基于MindX SDK的语音识别实验指导手册

版本：1.0



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2021。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  C:\Users\jwx341670\Desktop\华为标志 Huawei Logo 2018\竖版标志Vertical Version\PNG\HW_POS_RBG_Vertical-150ppi.png 和其他华为商标均为华为技术有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 华为技术有限公司 | |
| 地址： | 深圳市龙岗区坂田华为总部办公楼 邮编：518129 |
| 网址： | http://[e](http://e.huawei.com/).huawei.com |

目录

[1 实验总览 2](#_Toc88809355)

[1.1 实验背景 2](#_Toc88809356)

[1.2 实验目的 2](#_Toc88809357)

[1.3 实验清单 2](#_Toc88809358)

[2 基于MindX SDK的语言识别实验 3](#_Toc88809359)

[2.1 实验简介 3](#_Toc88809360)

[2.2实验环境 3](#_Toc88809361)

[2.3实验目的 3](#_Toc88809362)

[2.4实验步骤 4](#_Toc88809363)

[2.4.1 创建实验环境 4](#_Toc88809367)

[2.4.2 实验过程 12](#_Toc88809368)

[2.4.3 实验内容解析 15](#_Toc88809369)

[2.5实验总结 20](#_Toc88809370)

# 实验总览

## 实验背景

语音识别技术，也被称为自动语音识别Automatic Speech Recognition，(ASR)，其目标是将人类的语音中的词汇内容转换为计算机可读的输入，例如按键、二进制编码或者字符序列。

## 实验目的

本章实验的主要目的是掌握语音识别技术（ASR）相关基础知识点，使用MindX SDK工具包实现语言识别的功能，加深对相关理论的理解。

## 实验清单

表格：实验、简述、难度、软件环境、硬件环境。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| 基于MindX SDK的语音识别实验 | 通过MindX SDK来完成语音识别任务 | 高级 | Python3.7.5、CANN3.3.0 | ECS |

# 基于MindX SDK的语音识别实验

## 实验简介

语音识别技术，也被称为自动语音识别Automatic Speech Recognition，(ASR)，其目标是将人类的语音中的词汇内容转换为计算机可读的输入，例如按键、二进制编码或者字符序列。

本实验通过配置镜像环境、购买ECS服务、部署项目代码、修改配置文件及执行推理脚本全流程展示如何使用MindX SDK进行中文语音录音文件的识别。

## 2.2实验环境

对象存储服务（OBS）、弹性云服务器（ECS）、MobaXterm。

对象存储服务（Object Storage Service，OBS）提供海量、安全、高可靠、低成本的数据存储能力，可供用户存储任意类型和大小的数据。适合企业备份/归档、视频点播、视频监控等多种数据存储场景。

弹性云服务器（Elastic Cloud Server,ECS） 是由CPU、内存、操作系统、云硬盘组成的基础的计算组件。弹性云服务器创建成功后，您可以像使用自己的本地PC或服务器一样，在云上使用弹性云服务器。

MobaXterm (<https://mobaxterm.mobatek.net/>)是远程SSH连接工具。

MindX SDK提供昇腾AI处理器加速的各类AI软件开发套件(SDK)，提供极简易用的API，加速高性能AI应用的开发，赋能千行百业。详细资料可参考（https://support.huaweicloud.com/mindxsdk/index.html）。

## 2.3实验目的

了解如何使用对象存储服务（OBS）；了解如何创建华为云私有镜像并通过私有镜像搭建弹性云服务器（ECS）；了解如何在ECS中部署基于MindX SDK的语言识别项目并运行。

## 2.4实验步骤



### 创建实验环境

创建个人密钥

华为云平台的个人密钥，即Access Key和Secret Access Key。主要用于登录华为云平台的应用时候做安全验证。在后续步骤的登录OBS桶的时候会用到，创建方式如下:

首先登录华为云平台，然后进入到控制台，选择区域为“北京四”。



登录华为云并选择区域

在控制台选择个人账户，然后依次选择“我的凭证”—“访问密钥”—“新建访问密钥”来创建个人密钥。（注：访问密钥在创建完成之后会弹出下载按钮，密钥文件只能下载一次，下载后请妥善保存。）



创建并下载访问密钥

登录OBS并创建文件夹

使用OBS Brower+登录OBS（注：OBS Browser+是一款用于访问和管理对象存储服务OBS的图形化工具，支持完善的桶管理和对象管理操作。OBS Browser+的图形化界面可以非常方便地让用户在本地对OBS进行管理。下载地址为：https://developer.huaweicloud.com/tools#section-1。可以根据个人电脑所搭载的系统环境不同，下载对应版本的安装包，安装即可使用。）



