

ModelArts

# 常见问题

文档版本 01  
发布日期 2025-01-22



版权所有 © 华为云计算技术有限公司 2025。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



HUAWEI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云计算技术有限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，华为云计算技术有限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。

# 华为云计算技术有限公司

地址：贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址：<https://www.huaweicloud.com/>

# 目录

<b>1 权限相关</b>	<b>1</b>
1.1 使用 ModelArts 时提示“权限不足”，如何解决？	1
1.2 在 Notebook 中如何实现 IAM 用户隔离？	4
1.3 如何获取访问密钥？	4
<b>2 存储相关</b>	<b>6</b>
2.1 在 ModelArts 中如何查看 OBS 目录下的所有文件？	6
<b>3 Standard 自动学习</b>	<b>7</b>
3.1 ModelArts 自动学习与 ModelArts PRO 的区别是什么？	7
3.2 在 ModelArts 中图像分类和物体检测具体是什么？	7
3.3 在 ModelArts 自动学习中模型训练图片异常怎么办？	9
3.4 在 ModelArts 自动学习中，如何进行增量训练？	9
3.5 创建自动学习项目时，如何快速创建 OBS 桶及文件夹？	10
3.6 自动学习生成的模型，存储在哪里？支持哪些其他操作？	11
3.7 自动学习训练后的模型是否可以下载？	12
<b>4 Standard Workflow</b>	<b>13</b>
4.1 如何定位 Workflow 运行报错	13
<b>5 Standard 数据准备</b>	<b>14</b>
5.1 在 ModelArts 数据集中添加图片对图片大小有限制吗？	14
5.2 如何将本地标注的数据导入 ModelArts？	14
5.3 在 ModelArts 中数据标注完成后，标注结果存储在哪里？	14
5.4 在 ModelArts 中如何将标注结果下载至本地？	15
5.5 在 ModelArts 中进行团队标注时，为什么团队成员收不到邮件？	16
5.6 ModelArts 团队标注的数据分配机制是什么？	16
5.7 如何将两个 ModelArts 数据集合并？	16
5.8 在 ModelArts 中同一个账户，图片展示角度不同是为什么？	17
5.9 在 ModelArts 中智能标注完成后新加入数据需要重新训练吗？	18
5.10 在 ModelArts 中如何将图片划分到验证集或者训练集？	18
5.11 在 ModelArts 中物体检测标注时能否自定义标签？	19
5.12 ModelArts 数据集新建的版本找不到怎么办？	19
5.13 如何切分 ModelArts 数据集？	20
5.14 如何删除 ModelArts 数据集中的图片？	20

<b>6 Standard Notebook</b> .....	<b>22</b>
6.1 ModelArts 的 Notebook 是否支持 Keras 引擎? .....	22
6.2 如何在 ModelArts 的 Notebook 中上传下载 OBS 文件? .....	23
6.3 ModelArts 的 Notebook 实例 upload 后, 数据会上传到哪里? .....	25
6.4 在 ModelArts 中如何将 Notebook A 的数据复制到 Notebook B 中? .....	25
6.5 在 ModelArts 的 Notebook 中如何对 OBS 的文件重命名? .....	25
6.6 在 ModelArts 的 Notebook 中如何使用 pandas 库处理 OBS 桶中的数据? .....	26
6.7 在 ModelArts 的 Notebook 中, 如何访问其他账号的 OBS 桶? .....	26
6.8 在 ModelArts 的 Notebook 中 JupyterLab 默认工作路径是什么? .....	26
6.9 如何查看 ModelArts 的 Notebook 使用的 cuda 版本? .....	26
6.10 在 ModelArts 的 Notebook 中如何获取本机外网 IP? .....	27
6.11 ModelArts 的 Notebook 有代理吗? 如何关闭? .....	27
6.12 在 ModelArts 的 Notebook 中内置引擎不满足使用时, 如何自定义引擎 IPython Kernel? .....	27
6.13 在 ModelArts 的 Notebook 中如何将 git clone 的 py 文件变为 ipynb 文件? .....	30
6.14 在 ModelArts 的 Notebook 实例重启时, 数据集会丢失吗? .....	31
6.15 在 ModelArts 的 Notebook 的 Jupyterlab 可以安装插件吗? .....	31
6.16 在 ModelArts 的 Notebook 的 CodeLab 中能否使用昇腾卡进行训练? .....	33
6.17 如何在 ModelArts 的 Notebook 的 CodeLab 上安装依赖? .....	34
6.18 在 ModelArts 的 Notebook 中安装远端插件时不稳定要怎么办? .....	35
6.19 在 ModelArts 的 Notebook 中实例重新启动后要怎么连接? .....	36
6.20 在 ModelArts 的 Notebook 中使用 VS Code 调试代码无法进入源码怎么办? .....	37
6.21 在 ModelArts 的 Notebook 中使用 VS Code 如何查看远端日志? .....	38
6.22 在 ModelArts 的 Notebook 中如何打开 VS Code 的配置文件 settings.json? .....	38
6.23 在 ModelArts 的 Notebook 中如何设置 VS Code 背景色为豆沙绿? .....	38
6.24 在 ModelArts 的 Notebook 中如何设置 VS Code 远端默认安装的插件? .....	38
6.25 在 ModelArts 的 VS Code 中如何把本地插件安装到远端或把远端插件安装到本地? .....	39
6.26 在 ModelArts 的 Notebook 中, 如何使用昇腾多卡进行调试? .....	39
6.27 在 ModelArts 的 Notebook 中使用不同的资源规格训练时为什么训练速度差不多? .....	39
6.28 在 ModelArts 的 Notebook 中使用 MoXing 时, 如何进行增量训练? .....	40
6.29 在 ModelArts 的 Notebook 中如何查看 GPU 使用情况? .....	42
6.30 在 ModelArts 的 Notebook 中如何在代码中打印 GPU 使用信息? .....	43
6.31 在 ModelArts 的 Notebook 中 JupyterLab 的目录、Terminal 的文件和 OBS 的文件之间的关系是什么? .....	46
6.32 如何在 ModelArts 的 Notebook 实例中使用 ModelArts 数据集? .....	46
6.33 pip 介绍及常用命令.....	46
6.34 在 ModelArts 的 Notebook 中不同规格资源/cache 目录的大小是多少? .....	46
6.35 资源超分对在 ModelArts 的 Notebook 实例有什么影响? .....	47
6.36 如何在 Notebook 中安装外部库? .....	47
6.37 在 ModelArts 的 Notebook 中, 访问外网速度不稳定怎么办? .....	48
<b>7 Standard 模型训练</b> .....	<b>49</b>
7.1 在 ModelArts 训练得到的模型欠拟合怎么办? .....	49
7.2 在 ModelArts 中训练好后的模型如何获取? .....	49

7.3 在 ModelArts 上如何获得 RANK_TABLE_FILE 用于分布式训练? .....	50
7.4 在 ModelArts 上训练模型如何配置输入输出数据? .....	50
7.5 在 ModelArts 上如何提升训练效率并减少与 OBS 的交互? .....	51
7.6 在 ModelArts 中使用 Moxing 复制数据时如何定义路径变量? .....	51
7.7 在 ModelArts 上如何创建引用第三方依赖包的训练作业? .....	52
7.8 在 ModelArts 训练时如何安装 C++的依赖库? .....	53
7.9 在 ModelArts 训练作业中如何判断文件夹是否复制完毕? .....	53
7.10 如何在 ModelArts 训练作业中加载部分训练好的参数? .....	54
7.11 ModelArts 训练时使用 os.system('cd xxx')无法进入文件夹怎么办? .....	54
7.12 在 ModelArts 训练代码中, 如何获取依赖文件所在的路径? .....	54
7.13 自如何获取 ModelArts 训练容器中的文件实际路径? .....	55
7.14 ModelArts 训练中不同规格资源 “/cache” 目录的大小是多少? .....	55
7.15 ModelArts 训练作业为什么存在/work 和/ma-user 两种超参目录? .....	56
7.16 如何查看 ModelArts 训练作业资源占用情况? .....	57
7.17 如何将在 ModelArts 中训练好的模型下载或迁移到其他账号? .....	57
<b>8 Standard 推理部署.....</b>	<b>58</b>
8.1 如何将 Keras 的.h5 格式的模型导入到 ModelArts 中? .....	58
8.2 ModelArts 导入模型时, 如何编写模型配置文件中的安装包依赖参数? .....	58
8.3 在 ModelArts 中使用自定义镜像创建在线服务, 如何修改端口? .....	60
8.4 ModelArts 平台是否支持多模型导入? .....	60
8.5 在 ModelArts 中导入模型对于镜像大小有什么限制? .....	61
8.6 ModelArts 在线服务和批量服务有什么区别? .....	61
8.7 ModelArts 在线服务和边缘服务有什么区别? .....	61
8.8 在 ModelArts 中部署模型时, 为什么无法选择 Ascend Snt3 资源? .....	62
8.9 ModelArts 线上训练得到的模型是否支持离线部署在本地? .....	62
8.10 ModelArts 在线服务预测请求体大小限制是多少? .....	64
8.11 ModelArts 部署在线服务时, 如何避免自定义预测脚本 python 依赖包出现冲突? .....	64
8.12 ModelArts 在线服务预测时, 如何提高预测速度? .....	64
8.13 在 ModelArts 中调整模型后, 部署新版本模型能否保持原 API 接口不变? .....	64
8.14 ModelArts 在线服务的 API 接口组成规则是什么? .....	65
8.15 ModelArts 在线服务处于运行中时, 如何填写 request header 和 request body? .....	66
<b>9 Standard 镜像相关.....</b>	<b>68</b>
9.1 不在同一个主账号下, 如何使用他人的自定义镜像创建 Notebook? .....	68
9.2 如何登录并上传镜像到 SWR? .....	69
9.3 在 Dockerfile 中如何给镜像设置环境变量? .....	70
9.4 如何通过 docker 镜像启动容器? .....	71
9.5 如何在 ModelArts 的 Notebook 中配置 Conda 源? .....	71
9.6 ModelArts 的自定义镜像软件版本匹配有哪些注意事项? .....	72
9.7 镜像在 SWR 上显示只有 13G, 安装少量的包, 然后镜像保存过程会提示超过 35G 大小保存失败, 为什么? .....	73
9.8 如何保证自定义镜像能不会因为超过 35G 而保存失败? .....	73
9.9 如何减小本地或 ECS 构建镜像的目的镜像的大小? .....	73

9.10 镜像过大，卸载原来的包重新打包镜像，最终镜像会变小吗？	74
9.11 在 ModelArts 镜像管理注册镜像报错 ModelArts.6787 怎么处理？	74
9.12 用户如何设置默认的 kernel？	75
<b>10 Standard 专属资源池</b>	<b>76</b>
10.1 ModelArts 支持使用 ECS 创建专属资源池吗？	76
10.2 在 ModelArts 中 1 个节点的专属资源池，能否部署多个服务？	76
10.3 在 ModelArts 中公共资源池和专属资源池的区别是什么？	77
10.4 ModelArts 中的作业为什么一直处于等待中？	77
10.5 ModelArts 控制台为什么能看到创建失败被删除的专属资源池？	77
10.6 ModelArts 训练专属资源池如何与 SFS 弹性文件系统配置对等链接？	78
<b>11 Edge</b>	<b>79</b>
11.1 在 ModelArts 中使用边缘节点部署边缘服务时能否使用 http 接口协议？	79
<b>12 API/SDK</b>	<b>80</b>
12.1 ModelArts SDK、OBS SDK 和 MoXing 的区别是什么？	80
12.2 ModelArts 的 API 或 SDK 支持模型下载到本地吗？	81
12.3 ModelArts 通过 OBS 的 API 访问 OBS 中的文件，属于内网还是公网访问？	81
12.4 调用 ModelArts API 接口创建训练作业和部署服务时，如何填写资源池的参数？	81
<b>13 Lite Server</b>	<b>82</b>
13.1 GPU A 系列裸金属服务器如何进行 RoCE 性能带宽测试？	82
13.2 GPU A 系列裸金属服务器节点内如何进行 NVLINK 带宽性能测试方法？	84
13.3 如何将 Ubuntu20.04 内核版本从低版本升级至 5.4.0-144-generic？	85
13.4 如何禁止 Ubuntu 20.04 内核自动升级？	86
13.5 哪里可以了解 Atlas800 训练服务器硬件相关内容？	87
13.6 使用 GPU A 系列裸金属服务器有哪些注意事项？	88
13.7 GPU A 系列裸金属服务器如何更换 NVIDIA 和 CUDA？	88
<b>14 Lite Cluster</b>	<b>90</b>
14.1 Cluster 资源池如何进行 NCCL Test？	90
<b>15 历史文档待下线</b>	<b>91</b>
15.1 ModelArts 与其他服务的关系	91
15.2 如何上传数据至 OBS？	93

# 1 权限相关

## 1.1 使用 ModelArts 时提示“权限不足”，如何解决？

当您使用ModelArts时如果提示权限不足，请您按照如下指导对相关服务和用户进行授权，并对用户权限进行检查操作。

本案例中以OBS权限不足为例，介绍如何为用户授予OBS服务权限。其它权限不足的场景也可以参考本案例操作，只是授权范围不同。不同业务场景下的授权范围请参考[权限依赖和委托](#)章节。

由于ModelArts的使用权限依赖OBS服务的授权，您需要为用户授予OBS的系统权限。

- 如果您需要授予用户关于OBS的所有权限和ModelArts的基础操作权限，请参见[配置基础操作权限](#)。
- 如果您需要对用户使用OBS和ModelArts的权限进行精细化管理，进行自定义策略配置，请参见[创建ModelArts自定义策略](#)。

