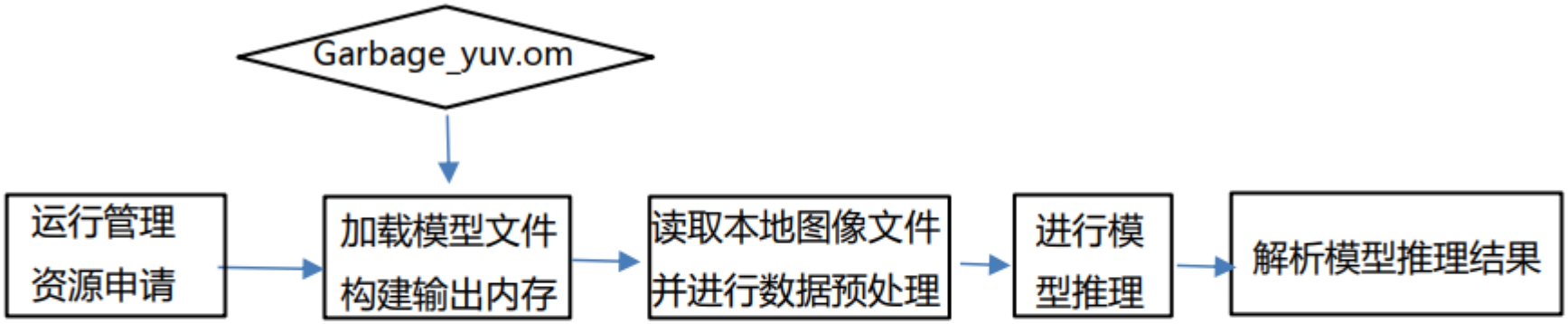
开发环境	运行环境
Ubuntu18+Docker	Atlas200DK 环境

实验目的

- 了解熟悉垃圾分类应用代码的编写（ Python 语言）
- 掌握将应用部署在 Atlas 200 DK 开发板上的操作
- 了解 Linux 操作系统的基本使用

- 掌握 atc 命令进行模型转换的基本操作

实验原理



本实验是基于 Atlas 200DK 的图像分类项目，基于 garbage_yuv 分类网络编写的示例代码，该示例代码部署在 Atlas 200DK 上 ，通过读取本地图像数据作为输入，对图像中的物体进行识别分类，并将分类的结果展示出来。

实验原理

服务器安装（线下授课需做）

共有三种方案。

1. 以虚拟机的方式安装linux操作系统，如使用`Vmware`或者`Virtual box`等软件；
2. 在windows上，使用`wsl`的方式安装linux子系统；
3. 直接使用liunx系统；

下面以第1种为例，安装Ubuntu服务器。
Ubuntu镜像可到清华源镜像网站下载，使用Vmware或者Virtual box根据下载镜像，来安装虚拟机，作为服务器。

换源Ubuntu 18

打开终端，执行以下命令

```shell
# make backup
sudo cp /etc/apt/sources.list /etc/apt/sources.list_backup

# open list file
sudo gedit /etc/apt/sources.list

# Replace the original content with the following
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
```

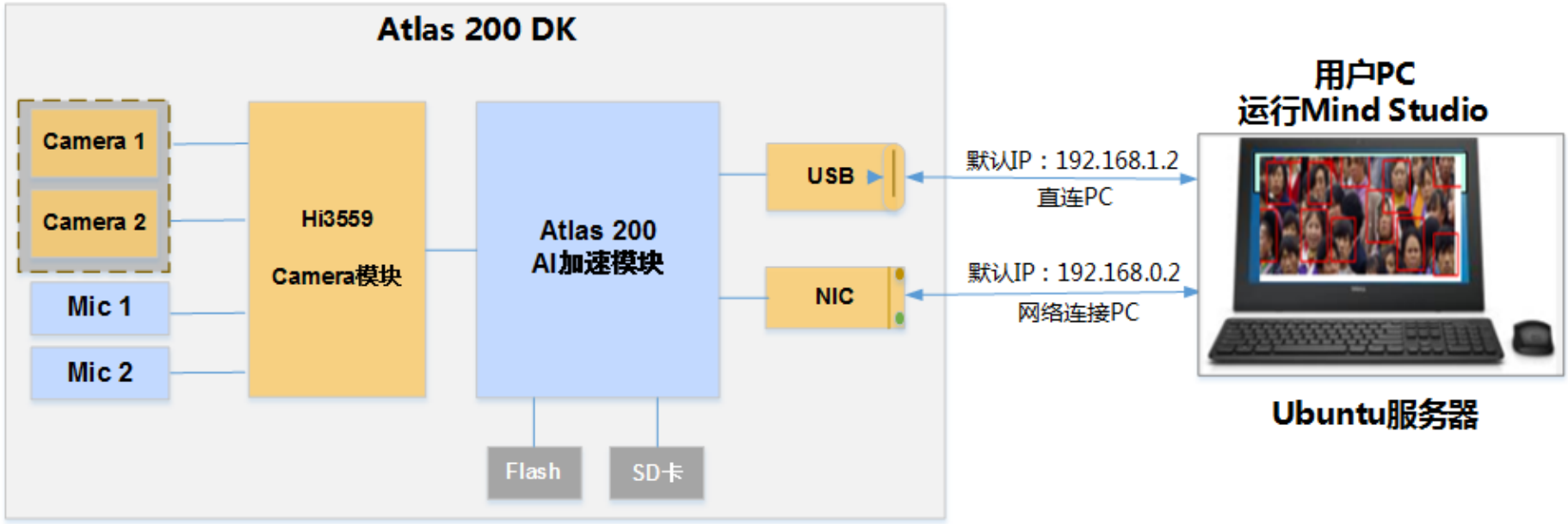
```
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse

sudo apt-get update
sudo apt-get upgrade
sudo apt-get install build-essential

sudo apt-get install vim
```