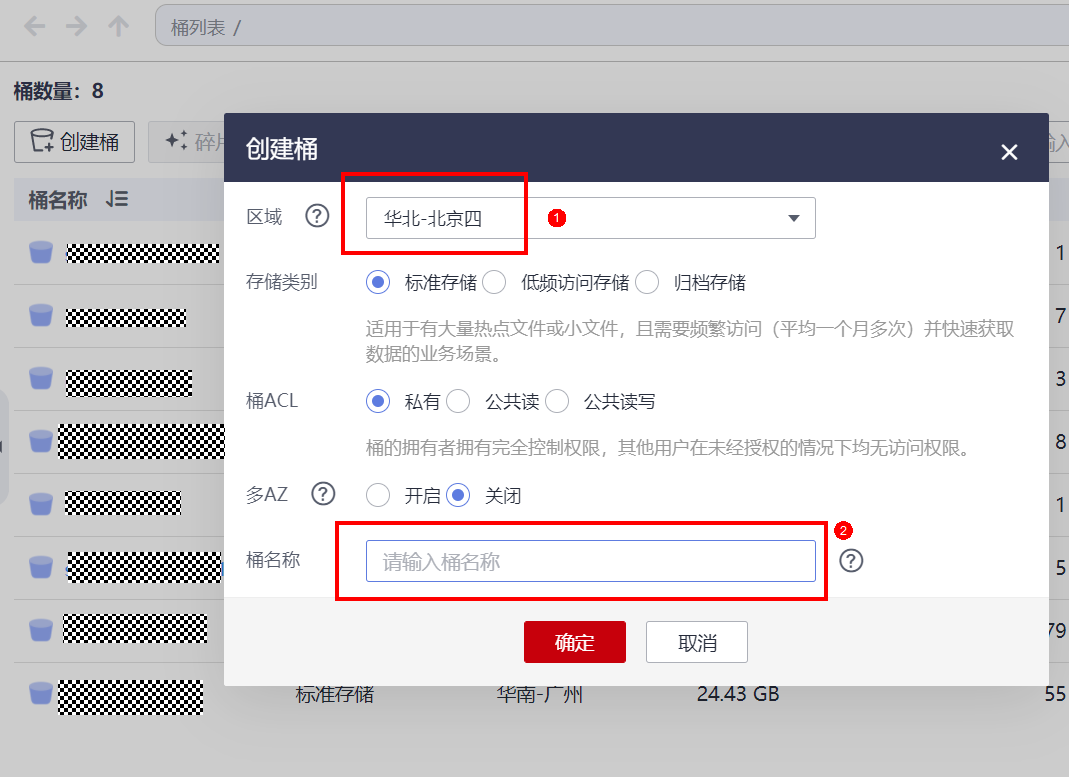
OBS Browser下载

推荐使用“AK方式登录”，Access Key和Secret Access Key为上一步创建并下载的密钥文件中获取。



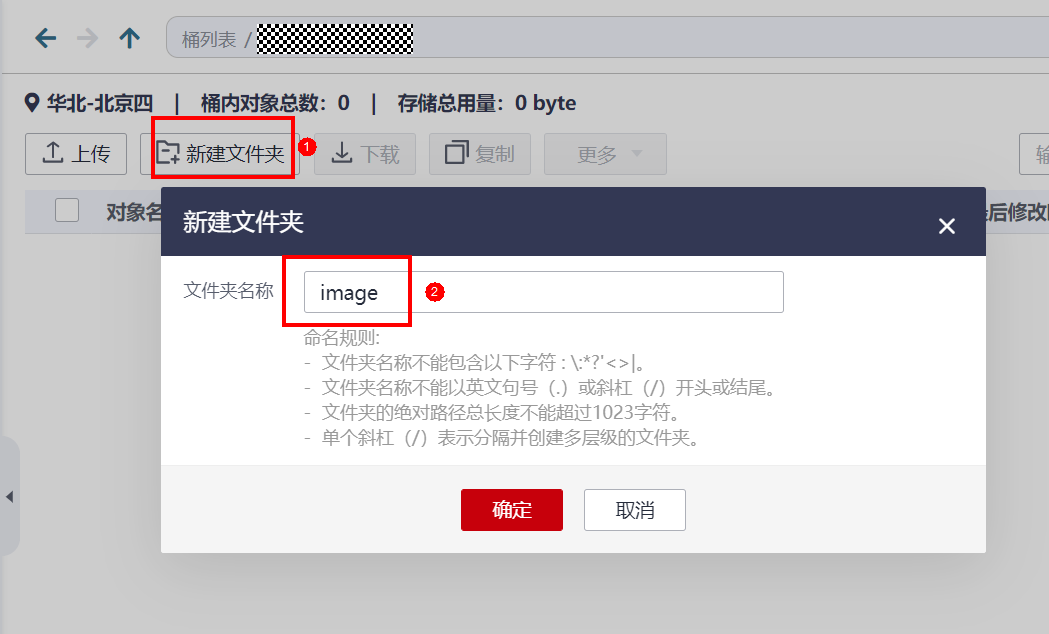
OBS Brower+登录

进入之后创建一个obs桶，作为存放镜像文件的路径，创建时选择区域为“北京四”。然后在这个桶路径下，我们创建一个 “image”目录用于本实验数据的存放：



创建OBS桶

继续在桶中创建一个文件夹“image”

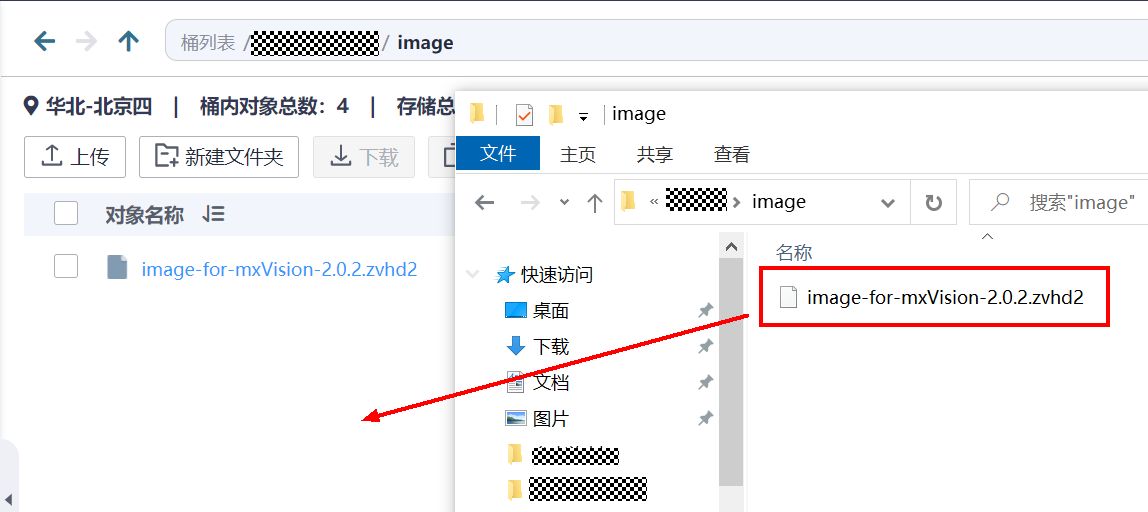


创建image文件夹

上传镜像文件

本次实验用到的镜像文件下载地址为：<https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/ASR/image-for-mxVision-2.0.2.zvhd2>。该镜像用于ECS服务器的创建，里面会包含我们实验用到的MindX SDK环境。