### 配置基础操作权限

使用ModelArts的基本功能，您需要为用户配置“作用范围”为“项目级服务”的“ModelArts CommonOperations”权限，由于ModelArts依赖OBS权限，您还需要登录IAM管理控制台为用户授予“作用范围”为“全局级服务”的“OBS Administrator”策略。

具体操作步骤如下：

#### 步骤1 创建用户组。

[登录IAM管理控制台](#)，单击“用户组>创建用户组”。在“创建用户组”界面，输入“用户组名称”单击“确定”。

#### 步骤2 配置用户组权限。

在用户组列表中，单击步骤1新建的用户组右侧的“授权”，在用户组“授权”页面，您需要配置的权限如下：

1. 配置“作用范围”为“项目级服务”的“ModelArts CommonOperations”权限，如下图所示，然后单击“确定”完成授权。

### 📖 说明

区域级项目授权后只在授权区域生效，如果需要所有区域都生效，则所有区域都需要进行授权操作。

2. 配置“作用范围”为“全局级服务”的“OBS Administrator”权限，然后单击“确定”完成授权。

### 步骤3 创建用户并加入用户组。

在IAM控制台创建用户，并将其加入步骤1中创建的用户组。

### 步骤4 用户登录并验证权限。

新创建的用户登录控制台，切换至授权区域，验证权限：

- 在“服务列表”中选择ModelArts，进入ModelArts主界面，选择不同类型的专属资源池，在页面单击“创建”，如果无法进行创建（当前权限仅包含ModelArts CommonOperations），表“ModelArts CommonOperations”已生效。
- 在“服务列表”中选择除ModelArts外（假设当前策略仅包含ModelArts CommonOperations）的任一服务，如果提示权限不足，表示“ModelArts CommonOperations”已生效。
- 在“服务列表”中选择ModelArts，进入ModelArts主界面，单击“数据管理>数据集>创建数据 > 集”，如果可以成功访问对应的OBS路径，表示全局级服务的“OBS Administrator”已生效。

----结束

## 创建 ModelArts 自定义策略

如果系统预置的ModelArts权限不满足您的授权要求，或者您需要管理用户操作OBS的操作权限，可以创建自定义策略。更多关于创建自定义策略操作和参数说明请参见[创建自定义策略](#)。

目前华为云支持可视化视图创建自定义策略和JSON视图创建自定义策略，本章节将使用JSON视图方式的策略，以为ModelArts用户授予开发环境的使用权限并且配置ModelArts用户OBS相关的最小化权限项为例，指导您进行自定义策略配置。

### 📖 说明

如果一个自定义策略中包含多个服务的授权语句，这些服务必须是同一属性，即都是全局级服务或者项目级服务。

由于OBS为全局服务，ModelArts为项目级服务，所以需要创建两条“作用范围”别为“全局级服务”以及“项目级服务”的自定义策略，然后将两条策略同时授予用户。

1. 创建ModelArts相关OBS的最小化权限的自定义策略。

登录IAM控制台，在“权限管理>权限”页面，单击“创建自定义策略”。参数配置说明如下：

- “策略名称”支持自定义。
- “策略配置方式”为“JSON视图”。
- “策略内容”请参见[ModelArts依赖的OBS权限自定义策略样例](#)，如果您需要了解更多关于OBS的系统权限，请参见[OBS权限管理](#)。

2. 创建ModelArts开发环境的使用权限的自定义策略。参数配置说明如下：

- “策略名称”支持自定义。



- “策略配置方式”为“JSON视图”。
  - “策略内容”请参见[ModelArts开发环境使用权限的自定义策略样例](#)，ModelArts自定义策略中可以添加的授权项（Action）请参见《[ModelArts API参考](#)》>权限策略和授权项。
  - 如果您需要对除ModelArts和OBS之外的其它服务授权，IAM支持服务的所有策略请参见[权限策略](#)。
3. 在IAM控制台[创建用户组并授权](#)。  
在IAM控制台创建用户组之后，将步骤1中创建的自定义策略授权给该用户组。
  4. [创建用户并加入用户组](#)。  
在IAM控制台创建用户，并将其加入3中创建的用户组。
  5. [用户登录](#)并验证权限。  
新创建的用户登录控制台，切换至授权区域，验证权限：
    - 在“服务列表”中选择ModelArts，进入ModelArts主界面，单击“数据管理>数据集”，如果无法进行创建（当前仅包含开发环境的使用权限），表示仅为ModelArts用户授予开发环境的使用权限已生效。
    - 在“服务列表”中选择除ModelArts，进入ModelArts主界面，单击“开发环境>Notebook>创建”，如果可以成功访问“存储配置”项对应的OBS路径，表示为用户配置的OBS相关权限已生效。

## ModelArts 依赖的 OBS 权限自定义策略样例

如下示例为ModelArts依赖OBS服务的最小化权限项，包含OBS桶和OBS对象的权限。授予示例中的权限您可以通过ModelArts正常访问OBS不受限制。

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Action": [
        "obs:bucket:ListAllMybuckets",
        "obs:bucket:HeadBucket",
        "obs:bucket:ListBucket",
        "obs:bucket:GetBucketLocation",
        "obs:object:GetObject",
        "obs:object:GetObjectVersion",
        "obs:object:PutObject",
        "obs:object:DeleteObject",
        "obs:object:DeleteObjectVersion",
        "obs:object:ListMultipartUploadParts",
        "obs:object:AbortMultipartUpload",
        "obs:object:GetObjectAcl",
        "obs:object:GetObjectVersionAcl",
        "obs:bucket:PutBucketAcl",
        "obs:object:PutObjectAcl"
      ],
      "Effect": "Allow"
    }
  ]
}
```

## ModelArts 开发环境使用权限的自定义策略样例

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
```

```
"Action": [  
  "modelarts:notebook:list",  
  "modelarts:notebook:create",  
  "modelarts:notebook:get",  
  "modelarts:notebook:update",  
  "modelarts:notebook:delete",  
  "modelarts:notebook:action",  
  "modelarts:notebook:access"  
]  
}  
]
```

## 1.2 在 Notebook 中如何实现 IAM 用户隔离？

开发环境如果需要通过IAM实现用户隔离，即多个IAM用户之间无法查看、修改和删除他人创建的Notebook。

目前有两种方案：

- 方案一：删除modelarts:notebook:listAllNotebooks细粒度权限。
- 方案二：使用[工作空间](#)功能：目前工作空间功能是“受邀开通”状态，作为企业用户您可以通过您对口的技术支持申请开通。

## 1.3 如何获取访问密钥？

### 获取访问密钥

1. 登录[华为云](#)，在页面右上方单击“控制台”，进入华为云管理控制台。

图 1-1 控制台入口



2. 在控制台右上角的账户名下方，单击“我的凭证”，进入“我的凭证”页面。

图 1-2 我的凭证



3. 在“我的凭证”页面，选择“访问密钥>新增访问密钥”，如图1-3所示。

图 1-3 单击新增访问密钥



4. 填写该密钥的描述说明，单击“确定”。根据提示单击“立即下载”，下载密钥。

图 1-4 新增访问密钥



5. 密钥文件会直接保存到浏览器默认的下载文件夹中。打开名称为“credentials.csv”的文件，即可查看访问密钥（Access Key Id和Secret Access Key）。

# 2 存储相关

## 2.1 在 ModelArts 中如何查看 OBS 目录下的所有文件？

在使用Notebook或训练作业时，需要查看目录下的所有文件，您可以通过如下方式实现：

- 通过OBS管理控制台进行查看。  
使用当前账户登录OBS管理控制台，去查找对应的OBS桶、文件夹、文件。
- 通过接口判断路径是否存在。在已有的Notebook实例，或者创建一个Notebook，执行如下命令，检查路径是否存在。

```
import moxing as mox
mox.file.list_directory('obs://bucket_name', recursive=True)
```

如果文件较多，请您耐心等待，最终文件路径信息会在提示信息之后显示。

# 3 Standard 自动学习

## 3.1 ModelArts 自动学习与 ModelArts PRO 的区别是什么？

ModelArts自动学习，提供了AI初学者，零编码、零AI基础情况下，可使用自动学习功能，开发用于图像分类、物体检测、预测分析、文本分类、声音分类等场景的模型。

而ModelArts PRO是一款为企业级AI应用打造的专业开发套件。用户可根据预置 workflow 生成指定场景模型，无需深究底层模型开发细节。ModelArts PRO底层依托 ModelArts平台提供数据标注、模型训练、模型部署等能力。也可以理解成增强版的自动学习，提供行业AI定制化开发套件，沉淀行业知识，让开发者聚焦自身业务。

## 3.2 在 ModelArts 中图像分类和物体检测具体是什么？

图像分类是根据各自在图像信息中所反映的不同特征，把不同类别的目标区分开来的图像处理方法。它利用计算机对图像进行定量分析，把图像或图像中的每个像元或区域划归为若干个类别中的某一种，以代替人的视觉判读。简单的说就是识别一张图中是否是某类/状态/场景，适合图中主体相对单一的场景，将下图识别为汽车的图片。

图 3-1 图像分类



物体检测是计算机视觉中的经典问题之一，其任务是用框去标出图像中物体的位置，并给出物体的类别。通常在一张图包含多个物体的情况下，定制识别出每个物体的位置、数量、名称，适合图片中有多个主体的场景，针对下图检测出图片包含树和汽车。

图 3-2 物体检测



### 3.3 在 ModelArts 自动学习中模型训练图片异常怎么办？

使用自动学习的图像分类或物体检测算法时，标注完成的数据在进行模型训练后，训练结果为图片异常。针对不同的异常情况说明及解决方案参见表3-1。

表 3-1 自动学习训练中图片异常情况说明（图像分类和物体检测）

序号	图片异常显示字段	图片异常说明	解决方案字段	解决方案说明
1	load failed	图片无法被解码且不能修复	ignore	系统已自动跳过这张图片，不需要用户处理。
2	tf-decode failed	图片无法被 TensorFlow 解码且不能修复	ignore	系统已跳过这张图片，不需要用户处理。
3	size over	图片大于5MB	resize to small	系统已将图片压缩到5MB以内处理，不需要用户处理。
4	mode illegal	图片非RGB模式	convert to rgb	系统已将图片转成RGB格式处理，不需要用户处理。
5	type illegal	非图片文件，但可以转换成JPG	convert to jpg	系统已将图片转换成JPG格式处理，不需要用户处理。

### 3.4 在 ModelArts 自动学习中，如何进行增量训练？

在自动学习项目中，每训练一次，将自动产生一个训练版本。当前一次的训练结果不满意时（如对训练精度不满意），您可以适当增加高质量的数据，或者增减标签，然后再次进行训练。

#### 📖 说明

- 增量训练目前仅支持“图像分类”、“物体检测”、“声音分类”类型的自动学习项目。
- 为提升训练效果，建议在增量训练时，选择质量较高的数据，提升数据标注的质量。

#### 增量训练的操作步骤

1. 登录ModelArts管理控制台，单击左侧导航栏的自动学习。
2. 在自动学习项目管理页面，单击对应的项目名称，进入此项目的自动学习详情页。
3. 在数据标注页面，单击未标注页签，在此页面中，您可以单击添加图片，或者增删标签。

如果增加了图片，您需要对增加的图片进行重新标注。如果您增删标签，建议对所有的图片进行排查和重新标注。对已标注的数据，也需要检查是否需要增加新的标签。

4. 在图片都标注完成后，单击右上角“开始训练”，在“训练设置”中，在“增量训练版本”中选择之前已完成的训练版本，在此版本基础上进行增量训练。其他参数请根据界面提示填写。

设置完成后，单击“确定”，即进行增量训练。系统将自动跳转至“模型训练”页面，待训练完成后，您可以在此页面中查看训练详情，如“训练精度”、“评估结果”、“训练参数”等。

图 3-3 选择增量训练版本

### 训练设置

* 数据集版本名称	<input type="text" value="V004"/>
训练验证比例 ?	训练集比例: <input type="text" value="0.8"/> ? 验证集比例: 0.2
增量训练版本 ?	<input type="text" value="V001"/>
最大训练时长 (分钟)	<input type="text" value="60"/>
训练偏好 ?	<input type="text" value="balance"/>
计算规格	<input type="text" value="增强计算型1实例-自动学习 (GPU)"/>

## 3.5 创建自动学习项目时，如何快速创建 OBS 桶及文件夹？

在创建项目时需要选择训练数据路径，本章节将指导您如何在选择训练数据路径时，快速创建OBS桶和OBS文件夹。


1. 在创建自动学习项目页面，单击数据集输入位置右侧的“”按钮，进入“数据集输入位置”对话框。
2. 单击“新建对象存储服务（OBS）桶”，进入创建桶页面，具体请参见《对象存储服务控制台指南》中的[创建桶](#)章节。



图 3-4 快速创建 OBS 桶



- 桶创建完成后，选择对应桶名称，单击“新建文件夹”，在“新建文件夹”对话框中，填写文件夹“名称”，单击“确定”完成创建，选择创建的文件夹。
  - 文件夹名称不能包含以下字符：\:\*? "<>|。
  - 文件夹名称不能以英文句号（.）或斜杠（/）开头或结尾。
  - 文件夹的绝对路径总长度不能超过1023字符。
  - 任何单个斜杠（/）表示分隔并创建多层级的文件夹。

图 3-5 新建文件夹



## 3.6 自动学习生成的模型，存储在哪里？支持哪些其他操作？

### 模型统一管理

针对自动学习项目，当模型训练完成后，其生成的模型，将自动进入“模型管理”页面，如下图所示。模型名称由系统自动命名，前缀与自动学习项目的名称一致，方便辨识。

#### ⚠ 注意

自动学习生成的模型，不支持下载使用。

图 3-6 自动学习生成的模型

AI应用名称	最新版本	状态	部署类型
▼ huahua@huahua-test	0.0.1	✔ 正常	在线服务
▼ MA-Service-model-11-15-18-54	1.0.0	✔ 正常	在线服务/批量服务/边缘服务
▼ model-60447285-flower	0.0.1	✔ 正常	在线服务/批量服务/边缘服务

### 自动学习生成的模型，支持哪些其他操作

- **支持部署为在线服务、批量服务或边缘服务。**  
在自动学习页面中，仅支持部署为在线服务，如需部署为批量服务或边缘服务，可在“模型部署”页面部署。
- **支持发布至市场**  
将产生的模型发布至AI Gallery，共享给其他用户。
- **支持创建新版本**  
创建新版本，仅支持从ModelArts训练作业、OBS、模型模板、或自定义镜像中选择元模型。无法从原自动学习项目中，创建新版本。
- **支持删除模型或其模型版本**

## 3.7 自动学习训练后的模型是否可以下载？

不可以下载。但是您可以在AI应用管理页面查看，或者将此模型部署为在线服务。

# 4 Standard Workflow

## 4.1 如何定位 Workflow 运行报错

使用run模式运行工作流报错时，分析解决思路如下：

1. 确认安装的SDK包是否是最新版本，避免出现包版本不一致问题。
2. 检查编写的SDK代码是否符合规范，具体可参考相应的代码示例。
3. 检查运行过程中输入的内容是否正确，格式是否与提示信息中要求的一致。
4. 根据具体报错信息定位到报错的代码行，分析上下文逻辑。

### 历史 SDK 包常见的报错如下

- 服务部署节点运行报错

```
'weight': 100, 'specification': 'custom',  
'cluster_id': '*****', 'custom_spec': {'cpu': 2.5, 'memory':  
1024}, 'envs': {'*****': '*****', '*****': '*****'}}]"
```

Note that: (1) The "[" at the beginning and "]" at the end are required.

