



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

GPU 计算实验报告

实验三 基于昇腾 CANN 的目标检测应用（AscendCL 接口）

学院：计算机科学与技术

姓名：

学号：

一、实验预习（10 分）

1、注册华为云账号：<https://www.huaweicloud.com/>

2、课程内容预习：

<https://education.huaweicloud.com/courses/course-v1:HuaweiX+CBUCNXA024+Self-paced/about>

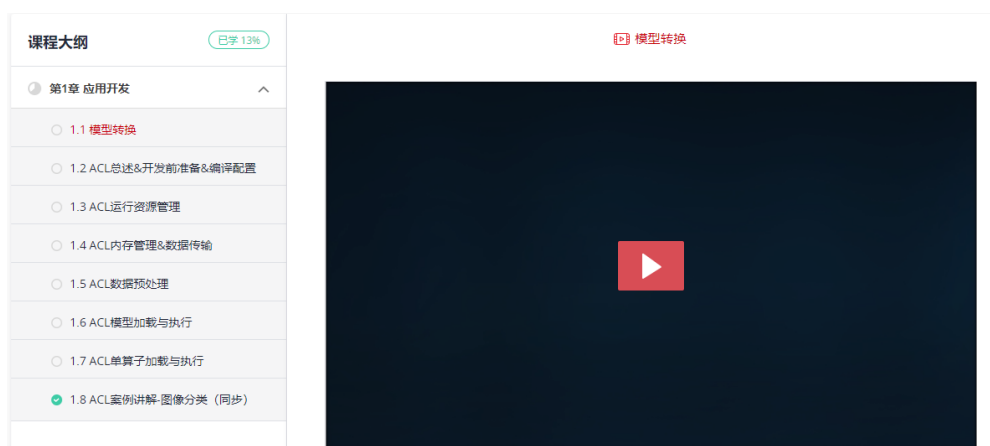


图 1 昇腾学院发布的 ASCENDCL 课程

https://www.bilibili.com/video/BV16K4y1976L?spm_id_from=333.999.0.0

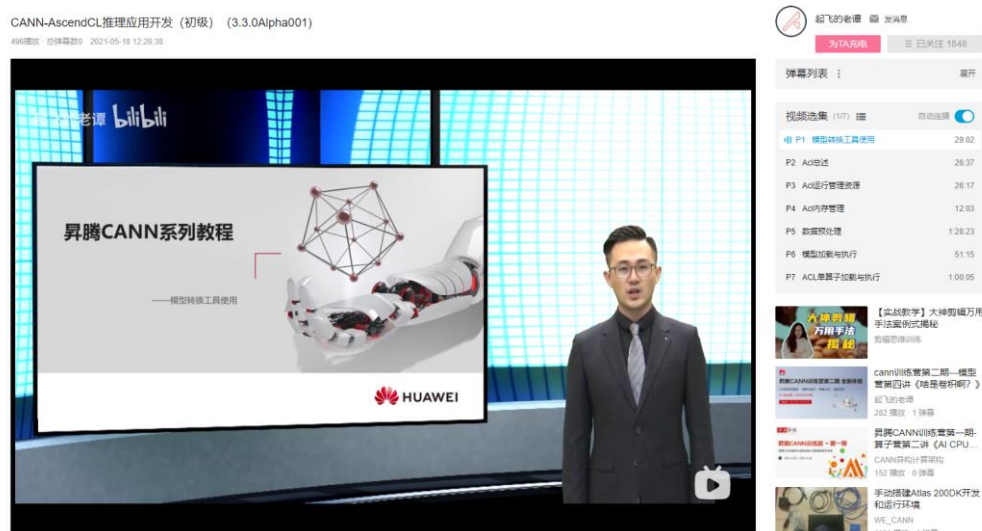


图 2 华为谭老师 bilibili 发布的视频

3、问题（10 分）：

（回答请使用红色文字）

1）在昇腾 310 上执行推理，模型格式必须是 om，需要使用 ATC 工具进行模型转换，目前支持哪些深度学习框架的模型转换？

A、Caffe B、TensorFlow C、MindSpore D、PyTorch

答：AB

2）模型转换时设置静态 AIPP 参数，模型生成后，AIPP 参数被保存在 om 模型中，

其功能主要有什么？

A、图像裁切对齐 B、色域转换 C、归一化 D、图片解码与缩放

答：B

3) 下面那些选项是 AscendCL 能够完成的？

A、运行资源管理（Device 管理、Context 管理、Stream 管理）

B、内存管理

C、数据预处理 DVPP

D、模型加载与执行

E、算子加载与执行

答：ABCDE

4) DVPP 用于数据预处理，能够对图像解码、缩放，输出 YUV420sp 格式的图片，解码对输出图片的宽高有对齐要求，宽要求 128 对齐，高 16 对齐，缩放也对宽高有对齐要求，具体为：

答：

宽：（16）对齐

高：（2）对齐

5) ASCENDCL 中推理所需的输入输出数据，是通过一种特定的数据结构来组织的，称为 Dataset，所有的输入组成了一个 Dataset，所有的输出组成了一个 Dataset，但对于很多模型来说，输入不止一个，那么所有的输入集合叫 Dataset，其中的每一个输入叫什么？

答：dataItem

二、实验目标

- 1、了解华为昇腾全栈开发工具 MindStudio 及其离线模型转换功能；
- 2、了解如何使用 ASCENDCL 开发基于华为昇腾处理器的神经网络推理应用。