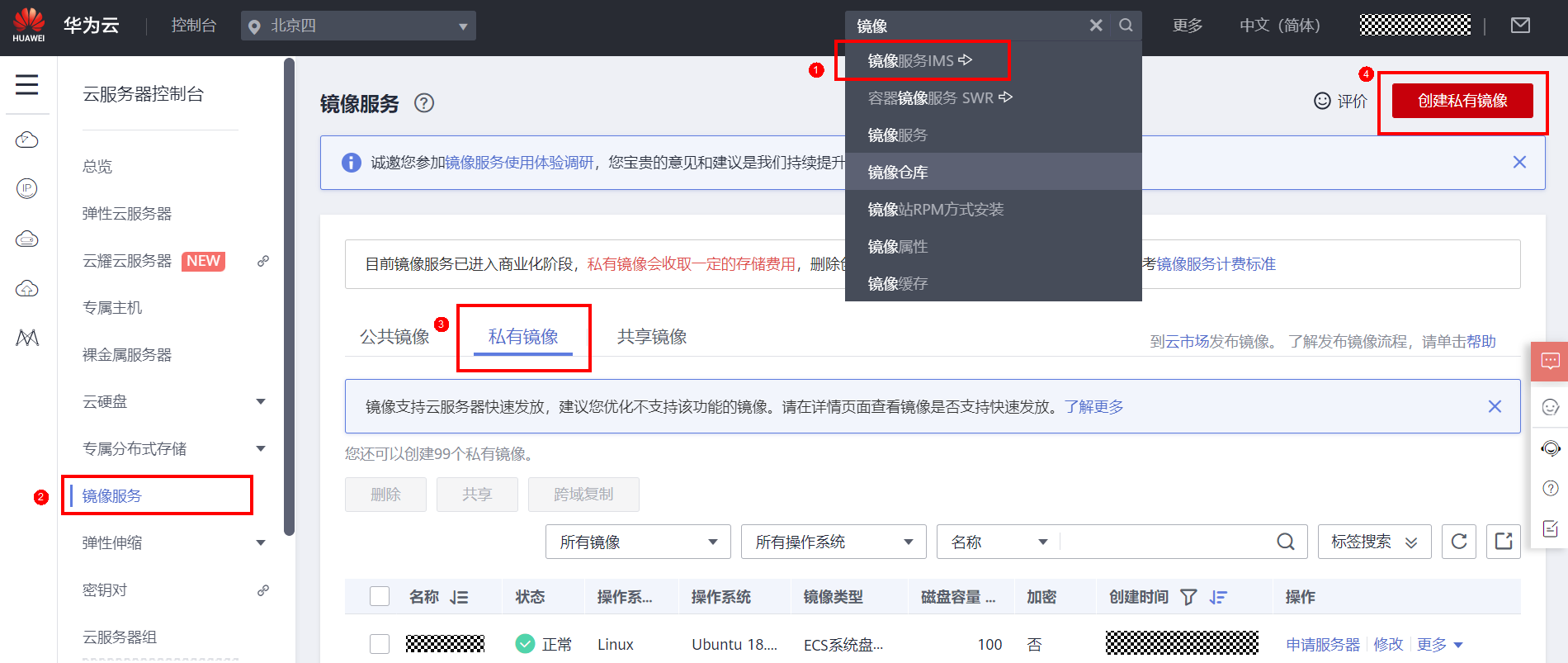
我们需要先将镜像文件下载到本地电脑中，然后从OBS进入到上一步建好的image文件夹，直接从本地文件夹把镜像文件拖到OBS中，即可开始上传。



上传镜像文件

创建私有镜像

在华为云任意页面上方的搜索栏中搜索“镜像”并点击“镜像服务IMS”进入镜像服务页面，注意选择左上角的区域为“北京四”，依次点击“镜像服务”—“私有镜像”—“创建私有镜像”，开始创建私有镜像服务



创建私有镜像

按照如下配置来创建私有镜像：

第一部分 镜像位置和来源：

* 区域：华北-北京四
* 创建方式：系统盘镜像
* 选择镜像源：镜像文件
* 在下方的桶列表中选中步骤三中上传的镜像文件
* 快速通道 勾选
* 镜像文件准备 勾选