(2) The sum of the weights must be equal to 100.

(3) All model must have the same model name. Two model versions cannot be the same.

```
[{'model_name': '*****', 'model_version': '*****', 'specification': '*****', 'weight': 100}]
```

Traceback (most recent call last):

- 输入服务相关的参数后，执行报错如下：

```
specnode.py, line 30, in __eq__  
    return self.name == obj.name and \  
  
AttributeError: 'str' object has no attribute 'name'
```

### 解决方案

以上两种常见报错均可通过升级最新的SDK包解决。

# 5 Standard 数据准备

## 5.1 在 ModelArts 数据集中添加图片对图片大小有限制吗？

在数据管理功能中，针对“物体检测”或“图像分类”的数据集，在数据集中上传更多的图片时，是有限制的。要求单张图片大小不超过8MB，且只支持JPG、JPEG、PNG和BMP四种格式的图片。

请注意，针对自动学习功能中的添加图片，其图片大小限制不同，要求上传的图片大小不超过5MB。

**解决方案：**

- 方法1：使用导入功能。将图片上传至OBS任意目录，通过“从OBS目录导入”方式导入到已有数据集。
- 方法2：使用同步数据源功能。将图片上传到数据集输入目录下（或者其子目录），单击数据集详情页中的“同步数据源”将新增图片导入。需注意的是，同步数据源同时也会将OBS已删除的文件从数据集也删除，请谨慎操作。
- 方法3：新建数据集。将图片上传至OBS任意目录，可以直接使用这些图片目录作为数据集的输入目录，新建一个数据集。

## 5.2 如何将本地标注的数据导入 ModelArts？

ModelArts支持通过导入数据集的操作，导入更多数据。本地标注的数据，当前支持从OBS目录导入或从Manifest文件导入两种方式。导入之后您还可以在ModelArts数据管理模块中对数据进行重新标注或修改标注情况。

从OBS目录导入或从Manifest详细操作指导和规范说明请参见[导入数据](#)。

## 5.3 在 ModelArts 中数据标注完成后，标注结果存储在哪里？

ModelArts管理控制台，提供了数据可视化能力，您可以在控制台查看详细数据以及标注信息。如需了解标注结果的存储路径，请参见如下说明。

## 背景说明

针对ModelArts中的数据集市，在创建数据集时，需指定“数据集输入位置”和“数据集输出位置”。两个参数填写的均是OBS路径。

- “数据集输入位置”即原始数据存储的OBS路径。
- “数据集输出位置”，指在ModelArts完成数据标注后，执行数据集发布操作后，在此指定路径下，按数据集版本，生成相关目录。包含ModelArts中使用的Manifest文件（包含数据及标注信息）。详细文件说明可参见[数据集发布后，相关文件的目录结构说明](#)。

## 查看步骤

1. 在ModelArts管理控制台，进入“数据管理>数据集”。
2. 选择需查看数据集，单击名称左侧小三角，展开数据集详情。可获得“数据集输出位置”指定的OBS路径。

### 📖 说明

获取标注信息前，需确保数据集已发布，至少有一个以上数据集版本。

图 5-1 数据集详情



3. 进入OBS管理控制台，根据上述步骤获得的路径，找到对应版本号目录，即可获取数据集对应的标注结果。

图 5-2 获取标注结果



## 5.4 在 ModelArts 中如何将标注结果下载至本地？

ModelArts数据集中的标注信息和数据在发布后，将以manifest格式存储在“数据集输出位置”对应的OBS路径下。

路径获取方式：

1. 在ModelArts管理控制台，进入“数据管理>数据集”。
2. 选择需查看数据集，单击名称左侧小三角，展开数据集详情。可获得“数据集输出位置”指定的OBS路径。
3. 进入OBS管理控制台，根据上述步骤获得的路径，找到对应版本号目录，即可获得数据集对应的标注结果。

如需将标注结果下载至本地，可前往manifest文件存储的OBS中，单击“下载”，即可将标注结果存储至本地。

图 5-3 下载标注结果



<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时...	操作
<a href="#">← 返回上一级</a>							
<input type="checkbox"/>	V001.manifest	标准存储	1.13 MB	未加密	--	2020/04/02 ...	<a href="#">下载</a> <a href="#">分享</a> <a href="#">更多</a>

## 5.5 在 ModelArts 中进行团队标注时，为什么团队成员收不到邮件？

团队标注时，成员收不到邮件的可能原因如下：

- 当数据集中的所有数据已完成标注，即“未标注”数据为空时，创建的团队标注任务，因为没有数据需要标注，不会给团队成员发送标注邮件。在发起团队标注任务时，请确保数据集中存在“未标注”数据。
- 只有当创建团队标注任务时，标注人员才会收到邮件。创建标注团队及添加标注团队的成员并不会发送邮件。
- 请确保您的邮箱已完成配置且配置无误。可参考[管理成员](#)，完成邮箱配置。
- 团队成员自检其邮箱是否有拦截设置。

## 5.6 ModelArts 团队标注的数据分配机制是什么？

目前不支持用户自定义成员任务分配，数据是平均分配的。

- 当数量和团队成员人数不成比例，无法平均分配时，则将多余的几张图片，随机分配给团队成员。
- 如果样本数少于待分配成员时，部分成员会存在未分配到样本的情况。样本只会分配给labeler，比如10000张都是未标注，且5个都是labeler的话，那就是每个人分2000。

## 5.7 如何将两个 ModelArts 数据集合并？

目前不支持直接合并。

但是可以参考如下操作方式，将两个数据集的数据合并在一个数据集中。

例如需将数据集A和数据集B进行合并。

1. 分别将数据集A和数据集B进行发布。
2. 发布后可获得数据集A和数据集B的Manifest文件。可通过数据集的“数据集输出位置”获得此文件。

3. 创建一个空数据集C，即无任何输出，其输入位置选择一个空的OBS文件夹。
4. 在数据集C中，执行导入数据操作，将数据集A和数据集B的Manifest文件导入。  
导入完成后，即将数据集A和数据集B的数据分别都合并至数据集C中。如需使用合并后的数据集，再针对数据集C执行发布操作即可。

## 5.8 在 ModelArts 中同一个账户，图片展示角度不同是为什么？

有的图片存在旋转角度等属性，不同的浏览器的处理策略不同，对浏览器的兼容性如表1和表2所示。

- L代表last，L3-产品版本上线时最新的3个稳定浏览器版本。
- 如果您当前使用的浏览器版本过低，将在一定程度上影响页面的显示效果，系统会提示您尽快对浏览器进行升级。
- 如果您当前使用的浏览器不支持访问管理控制台，系统会建议您对浏览器进行升级或安装支持的浏览器。

表 5-1 PC 端浏览器兼容性一览表

浏览器类型	版本	操作系统	兼容性
Internet Explorer	11	Windows 7	不承诺兼容。
Microsoft Edge	L3	Windows 10	完全兼容。
	<79	Windows 10	不承诺兼容。
Mozilla Firefox	L3	Windows 10	完全兼容。
	L3	CentOS 7+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。
	L3	Ubuntu 14.04 LTS+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。
	L3	macOS 10+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。
Google Chrome	L3	Windows 10	完全兼容。
	L3	CentOS 7+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。

浏览器类型	版本	操作系统	兼容性
	L3	Ubuntu 14.04 LTS+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。
	L3	macOS 10+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。
Safari	L2	macOS 10+	部分兼容。 能确保基本交互操作，但在视觉、交互效果上可能存在兼容性问题。

表 5-2 移动端浏览器兼容性一览表

浏览器类型	版本	操作系统	兼容性
Chrome	L3	Android	完全兼容。
Safari	L3	IOS	完全兼容。
UC浏览器	L3	Android	完全兼容。
QQ浏览器	L3	Android	完全兼容。
360浏览器	L3	Android	完全兼容。
百度浏览器	L3	Android	完全兼容。

## 5.9 在 ModelArts 中智能标注完成后新加入数据需要重新训练吗？

智能标注完成后，需要对标注结果进行确认。

- 如果未确认标注结果，直接加入新数据，重新智能标注，会将待确认的数据和新加入的数据全部重新训练。
- 如果确认标注结果后，再加入新数据，只重新训练标注新的数据。

## 5.10 在 ModelArts 中如何将图片划分到验证集或者训练集？

目前只能指定切分比例，随机将样本划分到训练集或者验证集，不支持指定。



## 切分比例的指定：

在发布数据集时，仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。

一般默认不启用该功能。启用后，需设置对应的训练验证比例。

输入“训练集比例”，数值只能是0~1区间内的数。设置好“训练集比例”后，“验证集比例”自动填充。“训练集比例”加“验证集比例”等于1。

“训练集比例”即用于训练模型的样本数据比例；“验证集比例”即用于验证模型的样本数据比例。“训练验证比例”会影响训练模板的性能。

## 5.11 在 ModelArts 中物体检测标注时能否自定义标签？

可以通过修改数据集给标签添加自定义属性来设置一些自定义的属性。

图 5-4 修改数据集

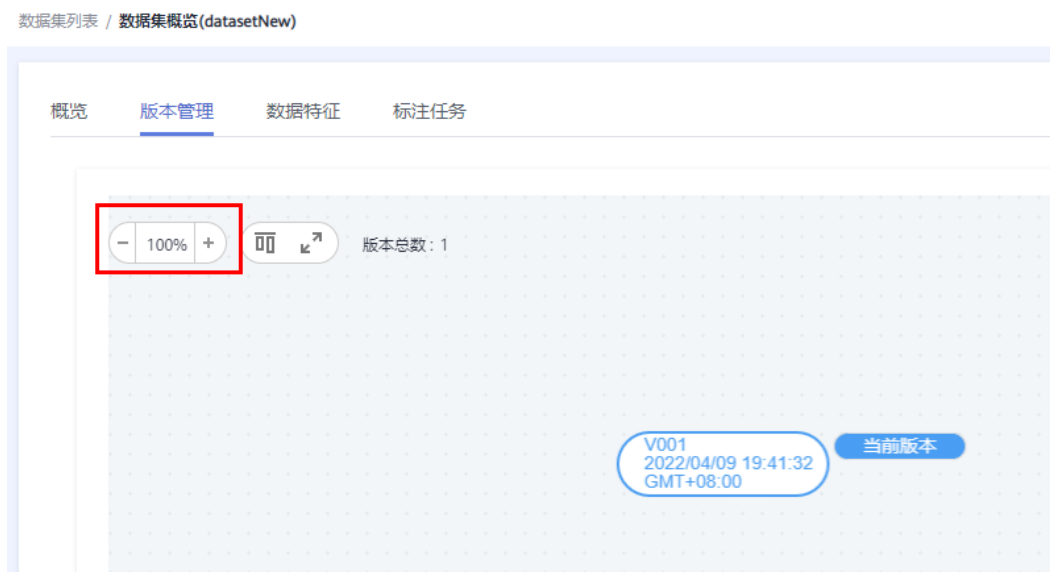
### 修改数据集

The screenshot shows the 'Modify Dataset' interface. It includes a '名称' (Name) field with the value 'autd', a '描述' (Description) field, and a '标签集' (Label Set) section. The 'Label Set' section displays two labels: 'blue' and 'none'. Each label has a color selection dropdown (currently set to blue) and a '+' icon. A tooltip '添加标签属性' (Add Label Attribute) is visible over the '+' icon.