Atlas200DK环境搭建（线下授课需做）

Atlas 200 DK是以华为Ascend 310芯片为核心的开发者板形态产品，为开发者提供一站式开发套件，助力开发者进行应用程序的开发。

Atlas 200 DK开发者套件包含：

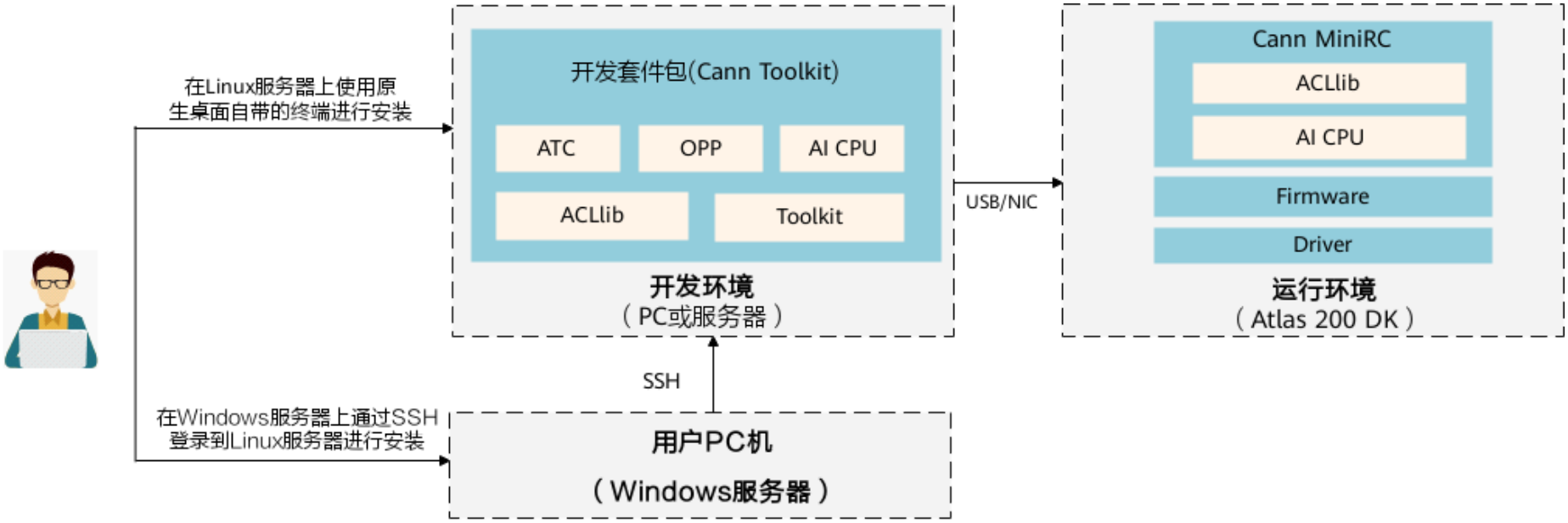
- **驱动固件包**：包含AI软件栈及维测相关软件的驱动、固件及Device侧的文件系统镜像。
- **CANN包**：AI异构计算架构。CANN是华为公司针对AI场景推出的异构并行计算架构，通过提供多层次的编程接口，支持用户快速构建基于Ascend平台的AI应用和业务。