三、实验内容

完成链接中的实验：

[https://lab.huaweicloud.com/testdetail.html?testId=458&ticket=ST-947652-](https://lab.huaweicloud.com/testdetail.html?testId=458&ticket=ST-947652-UdMQSjr2EsqynL7E0Qu4Dkqz-ss0)

[UdMQSjr2EsqynL7E0Qu4Dkqz-ss0](https://lab.huaweicloud.com/testdetail.html?testId=458&ticket=ST-947652-UdMQSjr2EsqynL7E0Qu4Dkqz-ss0)

基于昇腾AI处理器的目标检测应用（ACL）

本实验通过模型转换、数据预处理/网络模型加载/推理/结果输出全流程展示昇腾处理器推理应用开发过程，帮助您快速熟悉ACL这套计算加速库。

免费 40实验点 今日剩余免费名额：5个

提示：实验资源会在总时长用完或中途退出后释放，请预留足够时间并尽快操作。

开始实验

难度程度

中级

实验评分

★★★★★

实验时长

120 分钟

实验人次

2588 人次

实验简介

实验评价 (242)

实验手册

实验目标与基本要求

① 了解华为昇腾全栈开发工具MindStudio及其离线模型转换功能；
② 了解如何使用ACL开发基于华为昇腾处理器的神经网络推理应用。

实验摘要

1. 准备环境
2. 创建弹性云服务器ECS
3. 配置工程
4. 编写代码
5. 编译运行
6. 运行Profiling 查看推理性能

相关实验

- 使用昇腾AI弹性云服务器实现目标检...
- 使用昇腾AI弹性云服务器实现图像分...
- 使用昇腾AI弹性云服务器实现图像分...

实验所用产品

- 弹性云服务器ECS
- 虚拟私有云VPC

图 3 基于昇腾 AI 处理器的目标检测应用（ASCENDCL）

参考文档：

https://support.huaweicloud.com/aclcppdevg-cann504alpha1infer/aclcppdevg_01_0001.html

全部文档

昇腾CANN社区版(5.0.4.alpha001)(推理)

搜索本产品帮助文档

应用开发 (C++)

前言

使用约束

文档学习路径

应用开发向导

准备开发和运行环境

快速入门

获取更多样例

基础知识

接口调用流程

初级功能

高级功能

文档首页 > 昇腾CANN社区版(5.0.4.alpha001)(推理) > 应用开发 (C++) > 前言

前言

更新时间：2021/11/16 GMT+08:00

查看PDF

分享

本文导读

概述

读者对象

概述

本文用于指导开发人员基于现有模型，使用AscendCL（Ascend Computing Language）提供的C语言API库开发深度神经网络应用，用于实现目标识别、图像分类等功能。

读者对象

本文档适用于基于AscendCL接口进行应用开发的人员，通过本文档您可以达成：

- 了解AscendCL的功能架构、基本概念以及接口的典型调用流程。
- 使用AscendCL接口进行应用开发的基本流程和实现方法。
- 能够基于本文档中的样例，扩展进行其它应用的开发。

掌握以下经验和技能可以更好地理解本文档：

- 具备C++/C语言程序开发能力。
- 对机器学习、深度学习有一定的了解。

下一篇：使用约束 >

图 4 AscendCL 官方文档

四、实验过程（80 分）

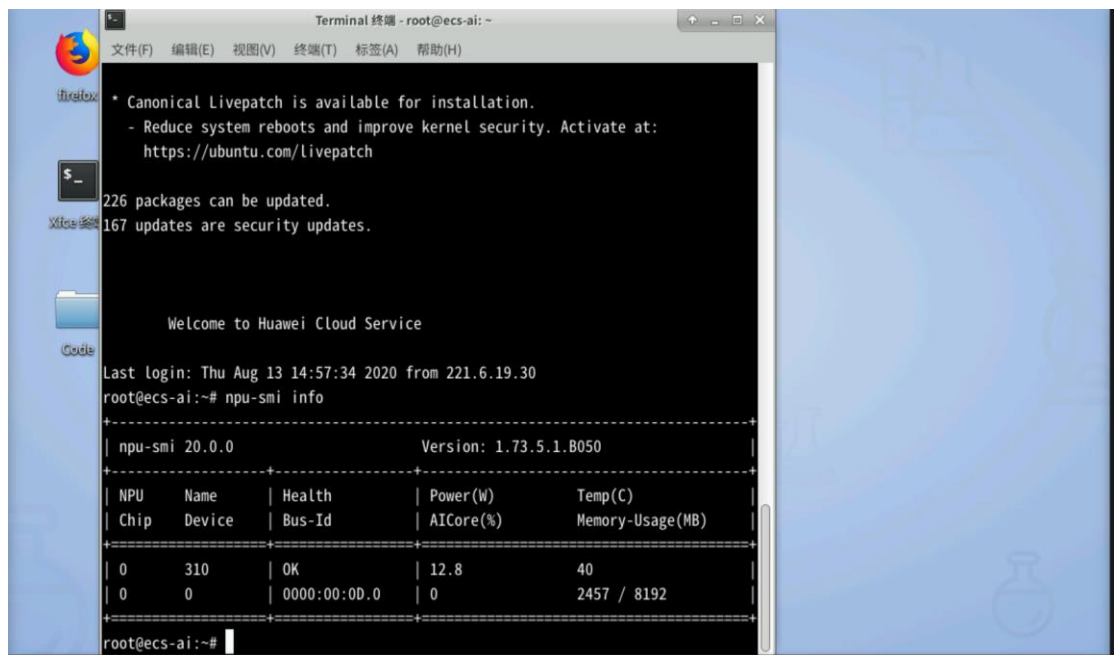
1. 环境准备（10 分）

完成弹性云服务器 ECS 创建，获取公网 IP，如 XX.XX.XX.XX，然后通过终端工具登录云服务器（密码为云服务器创建时自己设置的密码）：

```
ssh root@XX.XX.XX.XX
```