镜像位置和来源配置

第二部分 配置信息：

镜像用途：ECS系统盘镜像

架构类型：X86

操作系统：Ubuntu 18.04 server 64bit

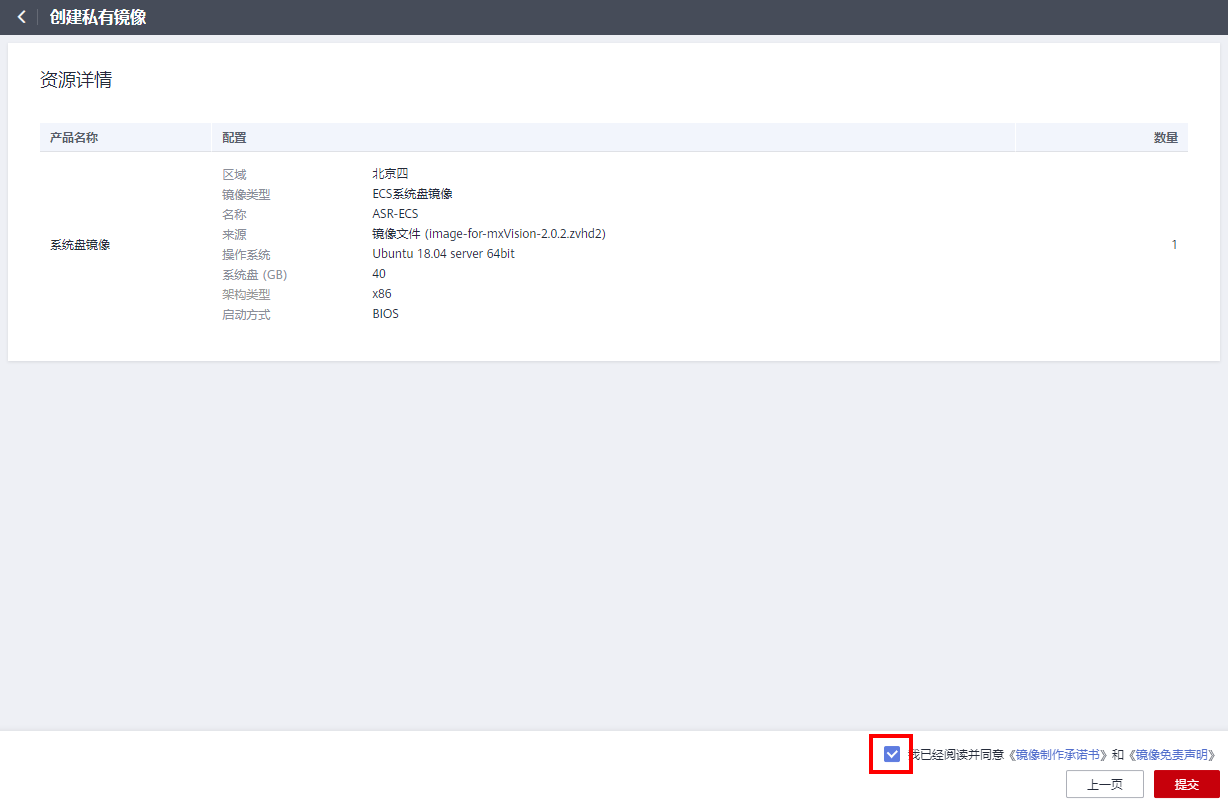
系统盘（GB）：40

名称：ASR-ECS（此处可自定义）



配置信息

选择完成的配置如下图所示，勾选“同意协议”，然后点“提交”开始创建



提交镜像创建任务

创建ECS

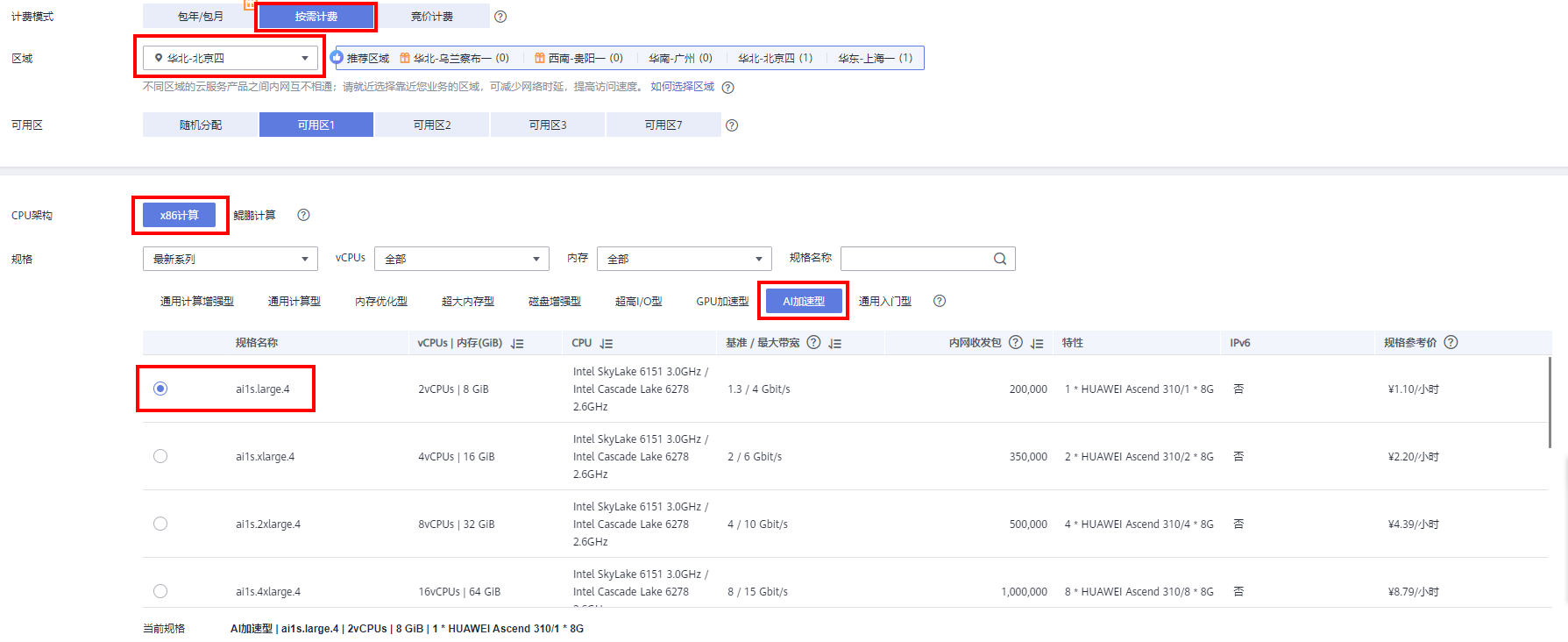
如下图所示，待镜像状态为“正常”以后，点击右边的“申请服务器”按钮开始创建ECS 。



开始创建ECS

根据以下参数选择服务器配置：

* 计费模式：按需计费
* 区域：华北-北京四
* CPU架构：x86计算
* 产品类型：AI加速型
* 规格名称：ai1s.large.4



选择ECS配置

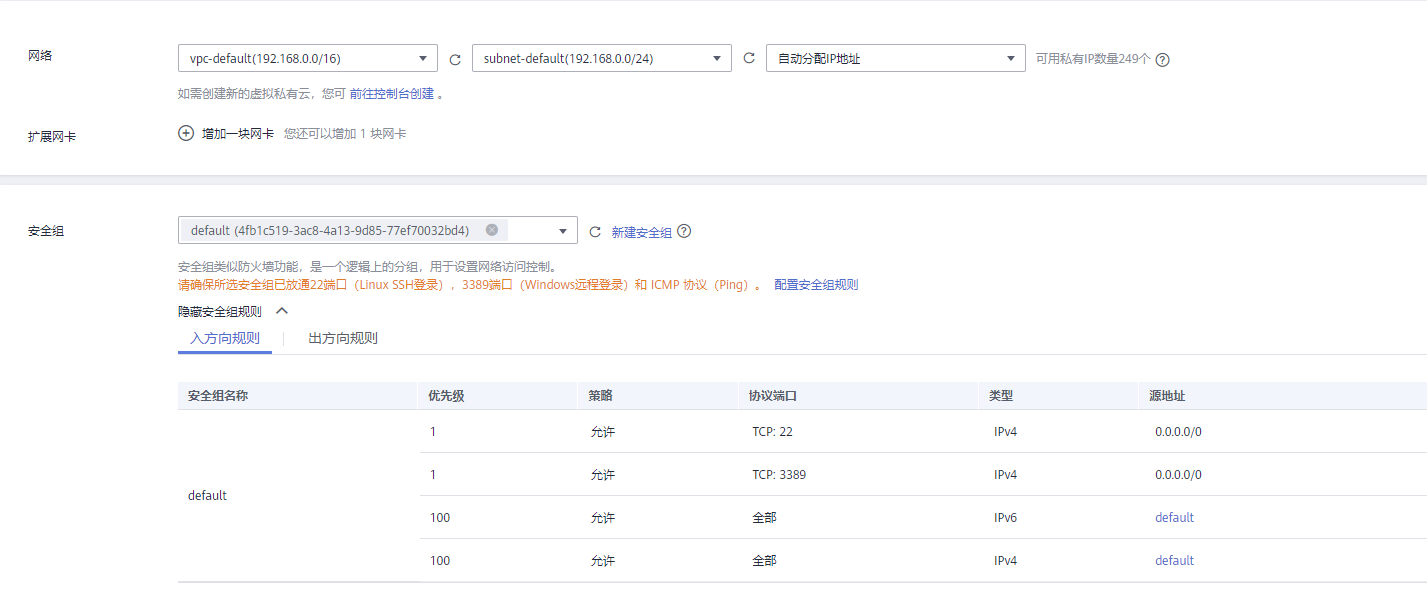
* 镜像：选择刚才创建好的私有镜像（默认选项）
* 系统盘：100GB（默认选项）