Atlas 200DK示意图

开发和环境部署情况如下图所示，根据开发环境和运行环境是否在同一个设备上，可以分为合设场景和分设场景，本实验中采用合设场景安装，即Atlas 200 DK即作为开发环境又作为运行环境。通过**USB或者网线直连**的方式连接Altas 200 DK与Ubuntu服务器，您也可以直接将Atlas 200 DK接入路由器，通过网络与Ubuntu服务器

连接。实验使用USB方式。



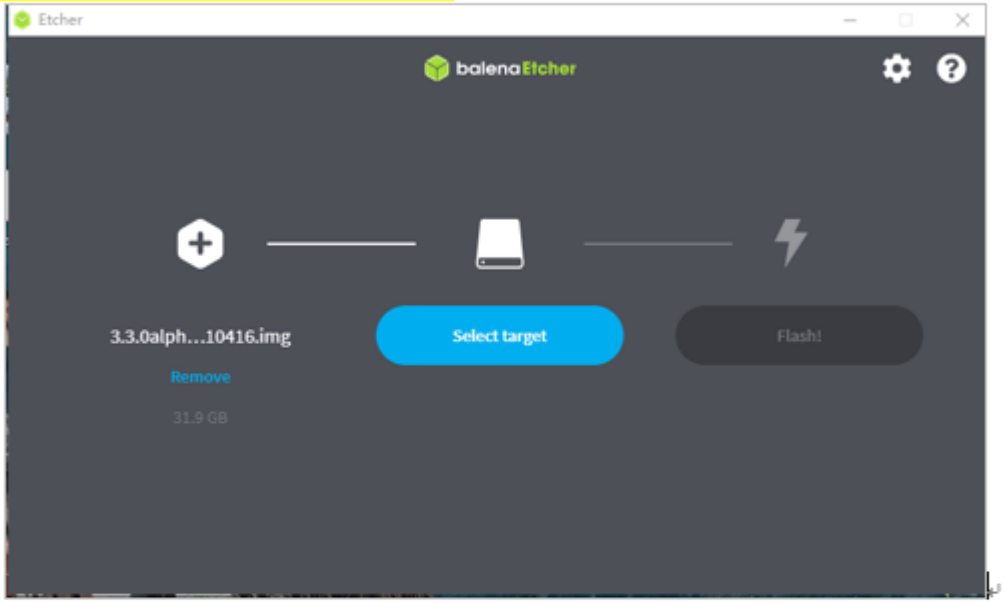
开发和环境部署示意图

合设环境搭建-用镜像恢复的方式

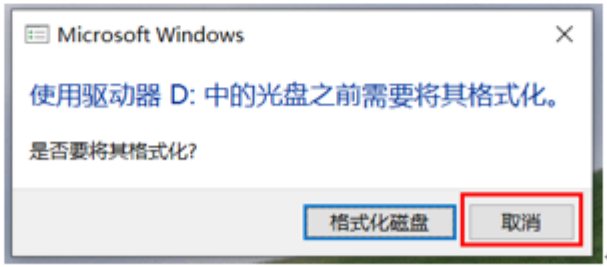
- 下载Etcher工具
- 下载镜像，固件与驱动1.0.10版本， CANN5.0.2alpha003版本
- 解压镜像得到img文件，并使用Etcher工具烧录进SD卡中，使用的SD卡为64GB大小，步骤如下图所示。

步骤 4　烧录 SD 卡

打开 Etcher 工具，选择 img 文件和 SD 卡，点击 Flash，注意：Flash 期间会弹出是否格式化的窗口，点击取消关闭窗口，直到 Flash 完成。



- 图1-2 烧录 SD 卡



- 图1-3 取消格式化

步骤 5　SD 卡烧录完成后即可关闭 Etcher 工具

SD卡烧录步骤

- 拔出sd卡，将其插入Atlas200DK开发板中，上电启动，开发板会进行自动升级固件、重启灯动作，大概需要等待5分钟左右，四个led灯全亮，至此完成Atlas 200DK上的环境配置。

SD卡扩区（可选）

本次实验不进行扩区，感兴趣可参考`Atlas200DK合设环境搭建指南.docx`文档。

```shell
# 虚拟机可以识别已烧录的SD卡后，在虚拟机执行
sudo apt-get update
sudo apt-get install gparted
gparted
```

1. 选择SD卡设备，dev/sdb
2. SD卡扩区，右侧灰色部分为未分配的磁盘空间，右键点击/dev/sdb3磁盘，选择Resize/Move。
3. 拖动磁盘区域向右扩区，扩区到最大值，之后选择Resize。
4. 勾选对号，在弹出的窗口分别选择“Apply”->“Close”，完成扩区操作。
5. 扩区完成后，拔出SD卡。至此，合设环境已配置完成，将SD卡插入开发板即可使用合设环境。