并且输入下列命令获取云服务器上昇腾芯片的信息并截图：

```
npu-smi info
```



```
Terminal 终端 - root@ecs-ai: ~
文件(F) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)

* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
https://ubuntu.com/livepatch

226 packages can be updated.
167 updates are security updates.

Welcome to Huawei Cloud Service

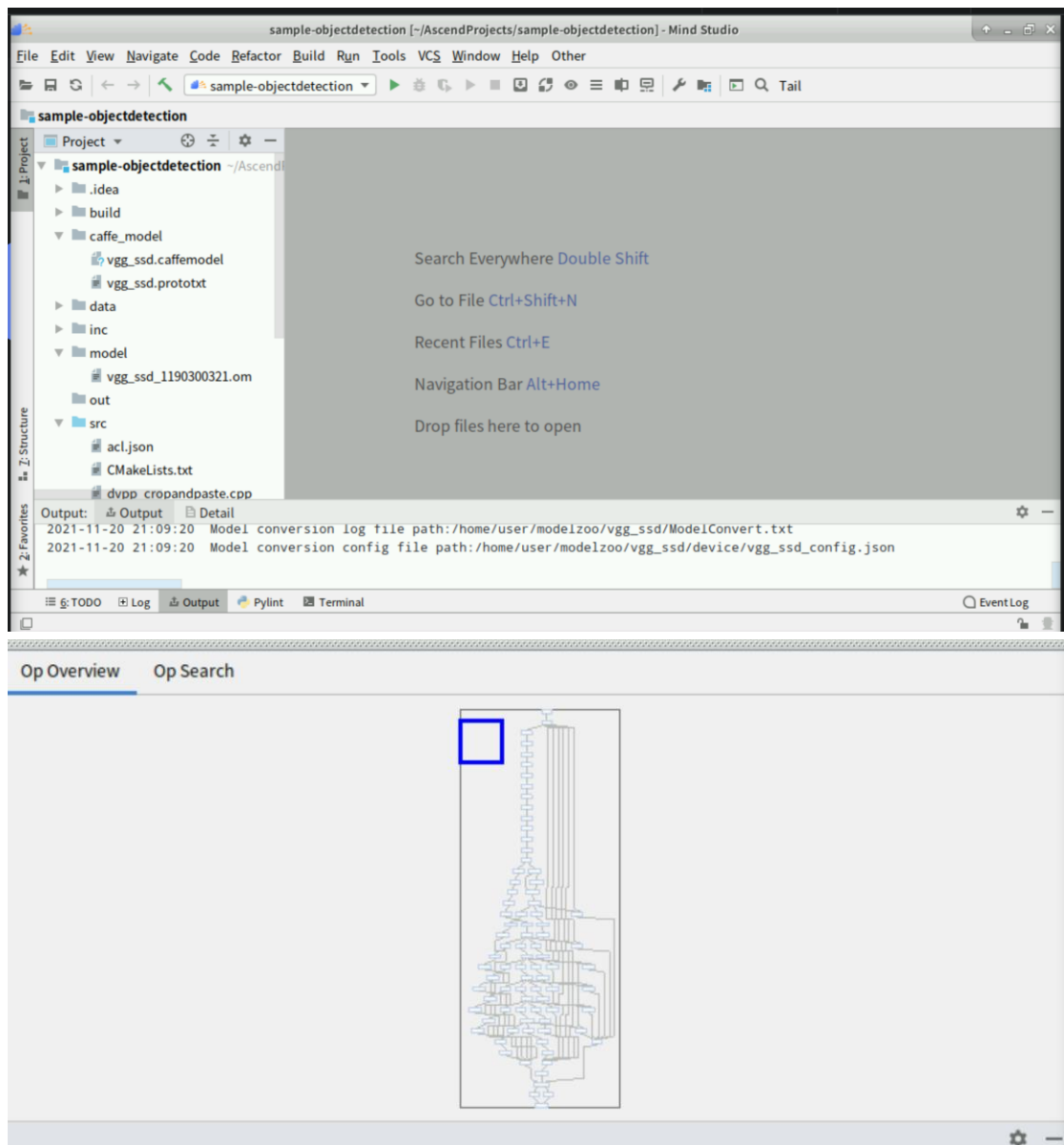
Last login: Thu Aug 13 14:57:34 2020 from 221.6.19.30
root@ecs-ai:~# npu-smi info
+-----+
| npu-smi 20.0.0 | Version: 1.73.5.1.B050 |
+-----+
| NPU   Name   | Health   | Power(W) | Temp(C) |
| Chip  Device | Bus-Id   | AICore(%) | Memory-Usage(MB) |
+-----+
| 0     310    | OK       | 12.8     | 40       |
| 0     0      | 0000:00:0D.0 | 0        | 2457 / 8192 |
+-----+
```