然后点击“下一步：网络配置”



选择镜像文件和系统盘

* 网络：vpc-default
* 安全组：default



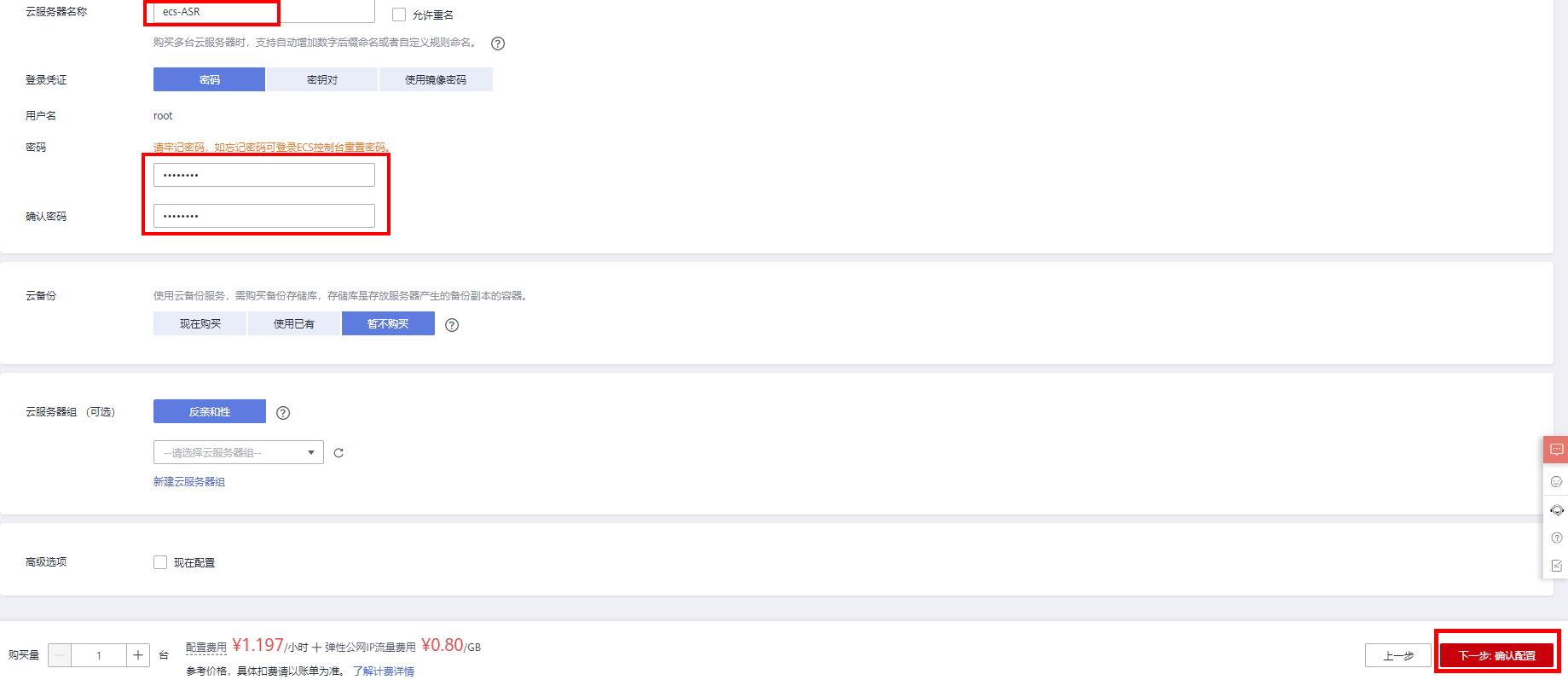
配置网络和安全组

* 弹性公网IP：现在购买
* 线路：全动态BGP（默认选项）
* 公网宽带：按流量计费
* 带宽大小：10

然后点击“下一步：高级配置”

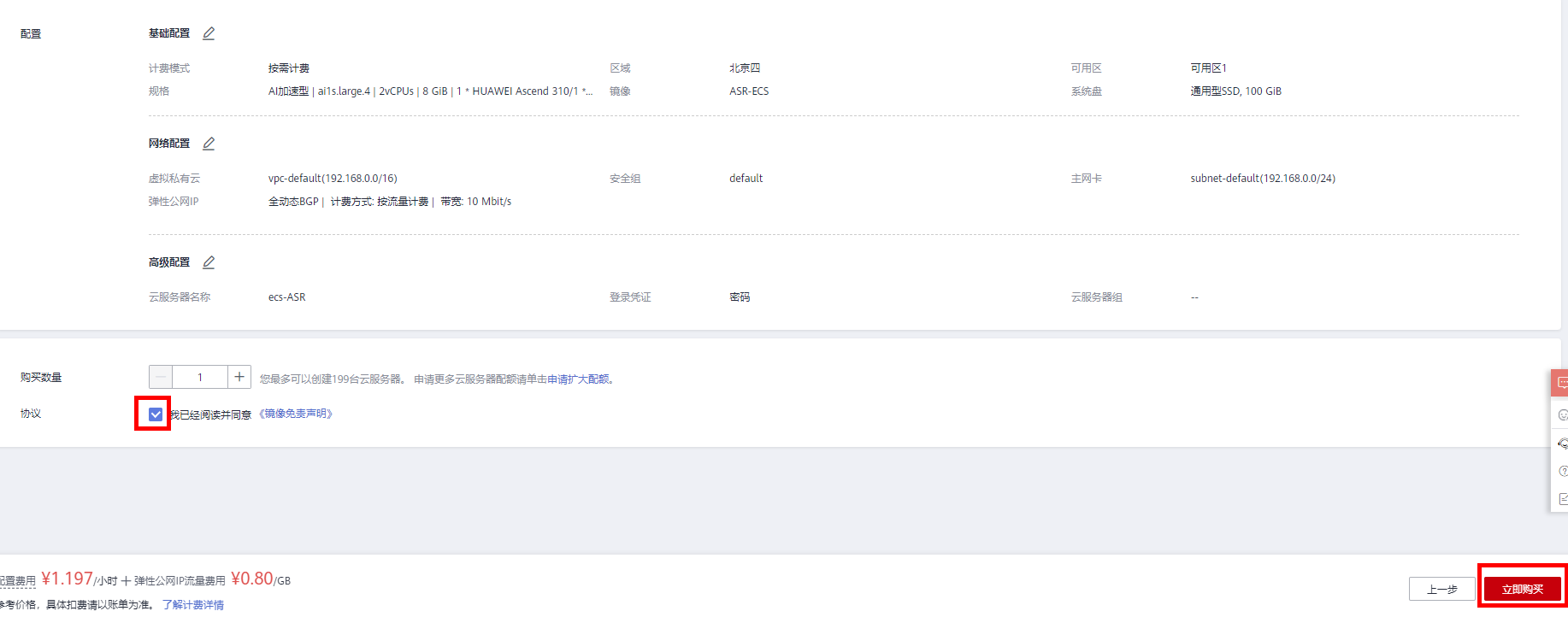


如下图所示按照个人习惯设置“云服务器名称”、“密码”、“确认密码”，然后点击“下一步：确认配置”