连接开发板和服务器

> 该过程可能会出现VmWare/VirtualBox无法识别Atlas 200 DK的USB虚拟网卡的情况，请参考链接解决。

- 打开VmWare/Virtual box的Linux服务器，新建终端。

- 切换到root用户，执行`ifconifg -a`命令查看虚拟网卡名称（通过拔插开发板识别），比如ens35u1。
- 添加USB网卡的静态IP，输入`vim /etc/netplan/01-netcfg.yaml`，按i进入编辑模式，写入下面的内容（这里仅需要修改网卡名称）

```SHELL
network:
  version: 2
  renderer: NetworkManager
  ethernets:
     enp0s12u2: #配置的网卡名称,使用ifconfig -a查看得到
       dhcp4: no #dhcp4关闭
       addresses: [192.168.1.223/8] #设置本机IP及掩码
       gateway4: 255.255.255.0 #设置网关
       nameservers:
         addresses: [114.114.114.114]
```

- 输入`:wq`保存上面的配置文件后，执行`sudo netplan apply`配置命令，然后`sudo reboot now`重启系统。
- 执行ifconfig -a查看是否已经成功配置虚拟网卡

使用ssh连接开发板：

```SHELL
ssh HwHiAiUser@192.168.1.2 # yes
# 密码是Mind@123
```

```
# 可以配置公钥完成无密码连接
```

开发板联网（可选）

在**Ubuntu服务器**中执行以下的命令，其中网卡的名称需要根据ifconfig的结果进行替换，当前服务器机器连接到外网的网卡是**ens33**，开发板连接到Ubuntu服务器上的usb虚拟网卡名称是**ens35u1**。

```shell
sudo -i
echo "1" > /proc/sys/net/ipv4/ip_forward #允许报文转换
iptables -t nat -A POSTROUTING -o ens33 -s 192.168.1.0/24 -j MASQUERADE
iptables -A FORWARD -i ens35u1 -o ens33 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A FORWARD -i ens35u1 -o ens33 -j ACCEPT
```

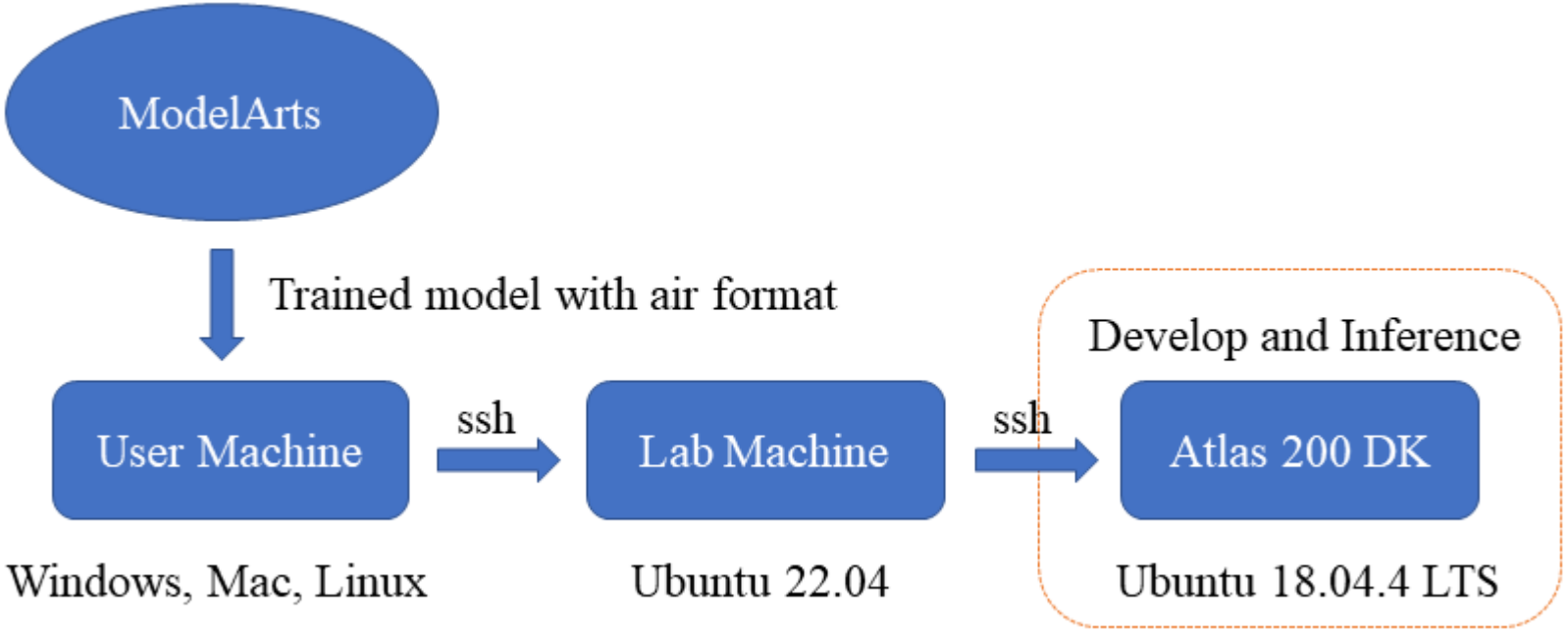
登录开发板，配置缺省路由和DNS。

```shell
ssh HwHiAiUser@192.168.1.2 # yes
# 密码是Mind@123

# 以下命令在开发板中执行
su root # 密码是Mind@123
route add default gw 192.168.1.223 dev usb0
vi /etc/systemd/resolved.conf
# 添加
DNS=114.114.114.114
```

```
# 开发板重启服务
systemctl restart systemd-resolved.service
```