2. 模型转换（30 分）

对于已经转换成功的.om 模型文件，可以在 MindStudio 界面呈现其网络拓扑结构，并可以查看模型所使用的算子：

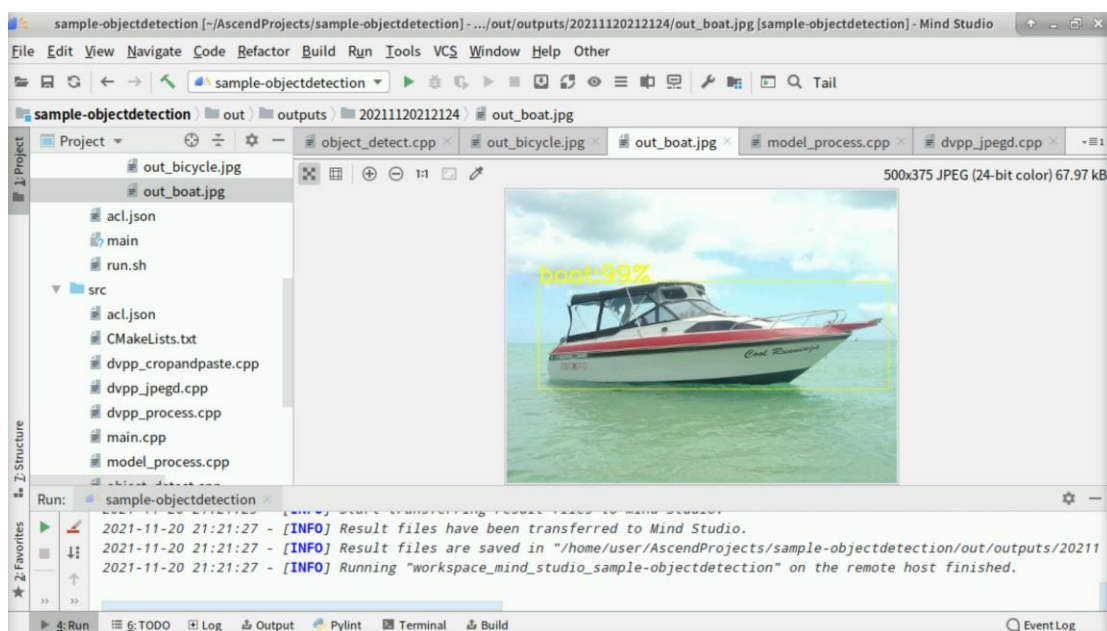
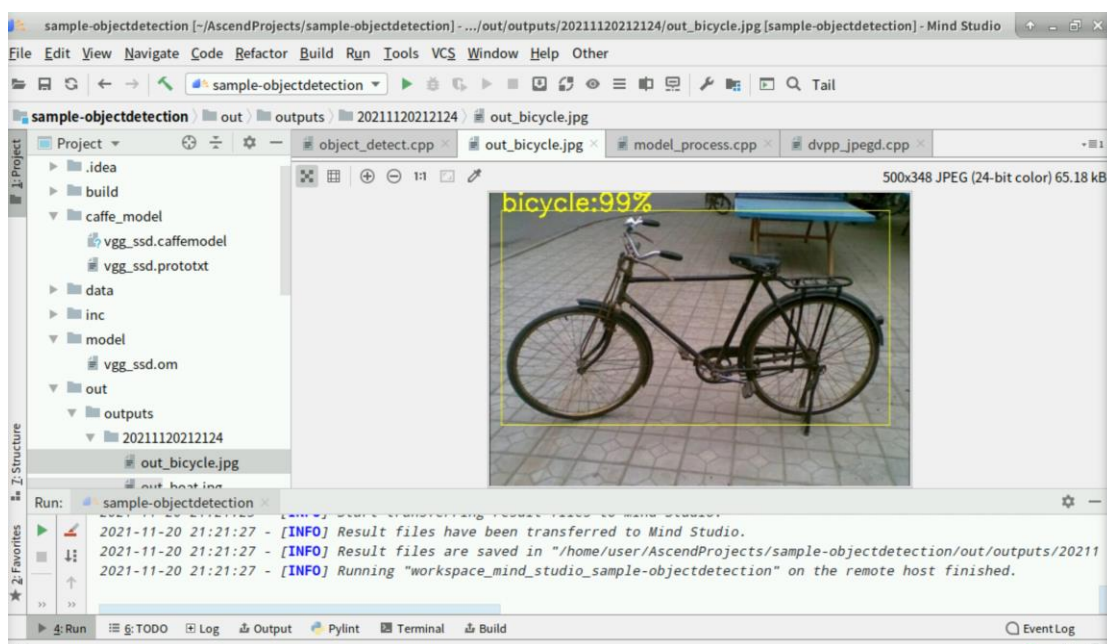
https://support.huaweicloud.com/usermanual-mindstudio302/atlasms_02_0060.html