设置ECS名称和密码

确认配置信息无误以后，勾选下方的“同意协议”，然后点击“立即购买”开始创建ECS。



确认配置并购买ECS

当服务器状态为“运行中”时，即可通过SSH工具连接弹性公网IP登录ECS。

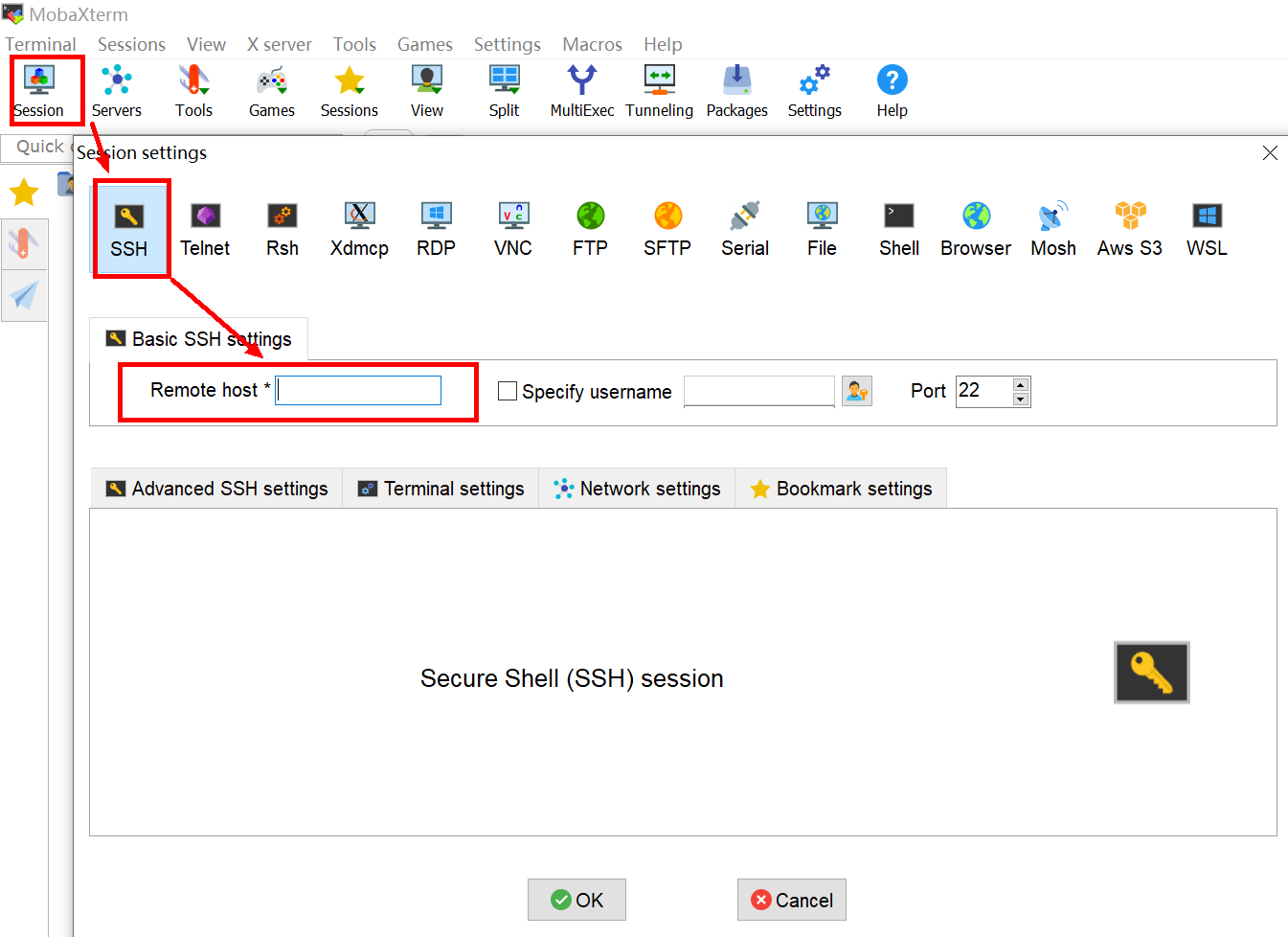


ECS创建成功

### 实验过程

使用MobaXterm登录ECS

使用MobaXterm软件，通过SSH方式登录到ECS，依次输入弹性公网IP、用户名root，密码为创建ECS时设定的密码。初次登录界面如下所示。



输入弹性公网IP登录华为云服务器



ECS登录成功界面

下载并部署项目

根据实验流程，需要安装zip、tree命令，输入：

apt install zip

apt install tree

安装实验所需要的依赖环境（安装mxVision、升级protobuf），输入以下命令：

bash /home/HwHiAiUser/MindX/Ascend-mindxsdk-mxvision\_2.0.2\_linux-x86\_64.run --install

pip3 install --upgrade protobuf

source ./bashrc

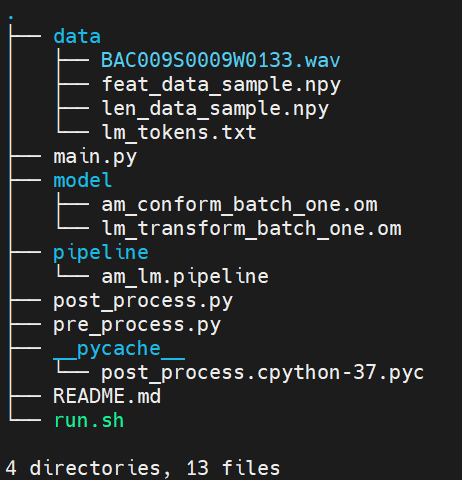
使用wget命令下载项目代码到ECS中（项目代码链接为：https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/ASR/AutoSpeechRecognition.zip），使用zip命令进行解压，然后进入到项目文件夹中，输入如下命令：