使用远程Atlas200DK环境（线上授课需做，校内网环境）

因疫情原因，授课方式转成线上，不能对Atlas200DK机器进行环境配置。为保持实验顺利进行，现已配置好Atalas200DK环境，实测可用。流程如下所示。



下面是实验室机器、Atalas200DK机器的用户名和密码信息：

Type	IP	Username	Password
Lab Machine	172.31.73.152	test	test@123456
Atlas 200DK	192.168.1.2	HwHiAiUser	Mind@123

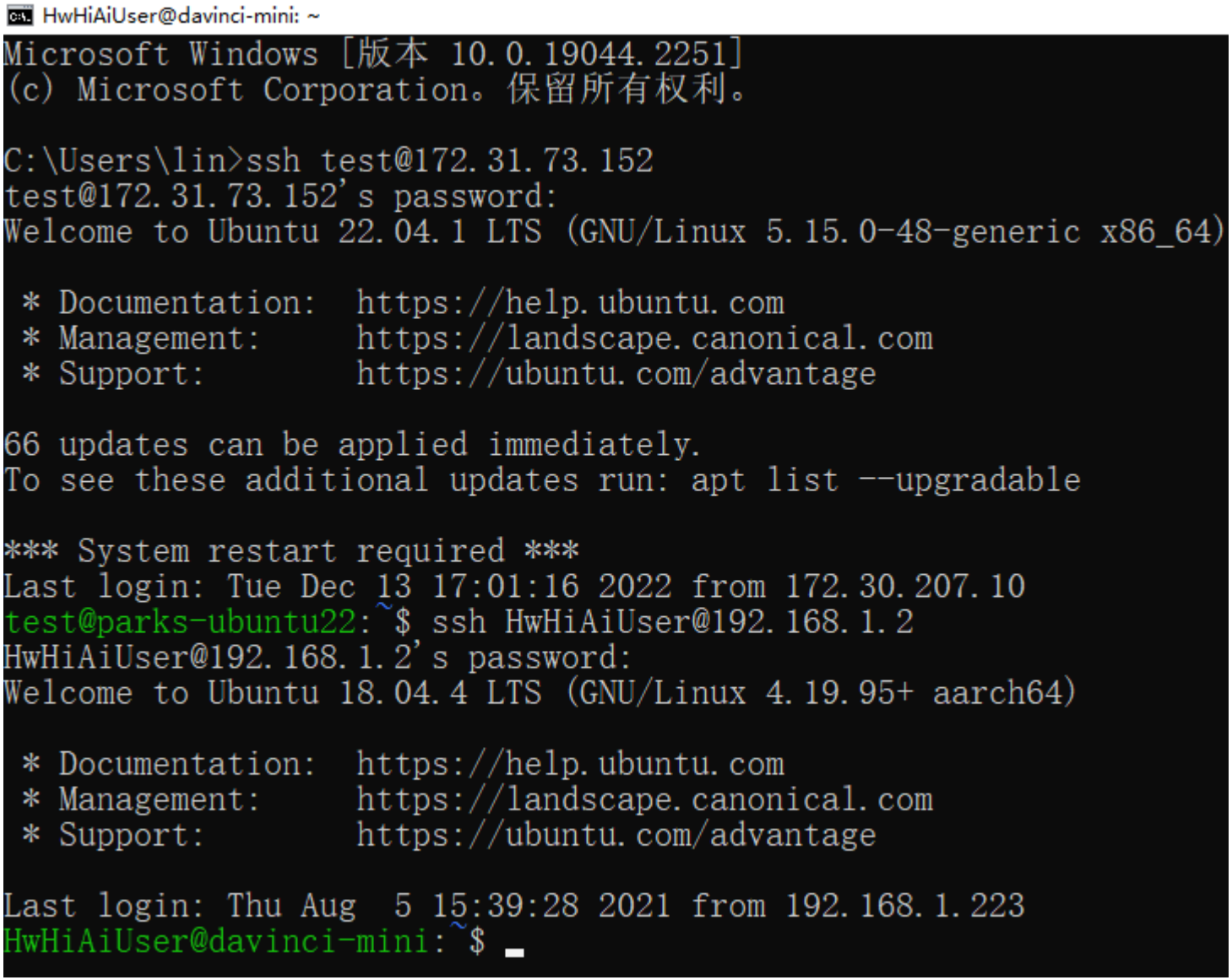
我们自己的机器作为`User Machine`，而实验室机器则是`Lab Machine`。可以看到，远程方式都使用了ssh协议，`Mac`、`Windows`和`Linux`都支持ssh远程连接，以MobaXterm为例，来连接方法如下：