## 5.12 ModelArts 数据集新建的版本找不到怎么办？

版本列表是可以缩放的，请缩小页面后查找。

单击数据集名称，进入数据集概览页，在概览页选择“版本管理”，可对页面进行缩小。



## 5.13 如何切分 ModelArts 数据集？

在发布数据集时，仅“图像分类”、“物体检测”、“文本分类”和“声音分类”类型数据集支持进行数据切分功能。


一般默认不启用该功能。启用后，需设置对应的训练验证比例。

输入“训练集比例”，数值只能是0~1区间内的数。设置好“训练集比例”后，“验证集比例”自动填充。“训练集比例”加“验证集比例”等于1。

“训练集比例”即用于训练模型的样本数据比例；“验证集比例”即用于验证模型的样本数据比例。“训练验证比例”会影响训练模板的性能。

## 5.14 如何删除 ModelArts 数据集中的图片？

1. 登录ModelArts管理控制台，左侧菜单栏选择“数据管理>数据标注”，进入数据标注列表，单击需要删除图片的数据集，进入标注详情页。
2. 在“全部”、“未标注”或“已标注”页面中，依次选中需要删除的图片，或者

“选择当前页”选中该页面所有图片，然后单击  删除。在弹出的对话框中，根据实际情况选择是否勾选“同时删除OBS源文件”，确认信息无误后，单击“确定”完成图片删除操作。

其中，被选中的图片，其左上角将显示为勾选状态。如果当前页面无选中图片


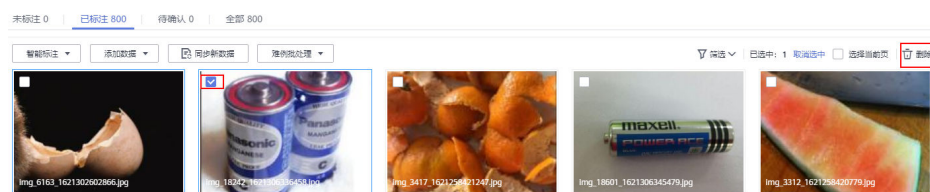
时， 按钮为灰色，无法执行删除操作。

图 5-5 删除数据集图片



# 6 Standard Notebook

---

## 6.1 ModelArts 的 Notebook 是否支持 Keras 引擎?

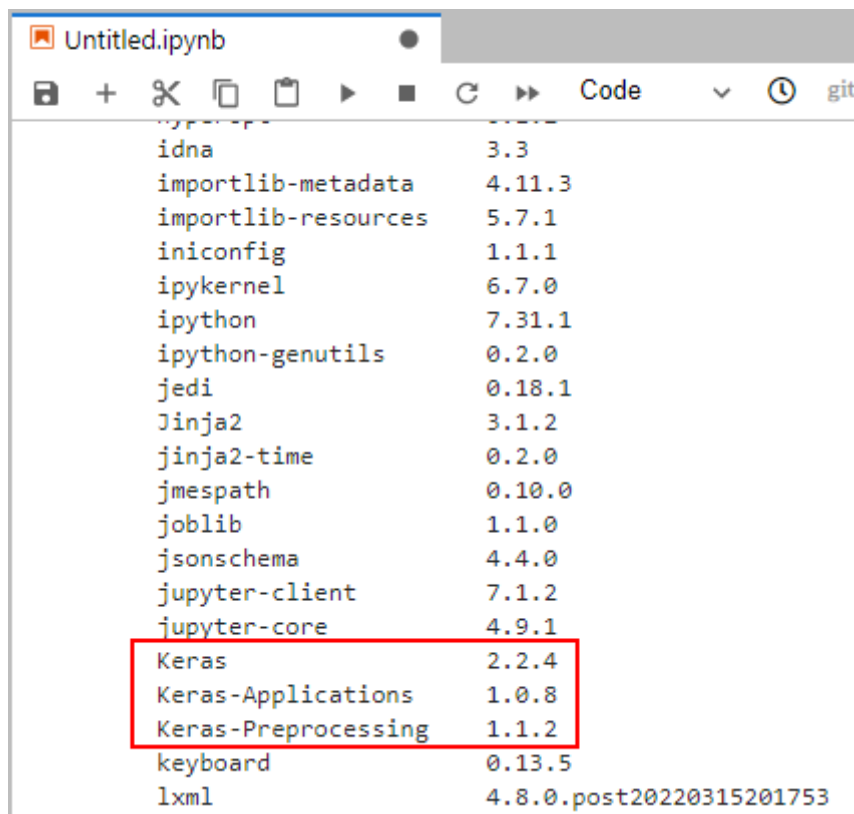
开发环境中的Notebook支持。训练作业和模型部署（即推理）暂时不支持。

Keras是一个用Python编写的高级神经网络API，它能够以TensorFlow、CNTK或者Theano作为后端运行。Notebook开发环境支持“tf.keras”。

### 如何查看 Keras 版本

1. 在ModelArts管理控制台，创建一个Notebook实例，镜像选择“TensorFlow-1.13”或“TensorFlow-1.15”。
2. 打开Notebook，在JupyterLab中执行**!pip list**查看Keras的版本。

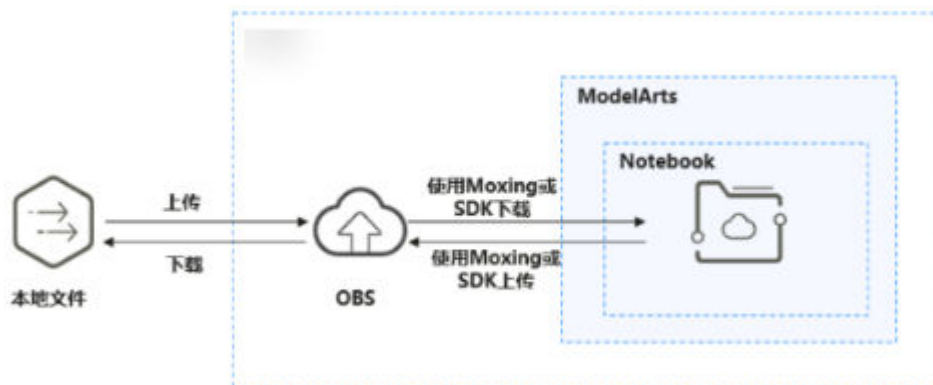
图 6-1 查看 Keras 引擎版本



## 6.2 如何在 ModelArts 的 Notebook 中上传下载 OBS 文件?

在 Notebook 中可以通过调用 ModelArts 的 Moxing 接口或者 SDK 接口与 OBS 交互，将 Notebook 中的文件上传至 OBS，或者下载 OBS 中的文件至 Notebook 中。

图 6-2 Notebook 中上传下载 OBS 文件



使用 OBS 客户端上传文件的操作指导：[上传文件](#)

## 方法一：在 Notebook 中通过 Moxing 上传下载 OBS 文件

MoXing是ModelArts自研的分布式训练加速框架，构建于开源的深度学习引擎TensorFlow、PyTorch等之上，使用MoXing API可让模型代码的编写更加简单、高效。

MoXing提供了一套文件对象API，可以用来读写OBS文件。

您可以通过MoXing API文档了解其与原生API对应关系，以及详细的接口调用示例，详细说明请参见[MoXing文件操作](#)。

示例代码：

```
import moxing as mox

# 下载一个OBS文件夹sub_dir_0，从OBS下载至Notebook
mox.file.copy_parallel('obs://bucket_name/sub_dir_0', '/home/ma-user/work/sub_dir_0')
# 下载一个OBS文件obs_file.txt，从OBS下载至Notebook
mox.file.copy('obs://bucket_name/obs_file.txt', '/home/ma-user/work/obs_file.txt')

# 上传一个OBS文件夹sub_dir_0，从Notebook上传至OBS
mox.file.copy_parallel('/home/ma-user/work/sub_dir_0', 'obs://bucket_name/sub_dir_0')
# 上传一个OBS文件obs_file.txt，从Notebook上传至OBS
mox.file.copy('/home/ma-user/work/obs_file.txt', 'obs://bucket_name/obs_file.txt')
```

## 方法二：在 Notebook 中通过 SDK 上传下载 OBS 文件

使用ModelArts SDK接口将OBS中的文件下载到Notebook后进行操作。

示例代码：将OBS中的文件file1.txt下载到Notebook的/home/ma-user/work/路径下。其中，桶名称、文件夹和文件的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.download_file(src_obs_file="obs://bucket-name/dir1/file1.txt", dst_local_dir="/home/ma-user/work/")
```

使用ModelArts SDK接口将OBS中的文件夹下载到Notebook后进行操作。

示例代码：将OBS中的文件夹dir1下载到Notebook的/home/ma-user/work/路径下。其中，桶名称和文件夹的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.download_dir(src_obs_dir="obs://bucket-name/dir1/", dst_local_dir="/home/ma-user/work/")
```

使用ModelArts SDK接口将Notebook中的文件上传到OBS后进行操作。

示例代码：将Notebook中的file1.txt文件上传到OBS桶路径obs://bucket-name/dir1/中。其中，桶名称、文件夹和文件的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.upload_file(src_local_file="/home/ma-user/work/file1.txt", dst_obs_dir='obs://bucket-name/dir1/')
```

使用ModelArts SDK接口将Notebook中的文件夹上传到OBS。

示例代码：将Notebook中的文件夹“/work/”上传至“bucket-name”桶的“dir1”文件夹下，路径为“obs://bucket-name/dir1/work/”。其中，桶名称和文件夹的名称均可以按照业务需求自定义。

```
from modelarts.session import Session
session = Session()
session.obs.upload_dir(src_local_dir='/home/ma-user/work/', dst_obs_dir='obs://bucket-name/dir1/')
```

## 异常处理

通过OBS下载文件到Notebook中时，提示Permission denied。请依次排查：

- 请确保读取的OBS桶和Notebook处于同一站点区域，例如：都在华北-北京四站点。不支持跨站点访问OBS桶。具体请参见[查看OBS桶与ModelArts是否在同一区域](#)。
- 请确认操作Notebook的账号有权限读取OBS桶中的数据。如没有权限，请参见在[ModelArts的Notebook中，如何访问其他账号的OBS桶？](#)。

## 6.3 ModelArts 的 Notebook 实例 upload 后，数据会上传到哪里？

针对这个问题，有两种情况：

- 如果您创建的Notebook使用OBS存储实例时  
单击“upload”后，数据将直接上传到该Notebook实例对应的OBS路径下，即创建Notebook时指定的OBS路径。
- 如果您创建的Notebook使用EVS存储实例时  
单击“upload”后，数据将直接上传至当前实例容器中，即在“Terminal”中的“~/work”目录下。

## 6.4 在 ModelArts 中如何将 Notebook A 的数据复制到 Notebook B 中？

目前不支持直接将Notebook A的数据复制到Notebook B，如果需要复制数据，可参考如下步骤操作：

1. 将Notebook A的数据上传至OBS；
2. 下载OBS中的数据至Notebook B。

文件的上传下载详细操作请参考[如何在ModelArts的Notebook中上传下载OBS文件？](#)。

## 6.5 在 ModelArts 的 Notebook 中如何对 OBS 的文件重命名？

由于OBS管理控制台不支持对OBS的文件重命名，当您需要对OBS文件进行重命名时需要通过调用MoXing API实现，在已有的或者新创建的Notebook中，执行如下命令，通过接口对OBS中的文件进行重命名。

具体操作如下：

如下示例为将文件“obs\_file.txt”重命名为“obs\_file\_2.txt”。

```
import moxing as mox
mox.file.rename('obs://bucket_name/obs_file.txt', 'obs://bucket_name/obs_file_2.txt')
```

## 6.6 在 ModelArts 的 Notebook 中如何使用 pandas 库处理 OBS 桶中的数据？

**步骤1** 参考[下载OBS文件到Notebook中](#)的指导，将OBS中的数据下载至Notebook本地处理。

**步骤2** 参考[pandas用户指南](#)处理pandas数据。

----结束

## 6.7 在 ModelArts 的 Notebook 中，如何访问其他账号的 OBS 桶？

创建Notebook时选择OBS存储，这种情况下只能访问到自己账号下的桶，无法访问到其他账号的OBS桶。

如果需要在Notebook中，访问其他账号的OBS文件，前提是，需获取目标OBS桶的读写权限。

1. 首先，请联系OBS桶的创建者，参考[对其他账号授予桶的读写权限](#)指导，授予当前账号OBS桶的读写权限。此操作指导是某一华为云账号将其OBS桶权限授予其他华为云账号。如果您的账号是IAM用户或其他场景时，请参见《[OBS权限配置指南](#)》> 典型场景配置案例，查找授予OBS桶权限的指导。
2. 获得OBS桶的读写权限后，您可以在Notebook中，使用moxing接口，访问对应的OBS桶，并读取数据。举例如下：

```
import moxing as mox
mox.file.copy_parallel('obs://bucket_1/dataset', 'obs://bucket_2/dataset')
```

其中，“bucket\_1”为其他账号的OBS桶，“bucket\_2”为自己的OBS桶。

## 6.8 在 ModelArts 的 Notebook 中 JupyterLab 默认工作路径是什么？

- **带OBS存储的Notebook实例**

JupyterLab文件默认存储路径，为创建Notebook时指定的OBS路径。

在文件列表的所有文件读写操作都是基于所选择的OBS路径下的内容操作的，跟当前实例空间没有关系。如果用户需要将内容同步到实例空间，需使用[JupyterLab上传下载功能](#)。

- **带EVS存储的Notebook实例**

JupyterLab文件默认存储路径，为创建Notebook实例时，系统自动分配的EVS空间。

在文件列表的所有文件读写操作都是基于所选择的EVS下的内容操作的。使用EVS类型的挂载，可将大数据挂载至“~/work”目录下。

## 6.9 如何查看 ModelArts 的 Notebook 使用的 cuda 版本？

执行如下命令查看环境中的cuda版本。