wget https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com/ASR/AutoSpeechRecognition.zip

unzip AutoSpeechRecognition.zip

cd AutoSpeechRecognition

tree



项目代码树状目录

从上面的项目文件中我们可以完整看到整个项目中文件信息：

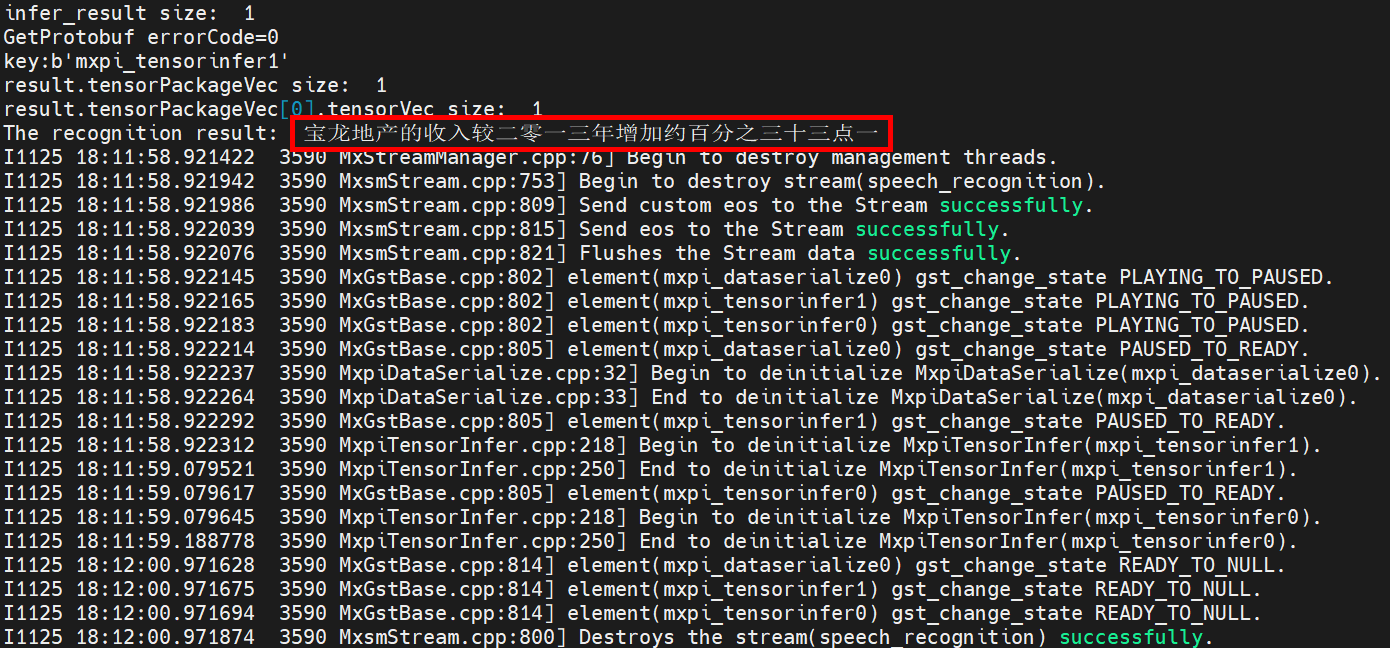
* data文件夹：待推理的语言文件、转换后特征数据和语言模型的token文件
* model文件夹：推理模型
* pipline文件夹：调用mindX SDK推理所需的配置文件
* main.py文件：推理任务执行业务代码
* pre\_process.py和post\_process.py文件：数据预处理和后处理文件
* run.sh：整个项目运行的脚本文件。

执行推理任务

接下来，我们通过运行run.sh脚本完成语言文件的识别，输入：

./run.sh

从下图的推理结果中可以，我们通过调用MindX SDK接口完成了语音推理任务，输出了wav语音内容的中文含义。



语音识别结果

### 实验内容解析

#### 实验流程

MindX SDK实现了端到端的自动语音识别（Automatic speech recognition, ASR）。

ASR主要分为两个步骤：

* 将语音转换成对应的拼音
* 将拼音转换成对应的文字

对于第一步将语音转换为对应的拼音的声学模型用的模型是Google在2020年提出的Conformer模型：Convolution-augmented Transformer for Speech Recognition；

对于第二步语言模型采用的是transformer模型。

#### 模型转换

代码中我们使用的om模型，这种文件是通过atc+pb模型转换而来的。通过ACT工具（注：我们可以把Tensorflow、Caffe框架训练完成的模型转换为Ascend芯片支持的om模型，从而可以在Ascend环境下使用。更多信息可参考：https://support.huaweicloud.com/atc-model-convert-Atlas200DK202/atlasatc\_16\_0003.html）。

#### SHELL脚本

实验过程中执行的run.sh中代码如下，需要根据实际情况设置MX\_SDK\_HOME以及LD\_LIBRARY\_PATH中ascend-toolkit的动态库路径。

set -e

# Simple log helper functions

info() { echo -e "\033[1;34m[INFO ][MxStream] $1\033[1;37m" ; }

warn() { echo >&2 -e "\033[1;31m[WARN ][MxStream] $1\033[1;37m" ; }

export LD\_LIBRARY\_PATH=${MX\_SDK\_HOME}/lib:${MX\_SDK\_HOME}/opensource/lib:${MX\_SDK\_HOME}/opensource/lib64:/usr/local/Ascend/ascend-toolkit/latest/acllib/lib64:${LD\_LIBRARY\_PATH}

export GST\_PLUGIN\_SCANNER=${MX\_SDK\_HOME}/opensource/libexec/gstreamer-1.0/gst-plugin-scanner

export GST\_PLUGIN\_PATH=${MX\_SDK\_HOME}/opensource/lib/gstreamer-1.0:${MX\_SDK\_HOME}/lib/plugins