依次单击 MindStudio 菜单栏 “Ascend > Model Visualizer”，选择转换成功的.om 模型，将 om 模型可视化界面截图。



3. 编译运行（30 分）

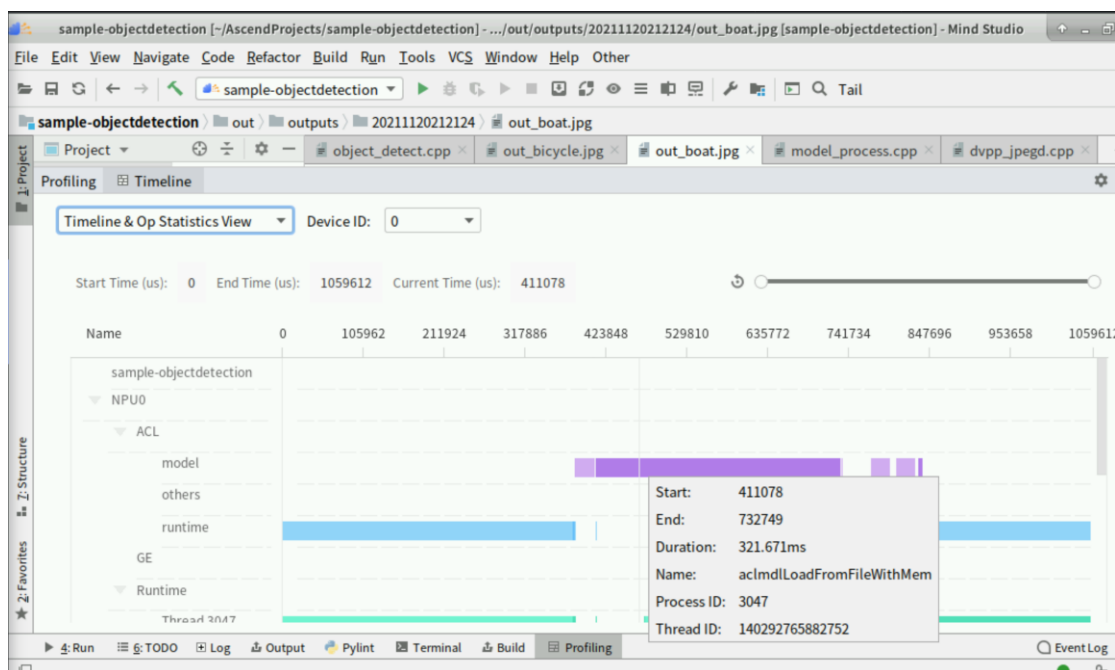
按实验手册将代码补全，并编译运行，MindStudio 会将工程发送到云服务器，完成推理后再将结果回传到沙箱环境，请将两个测试用例的推理结果截图。