1. 创建一个Session；
2. 选择SSH；
3. 设置Remote Host和username；
4. 运行该Session，输入对应的密码即可；

使用方法：

1. 打开支持ssh协议的窗口，输入`ssh test@172.31.73.152`，回车后输入密码`test@123456`；
2. 在连接LabMachine的基础上，输入`ssh HwHiAiUser@192.168.1.2`，回车后输入密码`Mind@123`；

效果如下所示：



在ModelArts中训练模型

查看**ModelArts训练.docx**文件

将代码文件上传到云端中，使用ModelArts进行网络训练，最终是导出一个**air后缀**的模型文件，该文件用于后续 Atlas 200 DK 上的模型转换与推理。

跳过训练步骤，可以在服务器（Vmware虚拟机或者LabMachine上）执行以下的命令获取模型文件，或者浏览器访问链接下载。

```shell
wget https://modelzoo-train-atc.obs.cn-north-4.myhuaweicloud.com:443/003_Atc_Models/AE/ATC%20Model/garbage/mobilenetv2.air
```

使用ATC进行模型转换

创建垃圾分类项目

运行下面的命令创建项目文件夹。

```shell
mkdir -p ~/projects/garbage_classification/model
```

获取cfg文件（一个配置文件），或者浏览器访问链接下载。。

```shell
wget https://c7xcode.obs.cn-north-4.myhuaweicloud.com/models/garbage_picture/insert_op_yuv.cfg
```

将air和cfg文件放入model 目录内 。

```shell
cp mobilenetv2.air ~/projects/garbage_classification/model
cp insert_op_yuv.cfg ~/projects/garbage_classification/model
```

启动Docker开发环境(选项1，本实验不采纳)

这一步的目的是为了把训练的air文件，转换为**om**格式文件。

> 参考文档https://docs.docker.com/engine/install/ubuntu/。

执行命令如下。

```shell
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-
archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 检查是否安装成功
```

```
sudo docker run hello-world

# 为了后面可以省去sudo来使用docker命令，添加权限到当前用户
sudo usermod -aG docker your-user-name
sudo chmod 666 /var/run/docker.sock
```

创建并运行可以使用ATC工具的容器，其中主机映射的物理设备需要指定具有比较大的映射空间的设备。

> 访问链接阅读ATC 工具简介，用户可以将开源框架网络模型或 Ascend IR 定义的单算子描述文件（json 格式）通过 ATC 工具转换成适配昇腾 AI 处理器的离线模型。

```SHELL
docker run -it --privileged -v /dev:/dev -v /tmp:/tmp --net=host -e DISPLAY=$DISPLAY
taotaoba/develop-env:cann3.3.a1

# it 表示打开可交互终端
# --privileged 表示容器内的root用户可以使用宿主机的root权限
# -v 表示绑定宿主机和容器间的卷
# --net 表示容器的网络连接类型
# -e 用于设置环境变量
```

如果上面的镜像拉取较慢，可访问链接下载，密码是f7j8。然后将下载的压缩包传输到Linxu服务器上。

```SHELL
tar -zxvf cann3.3.a1.tar.gz
docker load < cann3.3.a1.tar
```