```
ll /usr/local | grep cuda
```

举例：

图 6-3 查看当前环境的 cuda 版本

```
ll /usr/local | grep cuda
lrwxrwxrwx  1 root          9 Feb  9 09:28 cuda -> cuda-10.2/
drwxr-xr-x 12 root       4096 Feb 10 09:28 cuda-10.2/
```

如图1所示，当前环境中cuda版本为10.2

## 6.10 在 ModelArts 的 Notebook 中如何获取本机外网 IP?

本机的外网IP地址可以在主流搜索引擎中搜索“IP地址查询”获取。

图 6-4 查询外网 IP 地址

### IP地址查询



## 6.11 ModelArts 的 Notebook 有代理吗？如何关闭？

Notebook有代理。

执行`env|grep proxy`命令查询Notebook代理。

执行`unset https_proxy unset http_proxy`命令关闭代理。

## 6.12 在 ModelArts 的 Notebook 中内置引擎不满足使用时，如何自定义引擎 IPython Kernel?

### 使用场景

当前Notebook默认内置的引擎环境不能满足用户诉求，用户可以新建一个conda env 按需搭建自己的环境。本小节以搭建一个“python3.6.5和tensorflow1.2.0”的IPython Kernel为例进行展示。

### 操作步骤

1. 创建conda env。

在Notebook的Terminal中执行如下命令。其中，my-env是虚拟环境名称，用户可自定义。conda详细参数可参考[conda官网](#)。

```
conda create --quiet --yes -n my-env python=3.6.5
```

创建完成后，执行**conda info --envs**命令查看现有的虚拟环境列表，可以看到my-env虚拟环境：

```
sh-4.4$conda info --envs
# conda environments:
#
base                * /home/ma-user/anaconda3
TensorFlow-2.1      /home/ma-user/anaconda3/envs/TensorFlow-2.1
my-env              /home/ma-user/anaconda3/envs/my-env
python-3.7.10       /home/ma-user/anaconda3/envs/python-3.7.10      /opt/conda/envs/my-env
```

2. 执行如下命令进入conda env。

```
source /home/ma-user/anaconda3/bin/activate /home/ma-user/anaconda3/envs/my-env
```

3. 执行如下命令在my env里安装如下依赖包。

```
pip install ipykernel
```

如果遇到版本冲突，建议固定版本如下：

```
pip install jupyter_core==5.3.0
pip install jupyter_client==8.2.0
pip install ipython==8.10.0
pip install ipykernel==6.23.1
```

4. 执行下述命令添加虚拟环境为IPython Kernel。

其中--name的值可自定义。

```
python3 -m ipykernel install --user --name "my-py3-tensorflow-env"
```

执行完毕后，可以看到下述提示信息。

```
(my-env) sh-4.4$python3 -m ipykernel install --user --name "my-py3-tensorflow-env"
Installed kernelspec my-py3-tensorflow-env in /home/ma-user/.local/share/jupyter/kernels/my-py3-tensorflow-env
```

5. 自定义虚拟环境Kernel的环境变量。

执行**cat /home/ma-user/.local/share/jupyter/kernels/my-py3-tensorflow-env/kernel.json**，可以看到默认配置如下：

```
{
  "argv": [
    "/home/ma-user/anaconda3/envs/my-env/bin/python3",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "my-py3-tensorflow-env",
  "language": "python"
}
```

按需添加env字段的值，可参考下述配置。其中，PATH中增加了该虚拟环境python包所在路径：

```
{
  "argv": [
    "/home/ma-user/anaconda3/envs/my-env/bin/python3",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "my-py3-tensorflow-env",
  "language": "python",
  "env": {
    "PATH": "/home/ma-user/anaconda3/envs/my-env/bin:/opt/conda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/ma-user/modelarts/ma-cli/bin",

```

```

"http_proxy": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
"https_proxy": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
"ftp_proxy": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
"HTTP_PROXY": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
"HTTPS_PROXY": "http://proxy-notebook.modelarts-dev-proxy.com:8083",
"FTP_PROXY": "http://proxy-notebook.modelarts-dev-proxy.com:8083"
}
}

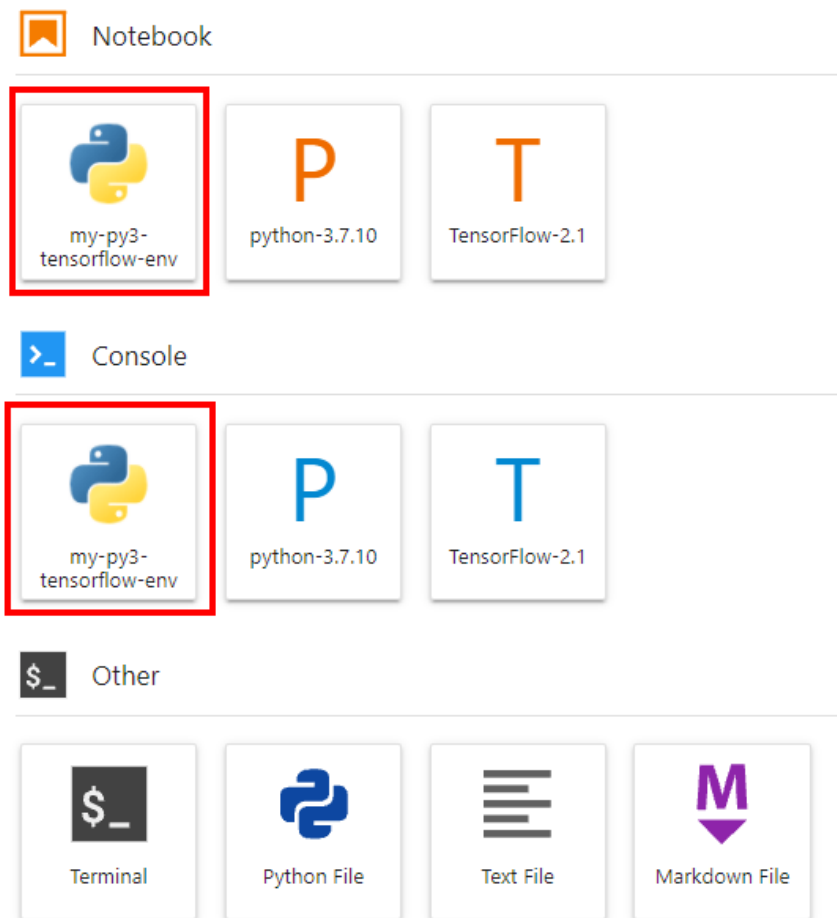
```

6. 手动删除无效图标。

```
rm -r /home/ma-user/.local/share/jupyter/kernels/my-py3-tensorflow-env/logo-*
```

7. 进入虚拟环境的IPython Kernel。

刷新JupyterLab页面，可以看到自定义的虚拟环境Kernel。如下所示：



单击my-py3-tensorflow-env图标，验证是否为当前环境，如下所示：

```

Untitled1.ipynb 2 vCPU + 4 GiB my-py3-tensorflow-env
[1]: !which python
/home/ma-user/anaconda3/envs/my-env/bin/python

[2]: !python --version
Python 3.6.5 :: Anaconda, Inc.

[3]: !pip show tensorflow
Name: tensorflow
Version: 1.2.0
Summary: TensorFlow helps the tensors flow
Home-page: http://tensorflow.org/
Author: Google Inc.
Author-email: opensource@google.com
License: Apache 2.0
Location: /home/ma-user/anaconda3/envs/my-env/lib/python3.6/site-packages
Requires: h5mlib, protobuf, markdown, werkzeug, bleach, backports.weakref, wheel, numpy, six
Required-by:

```

8. 清理环境。

删除虚拟环境的IPython Kernel。

```
jupyter kernelspec uninstall my-py3-tensorflow-env  
删除虚拟环境。  
conda env remove -n my-env
```

## 6.13 在 ModelArts 的 Notebook 中如何将 git clone 的 py 文件变为 ipynb 文件?

### 问题描述

在 ModelArts 的 Notebook 中如何将 git clone 的 py 文件变为 ipynb 文件?

### 处理方法

在 ipynb 文件中，执行 `%load XXX.py` 命令，即可将 py 文件内容加载到 ipynb 中。

以 “test.py” 文件为例，下图展示了如何将 “test.py” 的文件内容加载到 ipynb 文件中。

图 6-5 test.py 文件

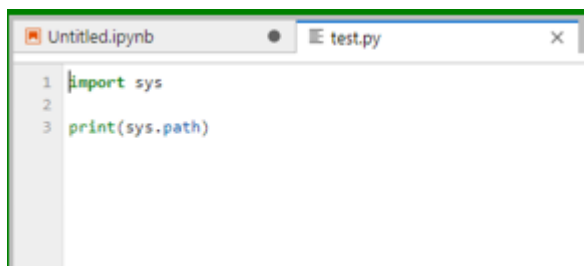


图 6-6 将 “test.py” 文件内容加载到 ipynb 文件里

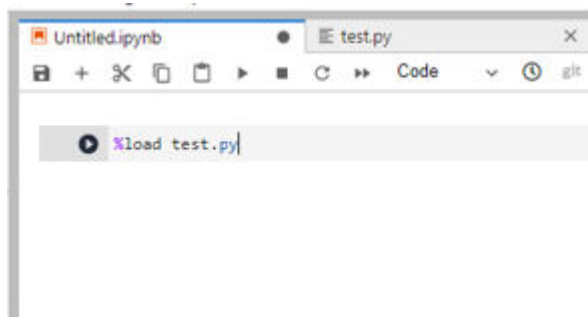
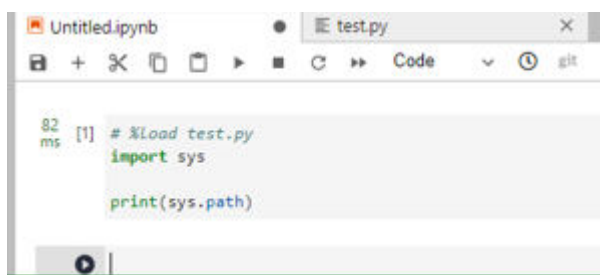


图 6-7 加载后的 ipynb 文件



## 6.14 在 ModelArts 的 Notebook 实例重启时，数据集会丢失吗？

ModelArts提供的Notebook实例是以ma-user启动的，用户进入实例后，工作目录默认是“/home/ma-user/work”。

创建实例，“/home/ma-user/work”目录下挂载的数据，在实例停止、重新启动后依然保留，其他目录下的内容会还原。

## 6.15 在 ModelArts 的 Notebook 的 Jupyterlab 可以安装插件吗？

Jupyter可以安装插件。

目前jupyter插件多数采用wheel包的形式发布，一次性完成前后端插件的安装，安装时注意使用jupyter服务依赖的环境“/modelarts/authoring/notebook-conda/bin/pip”进行安装，不要使用默认的anaconda（kernel依赖的python环境）的pip进行安装。



```
Terminal 1
ModelArts
Using user ma-user
Ubuntu 16.04.3 LTS
Tips:
1) Navigate to the target conda environment. For details, see /home/ma-user/README.
2) Copy (Ctrl+C) and paste (Ctrl+V) on the jupyter terminal.
3) Store your data in /home/ma-user/work, to which a persistent volume is mounted.
[ma-user work]$which pip
/modelarts/authoring/notebook-conda/bin/pip
[ma-user work]$ /modelarts/authoring/notebook-conda/bin/pip install jupyterlab-github
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: http://repo.myhuaweicloud.com/repository/pypi/simple
Collecting jupyterlab-github
  Downloading http://repo.myhuaweicloud.com/repository/pypi/packages/87/e0/b5b08472f4a9aaf69d3dbc69ef45c50a9dc7172c5538f16d4fec77c5f4/jupyterlab_github-3.0.1-py3-none-any.whl (113 kB)
    113.5/113.5 kB 14.6 MB/s eta 0:00:00
Requirement already satisfied: jupyterlab==3.0 in /modelarts/authoring/notebook-conda/lib/python3.7/site-packages (from jupyterlab-github) (3.2.3)
```

使用命令 `jupyter labextension list --app-dir=/home/ma-user/.lab/console` 查询

前端插件安装目录为： `/home/ma-user/.local/share/jupyter/labextensions`

```
[ma-user work]$jupyter labextension list --app-dir=/home/ma-user/.lab/console
JupyterLab v3.2.3
/home/ma-user/.local/share/jupyter/labextensions
  @jupyterlab/github v3.0.1 enabled OK (python, jupyterlab_github)

/modelarts/authoring/notebook-conda/share/jupyter/labextensions
  jupyter-matplotlib v0.10.5 enabled OK
  jupyterlab-plotly v5.6.0 enabled OK
  jupyterlab-pygments v0.2.2 enabled OK (python, jupyterlab_pygments)
  nbdime-jupyterlab v2.1.1 enabled OK
  @jupyter-widgets/jupyterlab-manager v3.1.1 enabled OK (python, jupyterlab_widgets)
```

后端插件代码安装目录： `/home/ma-user/.local/lib/python3.7/site-packages`

```
(PyTorch-1.8) [ma-user site-packages]$pwd
/home/ma-user/.local/lib/python3.7/site-packages
(PyTorch-1.8) [ma-user site-packages]$tree
.
├── jupyterlab_github
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-37.pyc
│   │   └── _version.cpython-37.pyc
│   ├── _version.py
│   └── labextension
│       ├── package.json
│       ├── schemas
│       │   └── @jupyterlab
│       │       └── github
│       │           ├── drive.json
│       │           └── package.json.orig
│       └── static
│           ├── 060d51417beb0f47daa10a4dcb2e147d.png
│           ├── 534.06ec9cb816bf9170af66.js
│           ├── 742.c0343f89d61252c41791.js
│           ├── 784.7af3b56872a0f8721f0b.js
│           ├── remoteEntry.e0263a028853908ae4bb.js
│           └── style.js
└── jupyterlab_github-3.0.1.dist-info
    ├── INSTALLER
    ├── LICENSE
    ├── METADATA
    ├── RECORD
    ├── REQUESTED
    ├── WHEEL
    └── top_level.txt