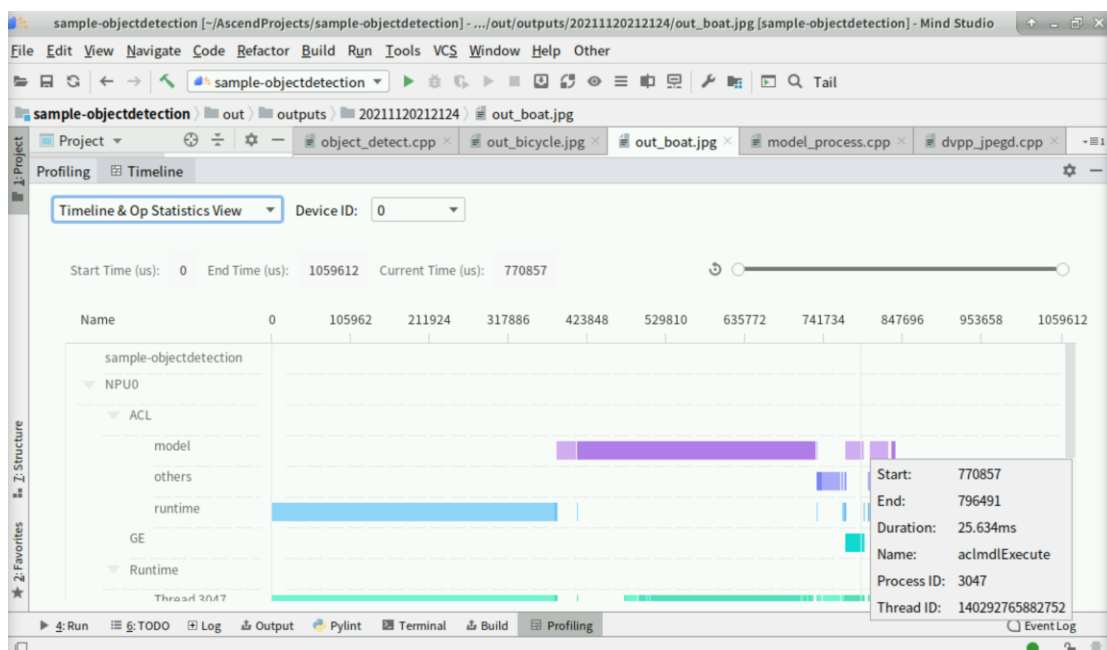
4. 模型性能（10分）

运行 Profiling 查看推理性能，截图并指出模型从文件加载进内存耗时以及两张图片推理耗时。

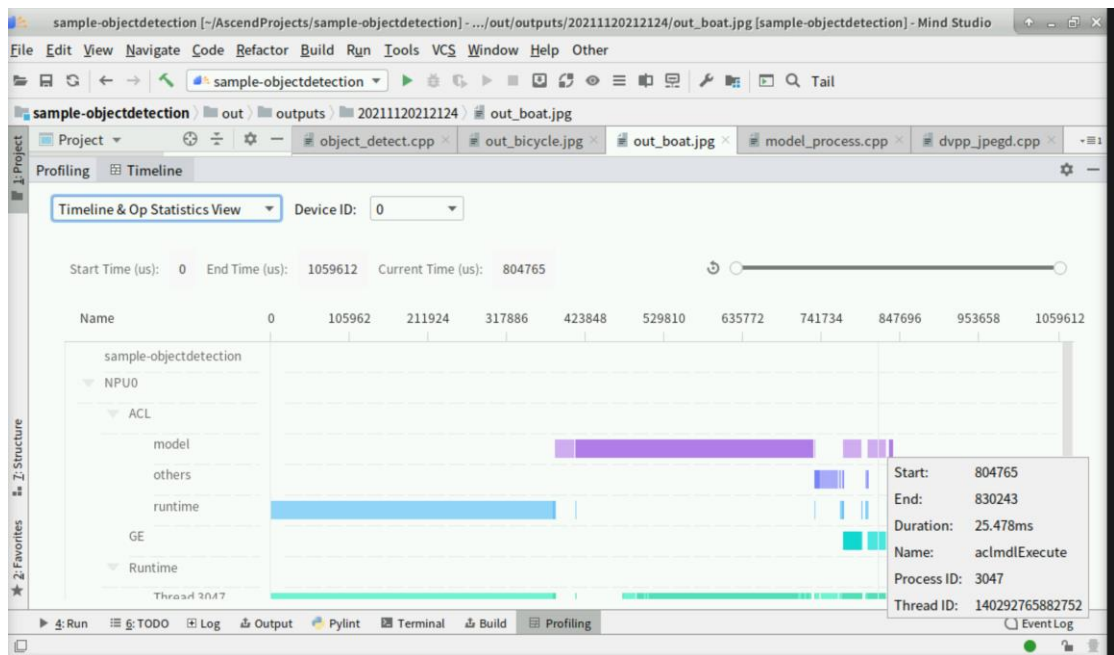
模型从文件加载进内存耗时：



第一张图像推理耗时:



第二张图像推理耗时:



五、代码分析（10 分）

（由于实验环境有时间限制，可以将代码下载到本地，参考华为实验手册与课程 PPT 分析）

源码工程：

<https://sandbox-experiment-resource-north-4.obs.cn-north-4.myhuaweicloud.com/shengteng-image-target/sample-objectdetection.zip>

简单分析：

<https://kqybjxvfo3.feishu.cn/docs/doccnr1WWpezXYWeeUykUKUpi8g#F8Ufjc>

分析 `main.cpp` 可知，代码主要有五部分：初始化、数据预处理、执行推理、数据后处理、资源销毁。

1. 初始化（包括 ASCENDCL 资源初始化、模型初始化、dvpp 资源初始化）（2 分）

```

Result ObjectDetect::Init() {
    if (isInitd_) {
        INFO_LOG("Classify instance is initied already!");
        return SUCCESS;
    }

    Result ret = InitResource();
    if (ret != SUCCESS) {
        ERROR_LOG("Init acl resource failed");
        return FAILED;
    }

    ret = InitModel(modelPath_);
    if (ret != SUCCESS) {
        ERROR_LOG("Init model failed");
        return FAILED;
    }

    ret = dvpp_.InitResource(stream_);
    if (ret != SUCCESS) {
        ERROR_LOG("Init dvpp failed");
        return FAILED;
    }

    isInitd_ = true;
    return SUCCESS;
}

```

请查看 **InitResource** 函数的实现代码，ASCENDCL 资源初始化时进行了哪些操作？

答：确定是否显式指定 **Device**，如果是，指定用于计算的 **Device**；确定是否显式创建 **Context** 和 **Stream**，如果是则创建；可选的获取软件栈的运行模式。

请查看 **InitModel** 函数的实现代码，模型初始化时进行了哪些操作？

答：模型加载；创建模型描述信息；创建模型输出；**dvpp** 初始化

2. 数据预处理（2 分）

```

Result ObjectDetect::Preprocess(ImageData& resizedImage, ImageData& srcImage) {
    ImageData yuvImage;
    Result ret = dvpp_.CvtJpegToYuv420sp(yuvImage, srcImage);
    if (ret == FAILED) {
        ERROR_LOG("Convert jpeg to yuv failed");
        return FAILED;
    }

    //resize
    ret = dvpp_.CropAndPaste(resizedImage, yuvImage, modelWidth_, modelHeight_);
    if (ret == FAILED) {
        ERROR_LOG("Resize image failed");
        return FAILED;
    }

    return SUCCESS;
}

```

这一部分主要有两个封装好的函数，其作用分别是：

CvtJpegToYuv420sp：实现 JPEG 图片解码。调用 **acldvppCreatChannel** 接口创建图片数据处理的通道；调用 **acldvppMalloc** 申请内存；调用 **acldvppJpegDecodeAsync** 异步接口进行解码；释放输入输出内存；销毁图片数据处理的通道。

CropAndPaste: 实现图片缩放

3. 执行推理（2 分）

```
Result ObjectDetect::Inference(acLmdlDataset*& inferenceOutput,
                               ImageData& resizedImage) {
    Result ret = model_.CreateInput(resizedImage.data.get(), resizedImage.size);
    if (ret != SUCCESS) {
        ERROR_LOG("Create mode input dataset failed");
        return FAILED;
    }

    ret = model_.Execute();
    if (ret != SUCCESS) {
        ERROR_LOG("Execute model inference failed");
        return FAILED;
    }

    inferenceOutput = model_.GetModelOutputData();

    return SUCCESS;
}
```