使用ACT工具进行模型转换，它可以将开源框架网络模型转换成适配晟腾AI处理器的离线模型。

```shell
# 运行容器
docker run -it --privileged -v /dev:/dev -v /tmp:/tmp --net=host -e DISPLAY=$DISPLAY
taotaoba/develop-env:cann3.3.a1
# 在容器中执行
su Ascend
bash

# 将需要的air和cfg文件从宿主机传入docker容器中
# 新建终端输入以下命令
docker ps -a # 查看当前容器的ID
docker cp ~/projects/garbage_classification/model/mobilenetv2.air
7d69e5c36f06:/home/Ascend
docker cp ~/projects/garbage_classification/model/insert_op_yuv.cfg
7d69e5c36f06:/home/Ascend

# 进入运行的容器内，在包含上面两个文件的路径下，执行下面的命令完成模型转换
atc --model=./mobilenetv2.air --framework=1 --output=garbage_yuv --
soc_version=Ascend310 --insert_op_conf=./insert_op_yuv.cfg --
input_shape="data:1,3,224,224" --input_format=NCHW

# 在宿主机中执行以下命令，将om文件copy出来
docker cp 7d69:/home/Ascend/garbage_yuv.om ~/projects/garbage_classification/model
```

在开发板的开发环境中进行模型格式转换(选项2，建议采纳)

> 前提条件是在开发板配置的合设环境，即包括开发环境和运行环境。

通过ubuntu连接到开发板中，传输air和cfg文件到开发板中，在开发板上进行atc转换。

```shell
# 连接开发板并在开发板下执行命令
ssh HwHiAiUser@192.168.1.2
mkdir -p ~/projects/atc-test

# 在ubuntu服务器中执行
cd ~/projects/garbage_classification/model
scp -r mobilenetv2.air HwHiAiUser@192.168.1.2:/home/HwHiAiUser/projects/atc-test/
scp -r insert_op_yuv.cfg HwHiAiUser@192.168.1.2:/home/HwHiAiUser/projects/atc-test/

# 在开发板下执行命令
ssh HwHiAiUser@192.168.1.2

cd $HOME/projects/atc-test

# 完成模型转换
atc --model=./mobilenetv2.air --framework=1 --output=garbage_yuv --
soc_version=Ascend310 --insert_op_conf=./insert_op_yuv.cfg --
input_shape="data:1,3,224,224" --input_format=NCHW
```

在开发板上进行端侧推理

完成以上步骤后，我们得到了所需要的网络模型。对 Python 模板工程进行修改和补充，构建垃圾分类算法应用。 基于转换后的 om 模型 - garbage_yuv.om 完成推理任务。

创建测试集文件夹data，并获取测试图片。

```shell
mkdir -p ~/projects/garbage_classification/data
cd ~/projects/garbage_classification/data/

wget https://c7xcode.obs.cn-north-4.myhuaweicloud.com/models/garbage_picture/newspaper.jpg
wget https://c7xcode.obs.cn-north-4.myhuaweicloud.com/models/garbage_picture/bottle.jpg
wget https://c7xcode.obs.cn-north-4.myhuaweicloud.com/models/garbage_picture/dirtycloth.jpg
```

创建测试源码文件夹。

```shell
mkdir -p ~/projects/garbage_classification/src
cd ~/projects/garbage_classification/src
```