8 directories, 20 files
```

配置文件目录：/home/ma-user/.jupyter/

```
(PyTorch-1.8) [ma-user jupyter_server_config.d]$pwd
/home/ma-user/.jupyter/jupyter_server_config.d
(PyTorch-1.8) [ma-user jupyter_server_config.d]$tree
.
├── jupyter_server_mathjax.json
├── jupyter_tensorboard.json
├── jupyterlab.json
├── jupyterlab_git.json
├── jupyterlab_github.json
├── modelarts_notebook_plugin.json
├── nbclassic.json
├── nbdime.json
└── notebook_shim.json

0 directories, 9 files
```

后端插件使用 `jupyter server extension list` 命令查询。

```
[ma-user work]$jupyter server extension list
Config dir: /home/ma-user/.jupyter
jupyter_server_mathjax enabled
- Validating jupyter_server_mathjax...
  jupyter_server_mathjax OK
jupyter_tensorboard enabled
- Validating jupyter_tensorboard...
  jupyter_tensorboard 0.2.0 OK
jupyterlab enabled
- Validating jupyterlab...
  jupyterlab 3.2.3 OK
jupyterlab git enabled
- Validating jupyterlab_git...
  jupyterlab git 0.34.0 OK
modelarts_notebook_plugin enabled
- Validating modelarts_notebook_plugin...
  modelarts_notebook_plugin OK
nbclassic enabled
- Validating nbclassic...
  nbclassic OK
nbdime enabled
- Validating nbdime...
  nbdime 3.1.1 OK
notebook_shim enabled
- Validating notebook_shim...
  notebook shim OK
```

## 6.16 在 ModelArts 的 Notebook 的 CodeLab 中能否使用昇腾卡进行训练？

有两种情况。

第一种，在ModelArts控制台的“总览”界面打开CodeLab，使用的是CPU或GPU资源，无法使用昇腾卡训练。

第二种，如果是AI Gallery社区的Notebook案例，使用的资源是ASCEND的，“Run in ModelArts”跳转到CodeLab，就可以使用昇腾卡进行训练。

#### 当前运行环境：

CPU: 24核

内存: 96GB

#### ASCEND:

架构: Ascend (16GB) \* 1

规格: aarch64

价格: modelarts.kat1.xlarge.free

限时免费

切换规格

也支持切换规格

#### 选择运行环境

[付费]Ascend: 1\*Ascend | CPU: 24核 96GB

¥19.5//小时

[付费]Ascend: 8\*Ascend CPU: 192核 768GB

¥155.98//小时

[售罄][付费]Ascend: 2\*Ascend | CPU: 48核 192GB

¥39//小时

切换规格

取消

## 6.17 如何在 ModelArts 的 Notebook 的 CodeLab 上安装依赖?

ModelArts CodeLab中已安装Jupyter、Python程序包等多种环境，您也可以使用pip install在Notebook或Terminal中安装依赖包。

### 在 Notebook 中安装

1. 在总览页面进入CodeLab。
2. 在“Notebook”区域下，新建一个ipynb文件。
3. 在新建的Notebook中，在代码输入栏输入如下命令。



```
!pip install xxx
```

## 在 Terminal 中安装

在Terminal里激活需要的anaconda python环境后再进行安装。

例如，通过terminal在“TensorFlow-1.8”的环境中使用**pip**安装Shapely。

1. 在总览页面进入CodeLab。
2. 在“Other”区域下，选择“Terminal”，新建一个terminal文件。
3. 在代码输入栏输入以下命令，获取当前环境的kernel，并激活需要安装依赖的python环境。

```
cat /home/ma-user/README
```

```
source /home/ma-user/anaconda3/bin/activate TensorFlow-1.8
```

### 说明

如果需要在其他python环境里安装，请将命令中“TensorFlow-1.8”替换为其他引擎。

4. 在代码输入栏输入以下命令安装Shapely。

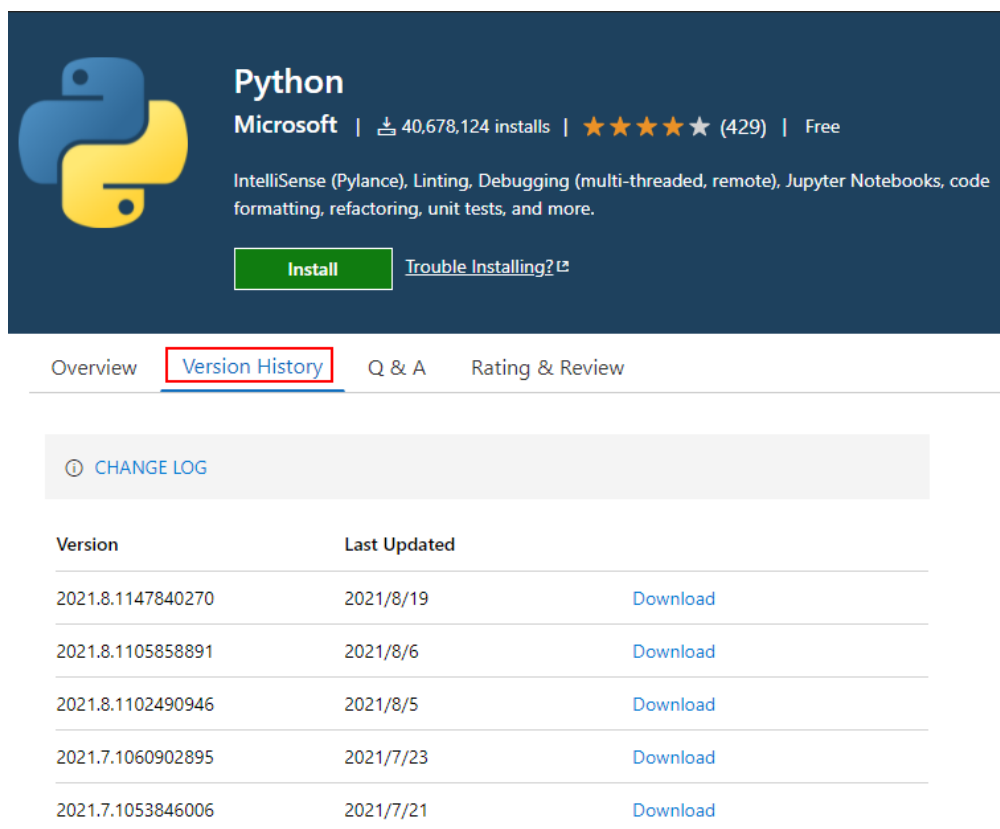
```
pip install Shapely
```

## 6.18 在 ModelArts 的 Notebook 中安装远端插件时不稳定要怎么办？

方法一：离线包安装方式（推荐）

1. 到VS Code插件官网vscode\_marketplace搜索待安装的Python插件，[Python插件路径](#)。
2. 单击进入Python插件的Version History页签后，下载该插件的离线安装包，如图所示。

图 6-8 Python 插件离线安装包



3. 在本地VS Code环境中，将下载好的.vsix文件拖动到远端Notebook中。
4. 右键单击该文件，选择Install Extension VSIX。

#### 方法二：设置远端默认安装的插件

按照[在ModelArts的Notebook中如何设置VS Code远端默认安装的插件?](#)配置，即会在连接远端时自动安装，减少等待时间。

方法三：VS Code官网排查方式<https://code.visualstudio.com/docs/remote/troubleshooting>

小技巧（按需调整远端连接的相关参数）：

```
"remote.SSH.connectTimeout": 10,
"remote.SSH.maxReconnectionAttempts": null,
"remote.downloadExtensionsLocally": true,
"remote.SSH.useLocalServer": false,
"remote.SSH.localServerDownload": "always",
```

## 6.19 在 ModelArts 的 Notebook 中实例重新启动后要怎么连接？

可以在本地的ssh config文件中对这个Notebook配置参数“StrictHostKeyChecking no”和“UserKnownHostsFile=/dev/null”，如下参考所示：

```
Host roma-local-cpu
  HostName x.x.x.x #IP地址
  Port 22522
  User ma-user
```

```
IdentityFile C:/Users/my.pem
StrictHostKeyChecking no
ForwardAgent yes
```

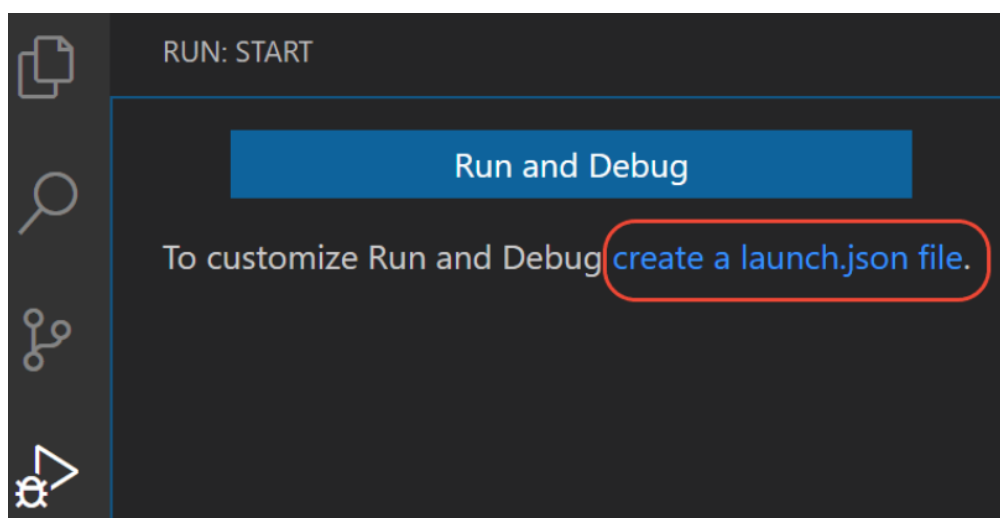
提示：因为SSH登录时会忽略known\_hosts文件，有安全风险

## 6.20 在 ModelArts 的 Notebook 中使用 VS Code 调试代码无法进入源码怎么办？

如果已有launch.json文件，请直接看步骤三。

### 步骤一：打开launch.json文件

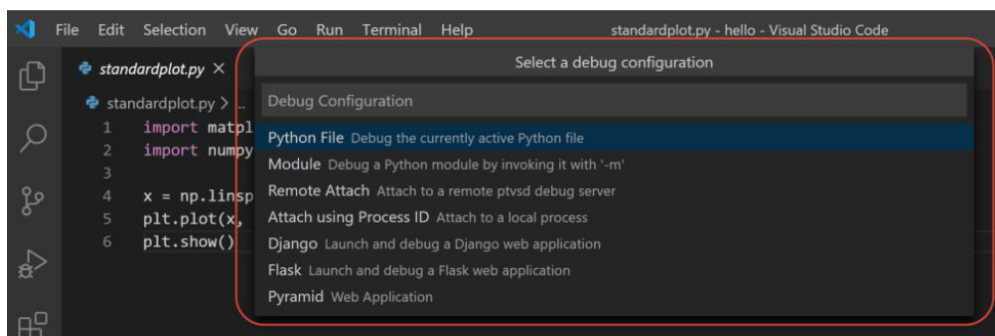
- 方法一：单击左侧菜单栏的Run (Ctrl+Shift+D) 按钮，再单击create a launch.json file。如下图所示：



- 方法二：单击上侧菜单栏中的Run > Open configurations按钮

### 步骤二：选择语言

如果需要对Python语言进行设置，在弹出的Select a debug configuration中选择Python File，其他语言操作类似。如下图所示：



步骤三：编辑launch.json，增加justMyCode": false配置，如下所示。

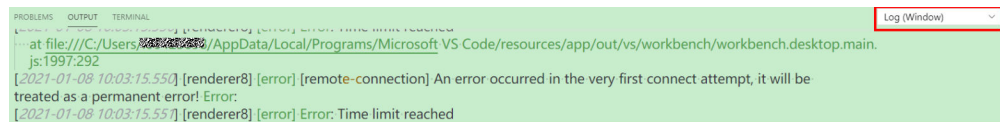
```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Python: 当前文件",
      "type": "python",
```

```
"request": "launch",  
"program": "${file}",  
"console": "integratedTerminal",  
"justMyCode": false  
  }  
]  
}
```

## 6.21 在 ModelArts 的 Notebook 中使用 VS Code 如何查看远端日志?

1. 在VS Code环境中执行Ctrl+Shift+P
2. 搜show logs
3. 选择Remote Server。

也可在如下截图的红框处切换至其他的Log



## 6.22 在 ModelArts 的 Notebook 中如何打开 VS Code 的配置文件 settings.json?

1. 在VS Code环境中执行Ctrl+Shift+P
2. 搜Open User Settings (JSON)

## 6.23 在 ModelArts 的 Notebook 中如何设置 VS Code 背景色为豆沙绿?

在VS Code的配置文件settings.json中添加如下参数

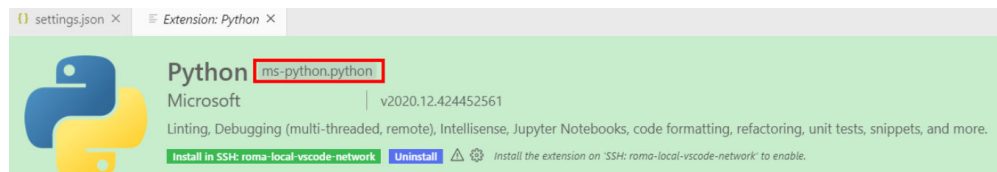
```
"workbench.colorTheme": "Atom One Light",  
"workbench.colorCustomizations": {  
  "[Atom One Light]": {  
    "editor.background": "#C7EDCC",  
    "sideBar.background": "#e7f0e7",  
    "activityBar.background": "#C7EDCC",  
  },  
},
```

## 6.24 在 ModelArts 的 Notebook 中如何设置 VS Code 远端默认安装的插件?

在VS Code的配置文件settings.json中添加remote.SSH.defaultExtensions参数，如自动安装Python和Maven插件，可配置如下。