请查看 **CreateInput** 函数的实现代码，模型输入数据是怎样构造的？

答：首先需要创建一个 **dataset**，之后创建一个输入数据的缓冲区。在读取数据的时候

从缓冲区想向 **dataset** 中添加数据。在完成数据传输之后释放缓存区空间。

4. 数据后处理（2 分）

```
Result ObjectDetect::Postprocess(acLmdlDataset* modelOutput, const string& path)
{
    size_t outDatasetNum = acLmdlGetDatasetNumBuffers(modelOutput);
    if (outDatasetNum != 2) {
        ERROR_LOG("outDatasetNum=%zu must be 2", outDatasetNum);
        return FAILED;
    }
    acLdataBuffer* dataBuffer = acLmdlGetDatasetBuffer(modelOutput, 0);
    if (dataBuffer == nullptr) {
        ERROR_LOG("get model output acLmdlGetDatasetBuffer failed");
        return FAILED;
    }
    void* data = acLGetDataBufferAddr(dataBuffer);
    if (data == nullptr) {
        ERROR_LOG("acLGetDataBufferAddr from dataBuffer failed.");
        return FAILED;
    }
}
```

```

float outInfo[dataBufferSize/sizeof(float)];
ret = aclrtMemcpy(outInfo, sizeof(outInfo), data, sizeof(outInfo), ACL_MEMCPY_DEVICE_TO_HOST);
if (ret != ACL_ERROR_NONE) {
    ERROR_LOG("box outInfo aclrtMemcpy failed!");
    return FAILED;
}

cv::Rect rect;
int font_face = 0;
double font_scale = 1;
int thickness = 2;
int baseline;
cv::Mat resultImage = cv::imread(path, CV_LOAD_IMAGE_COLOR);

for(uint32_t b=0;b<BBOX_MAX;b++) {
    uint32_t score=uint32_t(outInfo[SCORE+BBOXINFO_SIZE*b]*100);
    if(score<85) continue;
    //TODO:解析模型推理结果
    rect.x=outInfo[TOPLEFTX+BBOXINFO_SIZE*b]*resultImage.cols;
    rect.y=outInfo[TOPLEFTY+BBOXINFO_SIZE*b]*resultImage.rows;
    rect.width=outInfo[BOTTOMRIGHTX+BBOXINFO_SIZE*b]*resultImage.cols-rect.x;
    rect.height=outInfo[BOTTOMRIGHTY+BBOXINFO_SIZE*b]*resultImage.rows-rect.y;
    uint32_t objIndex = (uint32_t)outInfo[LABEL+BBOXINFO_SIZE*b];
    string text = vggssdLabel[objIndex]+" "+std::to_string(score)+"%";
    cv::Point origin;
    origin.x = rect.x;
    origin.y = rect.y;
    cv::putText(resultImage, text, origin, font_face, font_scale, cv::Scalar(0, 255, 255), thickness, 4, 0);
    cv::rectangle(resultImage, rect, cv::Scalar(0, 255, 255),1, 8,0);
    //END
}

```

上图为后处理的代码，它将模型输出的每一个检测结果（包括边界框点的位置、检测标签、置信度）在输入图像进行标注，简单分析一个检测结果占用多少字节？

答：resultImage.cols* resultImage.rows/8

5. 资源销毁（2分）

```

void ObjectDetect::DestroyResource()
{
    model_.Unload();
    model_.DestroyDesc();
    model_.DestroyInput();
    model_.DestroyOutput();
    dvpp_.DestroyResource();

    aclError ret;
    if (stream_ != nullptr) {
        ret = aclrtDestroyStream(stream_);
        if (ret != ACL_ERROR_NONE) {
            ERROR_LOG("destroy stream failed");
        }
        stream_ = nullptr;
    }
    INFO_LOG("end to destroy stream");

    if (context_ != nullptr) {
        ret = aclrtDestroyContext(context_);
        if (ret != ACL_ERROR_NONE) {
            ERROR_LOG("destroy context failed");
        }
        context_ = nullptr;
    }
    INFO_LOG("end to destroy context");

    ret = aclrtResetDevice(deviceId_);
    if (ret != ACL_ERROR_NONE) {
        ERROR_LOG("reset device failed");
    }
    INFO_LOG("end to reset device is %d", deviceId_);

    //TODO:acl去初始化
    ret = aclFinalize();
    if (ret != ACL_ERROR_NONE) {
        ERROR_LOG("finalize acl failed");
    }
    INFO_LOG("end to finalize acl");

    //END
}

ObjectDetect::~ObjectDetect() {
    DestroyResource();
}

```

`~ObjectDetect()`为析构函数，在该对象生命周期结束时自动调用，释放内存。其会调用`DestroyResource()`函数，该函数进行了哪些操作？