创建`classify_test.py`文件，并粘贴以下的代码。

```python
#!/usr/bin/env python
# encoding: utf-8
import sys
import os
```

```
path = os.path.dirname(os.path.abspath(__file__))

import numpy as np
import acl
import base64
from PIL import Image, ImageDraw, ImageFont
from atlas_utils.acl_dvpp import Dvpp
import atlas_utils.constants as const
from atlas_utils.acl_model import Model
from atlas_utils.acl_image import AclImage
from atlas_utils.acl_resource import AclResource

SRC_PATH = os.path.realpath(__file__).rsplit("/", 1)[0]
MODEL_PATH = os.path.join(SRC_PATH, "../model/garbage_yuv.om")
MODEL_WIDTH = 224
MODEL_HEIGHT = 224
image_net_classes = [
   "Seashel", "Lighter","Old Mirror", "Broom","Ceramic Bowl",
"Toothbrush","Disposable Chopsticks","Dirty Cloth",
     "Newspaper", "Glassware", "Basketball", "Plastic Bottle", "Cardboard","Glass
Bottle", "Metalware", "Hats", "Cans", "Paper",
      "Vegetable Leaf","Orange Peel", "Eggshell","Banana Peel",
    "Battery", "Tablet capsules","Fluorescent lamp", "Paint bucket"]
```

```
def get_image_net_class(class_id):
    if class_id >= len(image_net_classes):
        return "unknown"
    else:
        return image_net_classes[class_id]

def pre_process(image, dvpp):
    """preprocess"""
    image_input = image.copy_to_dvpp()
    yuv_image = dvpp.jpegd(image_input)

    print("decode jpeg end")
    resized_image = dvpp.resize(yuv_image,
                    MODEL_WIDTH, MODEL_HEIGHT)

    print("resize yuv end")
    return resized_image

def post_process(infer_output, image_file):
    print("post process")
    data = infer_output[0]
    vals = data.flatten()
    top_k = vals.argsort()[-1:-6:-1]
    object_class = get_image_net_class(top_k[0])
    output_path = os.path.join(os.path.join(SRC_PATH, "../outputs"),
```

```
os.path.basename(image_file))
    origin_image = Image.open(image_file)
    draw = ImageDraw.Draw(origin_image)
    font = ImageFont.load_default()
    font.size =50
    draw.text((10, 50), object_class, font=font, fill=255)
    origin_image.save(output_path)
    object_class = get_image_net_class(top_k[0])
    return

def construct_image_info():
    """construct image info"""
    image_info = np.array([MODEL_WIDTH, MODEL_HEIGHT,
                           MODEL_WIDTH, MODEL_HEIGHT],
                           dtype = np.float32)
    return image_info

def main():
    if (len(sys.argv) != 2):
        # 这里的第一个参数是脚本名称，第二个参数是图片的目录
        print("The App arg is invalid")
        exit(1)

    acl_resource = AclResource()
    acl_resource.init()
```

```
    model = Model(MODEL_PATH)
    dvpp = Dvpp(acl_resource)

    image_dir = sys.argv[1]
    images_list = [os.path.join(image_dir, img)
                   for img in os.listdir(image_dir)
                   if os.path.splitext(img)[1] in const.IMG_EXT]

    #Create a directory to store the inference results
    if not os.path.isdir(os.path.join(SRC_PATH, "../outputs")):
        os.mkdir(os.path.join(SRC_PATH, "../outputs"))

    image_info = construct_image_info()
    for image_file in images_list:
        image = AclImage(image_file)
        resized_image = pre_process(image, dvpp)
        print("pre process end")

        result = model.execute([resized_image,])
        post_process(result, image_file)

        print("process "+image_file+" end")
if __name__ == '__main__':
    main()
```

将`atlas_utils`放入当前项目路径下。

```shell
cd ~/projects/garbage_classification/
# 解压atlas_utils.zip文件
# unzip atals_utils.zip
```

最后，项目的目录树如下所示。

```
parks@ubuntu:~/projects/garbage_classification$ tree . -L 2
.
├── atlas_utils
│   ├── acl_dvpp.py
│   ├── acl_image.py
│   ├── acl_logger.py
│   ├── acl_model.py
│   ├── acl_resource.py
│   ├── acl_venc.py
│   ├── camera.py
│   ├── constants.py
│   ├── data_buf.py
│   ├── __init__.py
│   ├── lib
│   ├── presenteragent
│   ├── README.md
│   ├── resource_list.py
│   ├── utils.py
│   └── video.py
├── data
│   ├── bottle.jpg
│   ├── dirtycloth.jpg
│   └── newspaper.jpg
├── model
│   ├── garbage_yuv.om
│   ├── insert_op_yuv.cfg
│   └── mobilenetv2.air
└── src
    └── classify_test.py

6 directories, 21 files
```

连接Atlas200DK开发板

```shell
ssh HwHiAiUser@192.168.1.2
mkdir -p ~/projects
```

```
# 在宿主机内(VmWare安装的虚拟机)执行，将项目文件传输到开发板中
scp -r $HOME/projects/garbage_classification
HwHiAiUser@192.168.1.2:/home/HwHiAiUser/projects

# 在开发板中进行推理
cd $HOME/projects/garbage_classification/
python3.6 src/classify_test.py ./data/

# 在宿主机中执行，用于获取推理结果
cd $HOME/projects/garbage_classification
scp -r
HwHiAiUser@192.168.1.2:/home/HwHiAiUser/projects/garbage_classification/outputs
./outputs
```

可在`$HOME/projects/garbage_classification/outputs/`下查看分类结果，示例如下图所示。

实验总结

本章实验介绍了 CANN 垃圾分类实现过程， 包括：

- 模型训练；
- 模型转换；
- 在 Atlas200DK 上进行部署推理。
  通过实验使学员熟悉昇腾应用开发流程， 加深对昇腾 CANN 相关理论的理解。

一些后续可拓展的地方

访问Atlas 200dk的github仓库，这里提供了一些示例代码，可以参考这些代码，完善一下该项目。比如：

- 调用摄像头拍照进行识别

gitee仓库地址是：https://gitee.com/ascend。