```
"remote.SSH.defaultExtensions": [  
  "ms-python.python",  
  "vscjava.vscode-maven"  
],
```

其中，插件名称可以单击VS Code的某个插件后获取，如下所示。



## 6.25 在 ModelArts 的 VS Code 中如何把本地插件安装到远端或把远端插件安装到本地？

1. 在VS Code的环境中执行Ctrl+Shift+P
2. 搜install local，按需选择即可

## 6.26 在 ModelArts 的 Notebook 中，如何使用昇腾多卡进行调试？

昇腾多卡训练任务是多进程多卡模式，跑几卡需要起几个python进程。昇腾底层会读取环境变量：RANK\_TABLE\_FILE，开发环境已经设置，用户无需关注。比如跑八卡，可以如下片段代码：

```
export RANK_SIZE=8
current_exec_path=$(pwd)
echo 'start training'
for((i=0;i<=$RANK_SIZE-1;i++));
do
echo 'start rank '$i
mkdir ${current_exec_path}/device$i
cd ${current_exec_path}/device$i
echo $i
export RANK_ID=$i
dev=`expr $i + 0`
echo $dev
export DEVICE_ID=$dev
python train.py > train.log 2>&1 &
done
```

其中，train.py中设置环境变量DEVICE\_ID：

```
devid = int(os.getenv('DEVICE_ID'))
context.set_context(mode=context.GRAPH_MODE, device_target="Ascend", device_id=devid)
```

## 6.27 在 ModelArts 的 Notebook 中使用不同的资源规格训练时为什么训练速度差不多？

如果用户的代码中训练任务是单进程的，使用Notebook 8核64GB，72核512GB训练的速度是基本一致的，例如用户用的是2核4GB的资源，使用4核8GB，或者8核64GB效果是一样的。

如果用户的代码中训练任务是多进程的，使用Notebook 72核512GB训练速度要优于8核64GB。

## 6.28 在 ModelArts 的 Notebook 中使用 MoXing 时，如何进行增量训练？

在使用 MoXing 构建模型时，如果您对前一次训练结果不满意，可以在更改部分数据和标注信息后，进行增量训练。

### “mox.run”添加增量训练参数

在完成标注数据或数据集的修改后，您可以在“mox.run”中，修改“log\_dir”参数，并新增“checkpoint\_path”参数。其中“log\_dir”参数建议设置为一个新的目录，“checkpoint\_path”参数设置为上一次训练结果输出路径，如果是 OBS 目录，路径填写时建议使用“obs://”开头。

如果标注数据中的标签发生了变化，在运行“mox.run”前先执行[如果标签发生变化的操作](#)。

```
mox.run(input_fn=input_fn,
        model_fn=model_fn,
        optimizer_fn=optimizer_fn,
        run_mode=flags.run_mode,
        inter_mode=mox.ModeKeys.EVAL if use_eval_data else None,
        log_dir=log_dir,
        batch_size=batch_size_per_device,
        auto_batch=False,
        max_number_of_steps=max_number_of_steps,
        log_every_n_steps=flags.log_every_n_steps,
        save_summary_steps=save_summary_steps,
        save_model_secs=save_model_secs,
        checkpoint_path=flags.checkpoint_url,
        export_model=mox.ExportKeys.TF_SERVING)
```

### 如果标签发生变化

当数据集中的标签发生变化时，需要执行如下语句。此语句需在“mox.run”之前运行。

语句中的“logits”，表示根据不同网络中分类层权重的变量名，配置不同的参数。此处填写其对应的关键字。

```
mox.set_flag('checkpoint_exclude_patterns', 'logits')
```

如果使用的是 MoXing 内置网络，其对应的关键字需使用如下 API 获取。此示例将打印 Resnet\_v1\_50 的关键字，为“logits”。

```
import moxing.tensorflow as mox

model_meta = mox.get_model_meta(mox.NetworkKeys.RESNET_V1_50)
logits_pattern = model_meta.default_logits_pattern
print(logits_pattern)
```

您也可以通过如下接口，获取 MoXing 支持的网络名称列表。

```
import moxing.tensorflow as mox
print(help(mox.NetworkKeys))
```

打印出来的示例如下所示：

```
Help on class NetworkKeys in module
moxing.tensorflow.nets.nets_factory:
```

```
class NetworkKeys(builtins.object)
| Data descriptors defined here:
|
| _dict_
|     dictionary for instance variables (if defined)
|
| _weakref_
|     list of weak references to the object (if defined)
|
|-----
| Data and other attributes defined here:
|
| ALEXNET_V2 = 'alexnet_v2'
|
| CIFARNET = 'cifarnet'
|
| INCEPTION_RESNET_V2 = 'inception_resnet_v2'
|
| INCEPTION_V1 = 'inception_v1'
|
| INCEPTION_V2 = 'inception_v2'
|
| INCEPTION_V3 = 'inception_v3'
|
| INCEPTION_V4 = 'inception_v4'
|
| LENET = 'lenet'
|
| MOBILENET_V1 = 'mobilenet_v1'
|
| MOBILENET_V1_025 = 'mobilenet_v1_025'
|
| MOBILENET_V1_050 = 'mobilenet_v1_050'
|
| MOBILENET_V1_075 = 'mobilenet_v1_075'
|
| MOBILENET_V2 = 'mobilenet_v2'
|
| MOBILENET_V2_035 = 'mobilenet_v2_035'
|
| MOBILENET_V2_140 = 'mobilenet_v2_140'
|
| NASNET_CIFAR = 'nasnet_cifar'
|
| NASNET_LARGE = 'nasnet_large'
|
| NASNET_MOBILE = 'nasnet_mobile'
|
| OVERFEAT = 'overfeat'
|
| PNASNET_LARGE = 'pnasnet_large'
|
| PNASNET_MOBILE = 'pnasnet_mobile'
|
| PVANET = 'pvanet'
|
| RESNET_V1_101 = 'resnet_v1_101'
|
| RESNET_V1_110 = 'resnet_v1_110'
|
| RESNET_V1_152 = 'resnet_v1_152'
|
| RESNET_V1_18 = 'resnet_v1_18'
|
| RESNET_V1_20 = 'resnet_v1_20'
|
| RESNET_V1_200 = 'resnet_v1_200'
|
| RESNET_V1_50 = 'resnet_v1_50'
```

```
RESNET_V1_50_8K = 'resnet_v1_50_8k'  
RESNET_V1_50_MOX = 'resnet_v1_50_mox'  
RESNET_V1_50_OCT = 'resnet_v1_50_oct'  
RESNET_V2_101 = 'resnet_v2_101'  
RESNET_V2_152 = 'resnet_v2_152'  
RESNET_V2_200 = 'resnet_v2_200'  
RESNET_V2_50 = 'resnet_v2_50'  
RESNEXT_B_101 = 'resnext_b_101'  
RESNEXT_B_50 = 'resnext_b_50'  
RESNEXT_C_101 = 'resnext_c_101'  
RESNEXT_C_50 = 'resnext_c_50'  
VGG_16 = 'vgg_16'  
VGG_16_BN = 'vgg_16_bn'  
VGG_19 = 'vgg_19'  
VGG_19_BN = 'vgg_19_bn'  
VGG_A = 'vgg_a'  
VGG_A_BN = 'vgg_a_bn'  
XCEPTION_41 = 'xception_41'  
XCEPTION_65 = 'xception_65'  
XCEPTION_71 = 'xception_71'
```

## 6.29 在 ModelArts 的 Notebook 中如何查看 GPU 使用情况？

创建Notebook时，当您选择的类型为GPU时，查看GPU使用情况具体操作如下：

1. 登录ModelArts管理控制台，选择“开发空间>Notebook”。
2. 在Notebook列表中，单击目标Notebook“操作”列的“打开”，进入“Jupyter”开发页面。
3. 在Jupyter页面的“Files”页签下，单击“New”，然后选择“Terminal”，进入到Terminal界面。
4. 执行如下命令查看GPU使用情况。  
`nvidia-smi`
5. 查看当前Notebook实例中有哪些进程使用GPU。  
方法一：  
`python /modelarts/tools/gpu_processes.py`  
如果当前进程使用GPU



```

ModelArts
Using user ma-user
Ubuntu 18.04.6 LTS, CUDA-10.1
Tips:
1) Navigate to the target conda environment. For details, see /home/ma-user/README.
2) Copy (Ctrl+C) and paste (Ctrl+V) on the jupyter terminal.
3) Store your data in /home/ma-user/work, to which a persistent volume is mounted.
(PyTorch-1.4) [ma-user work]$python /modelarts/tools/gpu_processes.py
Fri Mar 17 11:27:17 2023
+-----+-----+-----+-----+
| Processes: |      |      |      |      |
| GPU        | PID  | Process name | GPU Memory Usage |
+-----+-----+-----+-----+
| 0          | 4608 | python      | 785 MiB           |
+-----+-----+-----+-----+
| 0          | 4731 | python      | 785 MiB           |
+-----+-----+-----+-----+
| 0          | 4860 | python      | 785 MiB           |
+-----+-----+-----+-----+
| 0          | 5000 | python      | 785 MiB           |
+-----+-----+-----+-----+
(PyTorch-1.4) [ma-user work]$

```

如果当前没有进程使用GPU

```

(PyTorch-1.4) [ma-user work]$python /modelarts/tools/gpu_processes.py
There is no GPU specification in the current environment, failing to get the GPU_UUIDS.
(PyTorch-1.4) [ma-user work]$

```

方法二：

打开文件 “/resource\_info/gpu\_usage.json”，可以看到有哪些进程在使用GPU。

```

{
  <notebook name>: {
    <GPU0 UUID>: [
      {
        "pid": 2263,
        "processName": "python",
        "gpuMemoryUsage": "4935Mi"
      },
      {...}
    ]
    <GPU1 UUID>: [...]
  }
}

```

如果当前没有进程使用GPU，该文件可能不存在或为空。

## 6.30 在 ModelArts 的 Notebook 中如何在代码中打印 GPU 使用信息？

用户可通过shell命令或python命令查询GPU使用信息。

## 使用 shell 命令

1. 执行nvidia-smi命令。

依赖CUDA nvcc

```
watch -n 1 nvidia-smi
```

```
Every 1.0s: nvidia-smi

Mon Oct 25 15:20:11 2021
+-----+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2    |
+-----+-----+-----+-----+-----+
| GPU   Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+
|    0   Tesla V100-SXM2...    On          | 00000000:5F:00.0 Off  |
| N/A   31C    P0     43W / 300W |  0MiB / 32510MiB |      0%    Default  |
+-----+-----+-----+-----+-----+
|    1   Tesla V100-SXM2...    On          | 00000000:B5:00.0 Off  |
| N/A   34C    P0     44W / 300W |  0MiB / 32510MiB |      0%    Default  |
+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
+-----+-----+-----+-----+-----+
| No running processes found                                     |
+-----+-----+-----+-----+-----+
```

2. 执行gpustat命令。

```
pip install gpustat
gpustat -cp -i
```

```
notebook-6a654129-698e-4635-b6be-67aedbdd4c54 Mon Oct 25 15:19:11 2021 440.33.01
[0] Tesla V100-SXM2-32GB | 31'C, 0% | 0 / 32510 MB |
[1] Tesla V100-SXM2-32GB | 34'C, 0% | 0 / 32510 MB |
```

使用Ctrl+C可以退出。

## 使用 python 命令

1. 执行nvidia-ml-py3命令（常用）。

```
!pip install nvidia-ml-py3
import nvidia_smi
nvidia_smi.nvmlInit()
deviceCount = nvidia_smi.nvmlDeviceGetCount()
for i in range(deviceCount):
    handle = nvidia_smi.nvmlDeviceGetHandleByIndex(i)
    util = nvidia_smi.nvmlDeviceGetUtilizationRates(handle)
    mem = nvidia_smi.nvmlDeviceGetMemoryInfo(handle)
    print(f"|Device {i}| Mem Free: {mem.free/1024**2:5.2f}MB / {mem.total/1024**2:5.2f}MB | gpu-util:
{util.gpu:3.1%} | gpu-mem: {util.memory:3.1%} |")
```

```
Output:
|Device 0| Mem Free: 32510.44MB / 32510.50MB | gpu-util: 0.0% | gpu-mem: 0.0% |
|Device 1| Mem Free: 32510.44MB / 32510.50MB | gpu-util: 0.0% | gpu-mem: 0.0% |
```

2. 执行nvidia\_smi + wapper + prettytable命令。

用户可以将GPU信息显示操作看作一个装饰器，在模型训练过程中就可以实时的显示GPU状态信息。

```
def gputil_decorator(func):
    def wrapper(*args, **kwargs):
        import nvidia_smi
        import prettytable as pt

        try:
```

```

table = pt.PrettyTable(['Devices','Mem Free','GPU-util','GPU-mem'])
nvidia_smi.nvmlInit()
deviceCount = nvidia_smi.nvmlDeviceGetCount()
for i in range(deviceCount):
    handle = nvidia_smi.nvmlDeviceGetHandleByIndex(i)
    res = nvidia_smi.nvmlDeviceGetUtilizationRates(handle)
    mem = nvidia_smi.nvmlDeviceGetMemoryInfo(handle)
    table.add_row([i, f"{mem.free/1024**2:5.2f}MB/{mem.total/1024**2:5.2f}MB",
f"{res.gpu:3.1%}", f"{res.memory:3.1%}"])

except nvidia_smi.NVMLError as error:
    print(error)

print(table)
return func(*args, **kwargs)
return wrapper

```

Output:

Devices	Mem Free	GPU-util	GPU-mem
0	32510.44MB/32510.50MB	0.0%	0.0%
1	32510.44MB/32510.50MB	0.0%	0.0%

### 3. 执行pynvml命令。

nvidia-ml-py3可以直接查询nvml c-lib库，而无需通过nvidia-smi。因此，这个模块比nvidia-smi周围的包装器快得多。

```

from pynvml import *
nvmlInit()
handle = nvmlDeviceGetHandleByIndex(0)
info = nvmlDeviceGetMemoryInfo(handle)
print("Total memory:", info.total)
print("Free memory:", info.free)
print("Used memory:", info.used)

```

Output:

```

Total memory: 34089730048
Free memory: 34089664512
Used memory: 65536

```

### 4. 执行gputil命令。

```

!pip install gputil
import GPUUtil as GPU
GPU.showUtilization()

```

Output:

ID	GPU	MEM
0	0%	25%
1	0%	0%
...		