答：销毁 `resizeConfig`，销毁 `dvppChannelDesc`

六、实验感想

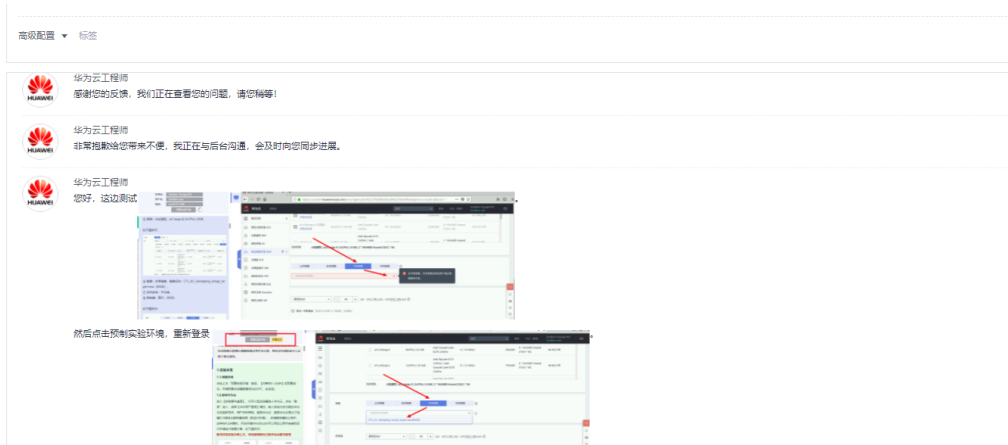
经过本次实验对于使用 ACL 实现基于昇腾 CANN 的目标检测应用有了一定的了解。通过此次实验对于昇腾 CANN 产生了一定的兴趣，也学习到了 ACL 接口的一些 API 的调用方式，虽然代码并不是自己实现的，但是通过了解整个代码框架的实现以及对于一些函数功能的分析对于实验也有了一定的了解。

附录：FAQ

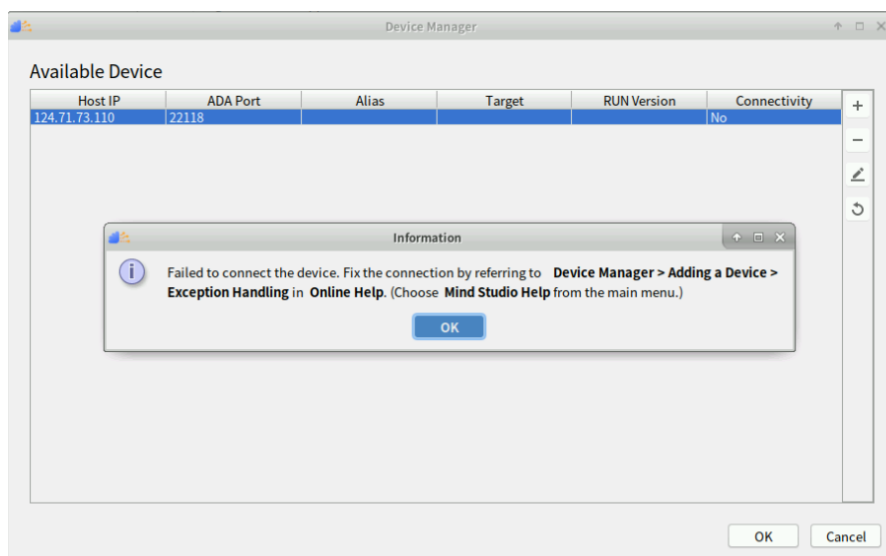
1. 找不到共享镜像？



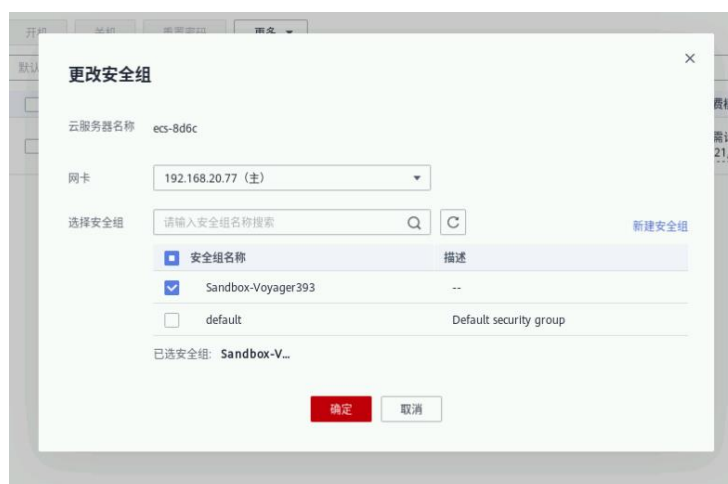
预置实验环境再重新登陆：

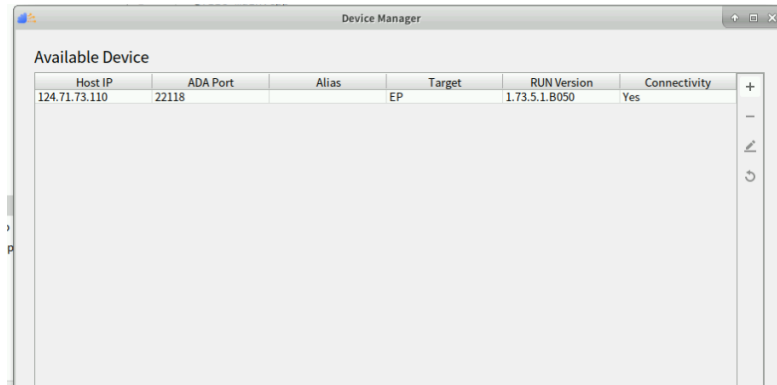


2. MindStudio 无法连接云服务器？



更改安全组策略即可：





3. MindStudio 运行项目，提示 ERROR，并且错误信息中 IP 地址和端口与自己在 Device Manager 设置的不一致？

原因：MindStudio 可能存在 bug，由于某些原因，.idea/workspace.xml 中 hostip 没有改变

解决一：将 Device Manager 中配置好的删掉，再重新配置一遍

解决二：按下图所示，选择 AI Core，并 Apply，再取消 AI Core，再 Apply