#to set PYTHONPATH, import the StreamManagerApi.py

export PYTHONPATH=$PYTHONPATH:${MX\_SDK\_HOME}/python

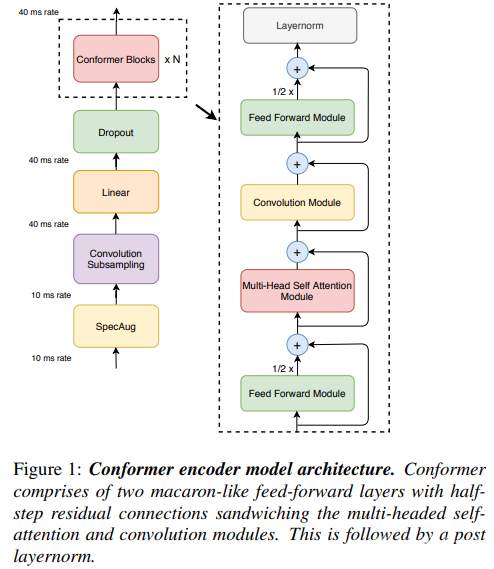
python3.7 main.py

exit 0

#### 模型及Pipline介绍

本实验中代码的核心部分为Conformer结构。Conformer结合了cnn获取局部信息能力，以及self-attention获取文本整体内容的能力，是一个end-to-end的语音理解模型。

从下图中可以看到，Conformer Block结构前后为FFN模块，中间夹着MHSA和Convolution Block，类似一个三明治模型。同时还有残差连接。



Conformer模型结构

本实验有两步推理，wav数据经过预处理后得到张量数据，数据进入声学模型得到文本特征，再进入语言模型得到结果文本。因此，可以使用mxpi\_tensorinfer插件进行推理。

Pipline文件的代码及注释如下所示（使用时需要删除多余的注释和空格）

{

"speech\_recognition": {

"stream\_config": {

"deviceId": "3" # 设置npu设备号

},

"appsrc0": {

"props": {

"blocksize": "320324"

},

"factory": "appsrc", # 数据入口，从main.py中传送预处理后的数据

"next": "mxpi\_tensorinfer0"

},

"mxpi\_tensorinfer0": {

"props": {

"dataSource": "appsrc0",

"modelPath": "model/am\_conform\_batch\_one.om" # 推理得到文本特征

},

"factory": "mxpi\_tensorinfer",

"next": "mxpi\_tensorinfer1"

},

"mxpi\_tensorinfer1": {

"props": {

"dataSource": "mxpi\_tensorinfer0",

"modelPath": "model/lm\_transform\_batch\_one.om" # 得到最终结果

},

"factory": "mxpi\_tensorinfer",

"next": "mxpi\_dataserialize0"

},

"mxpi\_dataserialize0": {

"props": {

"outputDataKeys": "mxpi\_tensorinfer1"

},

"factory": "mxpi\_dataserialize",

"next": "appsink0"

},

"appsink0": {

"factory": "appsink"

}

}

}

#### main脚本解析

main.py脚本定义的推理任务的核心处理流程，分为以下3个步骤：

* 初始化流管理器，加载pipeline。
* 由于模型直接输入张量数据，因此使用sendprotobuf与getprotobuf接口进行数据发送与接收。
* 接受模型输出张量，进行后处理得到文本。

脚本中关键代码及注释如下:

张量数据结构定义

# 创建MxpiTensorPackageList，由于传入的是张量数据，而不是图片数据，因此可以用此接口

mxpi\_tensor\_package\_list = MxpiDataType.MxpiTensorPackageList()

tensor\_package\_vec = mxpi\_tensor\_package\_list.tensorPackageVec.add() # 固定写法，插入tensorPackageVec

# 固定写法，往packageVec中插入tensorVec，tensorVec为内置结构体

# 详情见（MxpiTensor）：https://support.huawei.com/enterprise/zh/doc/EDOC1100207070/d71c9c39

tensorVec = tensor\_package\_vec.tensorVec.add()

tensorVec.memType = 1

tensorVec.deviceId = 0

tensorVec.tensorDataSize = int(feat\_data.shape[1]\*feat\_data.shape[2]\*4)

tensorVec.tensorDataType = 0 # float32

for i in feat\_data.shape:

tensorVec.tensorShape.append(i)

tensorVec.dataStr = feat\_data.tobytes()

# 插入第二个输入。此模型需要2个输入。同样是往tensor package vec中插入tensor vec

tensorVec2 = tensor\_package\_vec.tensorVec.add()

tensorVec2.memType = 1

tensorVec2.deviceId = 0

tensorVec2.tensorDataSize = int(4) # bytes of length data.

tensorVec2.tensorDataType = 3 # int32

for i in len\_data.shape:

tensorVec2.tensorShape.append(i)

tensorVec2.dataStr = len\_data.tobytes()

protobuf结构定义：

protobuf\_vec = InProtobufVector() # 创建protobufVec，保存所有protobuf

# 创建每一个protobuf，并插入protobufVec中

protobuf = MxProtobufIn()

protobuf.key = b'appsrc0' # pipeline中通过appsrc0接受数据，因此这里写appsrc0在，注意需要二进制类型字符串

# 数据类型为tensorPackageList，这里需要与前面定义的对应起来

protobuf.type = b'MxTools.MxpiTensorPackageList'

protobuf.protobuf = mxpi\_tensor\_package\_list.SerializeToString() # 固定写法

protobuf\_vec.push\_back(protobuf)

发送数据

unique\_id = stream\_manager.SendProtobuf(stream\_name, in\_plugin\_id, protobuf\_vec)

接收推理结果

key\_vec = StringVector()

key\_vec.push\_back(b'mxpi\_tensorinfer1') # 需要第二个模型输出结果，因此写mxpi\_tensorinfer1

infer\_result = stream\_manager.GetProtobuf(stream\_name, in\_plugin\_id, key\_vec) # getProtobuf接口获取protobuf结构体

解析protobuf

# 错误检验代码略

# 模型输出个数对应infer\_result长度，（这里为1个输出，yolov3则为3个）

print("key:" + str(infer\_result[0].messageName))

# 发送的是MxpiTensorPackageList类型，接受的也是同样类型

result = MxpiDataType.MxpiTensorPackageList()

result.ParseFromString(infer\_result[0].messageBuf) # 固定写法，解析获得messageBuf

# 由于定义输入的流程为MxpiTensorPackageList->tensorPackageVec->tensorVec(dataStr保存数据）

# 因此解析如下。ids即为模型输出的张量结果

ids = np.frombuffer(result.tensorPackageVec[0].tensorVec[0].dataStr, dtype=np.int32)

## 2.5实验总结

本章实验在华为云ECS中上，调用MindX SDK中已实现的Conformer模型来完成了语音数据的识别任务，从中可以了解到如何使用镜像文件搭建ECS服务器；了解Conformer模型的结果及其在语音识别任务中的应用；了解如何通过MindX SDK通过简单配置快速完成一个语言识别接口的开发。