```

import GPUUtil as GPU
GPUs = GPU.getGPUs()
for gpu in GPUs:
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total
{3:.0f}MB".format(gpu.memoryFree, gpu.memoryUsed, gpu.memoryUtil*100, gpu.memoryTotal))

```

```
Output:
GPU RAM Free: 32510MB | Used: 0MB | Util 0% | Total 32510MB
GPU RAM Free: 32510MB | Used: 0MB | Util 0% | Total 32510MB
```

注：用户在使用pytorch/tensorflow等深度学习框架时也可以使用框架自带的api进行查询。

## 6.31 在 ModelArts 的 Notebook 中 JupyterLab 的目录、Terminal 的文件和 OBS 的文件之间的关系是什么？

- JupyterLab目录的文件与Terminal中work目录下的文件相同。即用户在Notebook中新建的，或者是从OBS目录中同步的文件。
- 挂载OBS存储的Notebook，JupyterLab目录的文件可以与OBS的文件进行同步，使用JupyterLab文件上传下载功能。Terminal的文件与JupyterLab目录的文件相同。
- 挂载EVS存储的Notebook，JupyterLab目录的文件可使用Moxing接口或SDK接口，读取OBS中的文件。Terminal的文件与JupyterLab目录的文件相同。

## 6.32 如何在 ModelArts 的 Notebook 实例中使用 ModelArts 数据集？

ModelArts上创建的数据集存放在OBS中，可以将OBS中的数据下载到Notebook中使用。

Notebook中读取OBS数据方式请参见[如何在ModelArts的Notebook中上传下载OBS文件？](#)。

## 6.33 pip 介绍及常用命令

pip是通用的python包的管理工具。它提供了对Python包的查找、下载、安装和卸载的功能。

pip常用命令如下：

```
pip --help#获取帮助
pip install SomePackage==XXXX #指定版本安装
pip install SomePackage #最新版本安装
pip uninstall SomePackage #卸载软件版本
```

其他命令请使用pip --help命令查询。

## 6.34 在 ModelArts 的 Notebook 中不同规格资源/cache 目录的大小是多少？

创建Notebook时，可以根据业务数据量的大小选择资源。

ModelArts会挂载硬盘至“/cache”目录，用户可以使用此目录来储存临时文件。“/cache”与代码目录共用资源，不同资源规格有不同的容量。

映射规则：当前不支持CPU配置cache盘；GPU与昇腾资源为单卡时，cache目录保持500G大小限制；除单卡外，cache盘大小与卡数有关，计算方式为卡数\*500G，上限为3T。详细[表6-1](#)所示。

表 6-1 不同 Notebook 规格资源 “/cache” 目录的大小

规格类别	cache盘大小
GPU-0.25卡	500G*0.25
GPU-0.5卡	500G*0.5
GPU-单卡	500G
GPU-双卡	500G*2
GPU-四卡	500G*4
GPU-八卡	3T
昇腾-单卡	500G
昇腾-双卡	500G*2
昇腾-四卡	500G*4
昇腾-八卡	3T
CPU	--

## 6.35 资源超分对在 ModelArts 的 Notebook 实例有什么影响？

Notebook超分，是指一个节点中CPU、内存共享的场景。为了充分利用资源，在专属池中存在超分情况。

举例：一个专属池中有1个8U64G的CPU节点，如创建2U8G规格的Notebook，因为超分最多可启动  $8U/(2U*0.6) = 6.67$ 个Notebook实例。这里的0.6就是超分比率。即启动该Notebook实例最少需要1.2U的CPU，运行Notebook时最大使用到2U的资源；内存同理，最少需要4.8G的内存，运行时最大使用到8U的内存。

超分情况下会存在实例终止的风险。如1个8U的节点上同时启动了6个2U的实例，如果其中一个实例CPU使用增大到超过节点的上限（8U）时，k8S会将使用资源最多的实例终止掉。

因此超分会带来实例重启的风险，请不要超分使用。

## 6.36 如何在 Notebook 中安装外部库？

ModelArts Notebook中已安装Jupyter、Python程序包等多种环境，包括TensorFlow、MindSpore、PyTorch、Spark等。您也可以使用pip install在Notebook或Terminal中安装外部库。

## 在 Notebook 中安装

例如，通过JupyterLab在“TensorFlow-1.8”的环境中安装Shapely。

1. 打开一个Notebook实例，进入到Launcher界面。
2. 在“Notebook”区域下，选择“TensorFlow-1.8”，新建一个ipynb文件。
3. 在新建的Notebook中，在代码输入栏输入如下命令。

```
!pip install Shapely
```

## 在 Terminal 中安装

例如，通过terminal在“TensorFlow-1.8”的环境中使用pip安装Shapely。

1. 打开一个Notebook实例，进入到Launcher界面。
2. 在“Other”区域下，选择“Terminal”，新建一个terminal文件。
3. 在代码输入栏输入以下命令，获取当前环境的kernel，并激活需要安装依赖的python环境。

```
cat /home/ma-user/README
```

```
source /home/ma-user/anaconda3/bin/activate TensorFlow-1.8
```

### 📖 说明

如果需要在其他python环境里安装，请将命令中“TensorFlow-1.8”替换为其他引擎。

图 6-9 激活环境

```
sh-4.3$cat /home/ma-user/README
Please use one of following command to start the specified framework environment.

for Conda-python3 ----- source /home/ma-user/anaconda3/bin/activate base
for MXNet-1.2.1 ----- source /home/ma-user/anaconda3/bin/activate MXNet-1.2.1
for PySpark-2.3.2 ----- source /home/ma-user/anaconda3/bin/activate PySpark-2.3.2
for Pytorch-1.0.0 ----- source /home/ma-user/anaconda3/bin/activate Pytorch-1.0.0
for TensorFlow-1.13.1 ----- source /home/ma-user/anaconda3/bin/activate TensorFlow-1.13.1
for TensorFlow-1.8 ----- source /home/ma-user/anaconda3/bin/activate TensorFlow-1.8
for XGBoost-Sklearn ----- source /home/ma-user/anaconda3/bin/activate XGBoost-Sklearn
```

4. 在代码输入栏输入以下命令安装Shapely。

```
pip install Shapely
```

## 6.37 在 ModelArts 的 Notebook 中，访问外网速度不稳定怎么办？

为了方便AI开发者在使用Notebook时访问外部资源，ModelArts提供了一个免费的共享网络代理服务。借助这个代理，开发者可以更加便捷地下载所需的各类资源，助力开发工作的顺利进行。

由于该网络代理免费且共享，其性能会受到实时访问量大小的显著影响。当众多用户同时使用代理进行资源下载时，网络带宽会被大量占用，从而导致代理速度下降，下载速度变慢。相反，在访问量较少时，下载速度可能会相对较好。因此，ModelArts无法保证每位用户在任何时刻都能获得稳定、快速的下载体验。

为了避免因网络下载不稳定而产生不必要的困扰，建议开发者合理安排下载时间，尽量避开高峰时段。同时，对于一些对下载速度有较高要求的场景，建议提前做好规划，或者考虑使用其他备选方案。

# 7 Standard 模型训练

## 7.1 在 ModelArts 训练得到的模型欠拟合怎么办？

1. 模型复杂化。
  - 对同一个算法复杂化。例如回归模型添加更多的高次项，增加决策树的深度，增加神经网络的隐藏层数和隐藏单元数等。
  - 弃用原来的算法，使用一个更加复杂的算法或模型。例如用神经网络来替代线性回归，用随机森林来代替决策树。
2. 增加更多的特征，使输入数据具有更强的表达能力。
  - 特征挖掘十分重要，尤其是具有强表达能力的特征，可以抵过大量的弱表达能力的特征。
  - 特征的数量并非重点，质量才是，总之强表达能力的特征最重要。
  - 能否挖掘出强表达能力的特征，还在于对数据本身以及具体应用场景的深刻理解，这依赖于经验。
3. 调整参数和超参数。
  - 神经网络中：学习率、学习衰减率、隐藏层数、隐藏层的单元数、Adam优化算法中的 $\beta_1$ 和 $\beta_2$ 参数、batch\_size数值等。
  - 其他算法中：随机森林的树数量，k-means中的cluster数，正则化参数 $\lambda$ 等。
4. 增加训练数据作用不大。

欠拟合一般是因为模型的学习能力不足，一味地增加数据，训练效果并不明显。
5. 降低正则化约束。

正则化约束是为了防止模型过拟合，如果模型压根不存在过拟合而是欠拟合了，那么就考虑是否降低正则化参数 $\lambda$ 或者直接去除正则化项。

## 7.2 在 ModelArts 中训练好后的模型如何获取？

使用自动学习产生的模型只能在ModelArts上部署上线，无法下载至本地使用。

使用自定义算法或者订阅算法训练生成的模型，会存储至用户指定的OBS路径中，供用户下载。

## 7.3 在 ModelArts 上如何获得 RANK\_TABLE\_FILE 用于分布式训练？

ModelArts会帮用户生成RANK\_TABLE\_FILE文件，可通过环境变量查看文件位置。

- 在Notebook中打开terminal，可以运行如下命令查看RANK\_TABLE\_FILE：  
env | grep RANK
- 在训练作业中，您可以在训练启动脚本的首行加入如下代码，把RANK\_TABLE\_FILE的值打印出来：  
os.system('env | grep RANK')

## 7.4 在 ModelArts 上训练模型如何配置输入输出数据？

ModelArts支持用户上传自定义算法创建训练作业。上传自定义算法前，请完成[创建算法](#)并上传至OBS桶。创建算法请参考[开发用于预置框架训练的代码](#)。创建训练作业请参考[创建训练作业](#)指导。

### 解析输入路径参数、输出路径参数

运行在ModelArts的模型读取存储在OBS服务的数据，或者输出至OBS服务指定路径，输入和输出数据需要配置3个地方：

1. 训练代码中需解析输入路径参数和输出路径参数。ModelArts推荐以下方式实现参数解析。

```
import argparse
# 创建解析
parser = argparse.ArgumentParser(description="train mnist",
                                formatter_class=argparse.ArgumentDefaultsHelpFormatter)
# 添加参数
parser.add_argument('--train_url', type=str,
                    help='the path model saved')
parser.add_argument('--data_url', type=str, help='the training data')
# 解析参数
args, unknown = parser.parse_known_args()
```

完成参数解析后，用户使用“data\_url”、“train\_url”代替算法中数据来源和数据输出所需的参数。

2. 在使用预置框架创建算法时，根据1中的代码参数设置定义的输入输出参数。
  - 训练数据是算法开发中必不可少的输入。“输入”参数建议设置为“data\_url”，表示数据输入来源，也支持用户根据1的算法代码自定义代码参数。
  - 模型训练结束后，训练模型以及相关输出信息需保存在OBS路径。“输出”数据默认配置为模型输出，代码参数为“train\_url”，也支持用户根据1的算法代码自定义输出路径参数。
3. 在创建训练作业时，填写输入路径和输出路径。  
训练输入选择对应的OBS路径或者数据集路径，训练输出选择对应的OBS路径。



## 7.5 在 ModelArts 上如何提升训练效率并减少与 OBS 的交互？

### 场景描述

在使用ModelArts进行自定义深度学习训练时，训练数据通常存储在对象存储服务（OBS）中，且训练数据较大时（如200GB以上），每次都需要使用GPU资源池进行训练，且训练效率低。

希望提升训练效率，同时减少与对象存储OBS的交互。可通过如下方式进行调整优化。

### 优化原理

对于ModelArts提供的GPU资源池，每个训练节点会挂载500GB的NVMe类型SSD提供给用户免费使用。此SSD挂载到“/cache”目录，“/cache”目录下的数据生命周期与训练作业生命周期相同，当训练作业运行结束以后“/cache”目录下面所有内容会被清空，腾出空间，供下一次训练作业使用。因此，可以在训练过程中将数据从OBS复制到“/cache”目录，然后每次从“/cache”目录读取数据，直到训练结束。训练结束以后“/cache”目录的内容会自动被清空。

### 优化方式

以TensorFlow代码为例。

优化前代码如下所示：

```
...  
tf.flags.DEFINE_string('data_url', '', 'dataset directory.')
```

优化后的代码示例如下，将数据复制至“/cache”目录。

```
...  
tf.flags.DEFINE_string('data_url', '', 'dataset directory.')
```

## 7.6 在 ModelArts 中使用 Moxing 复制数据时如何定义路径变量？

### 问题描述

```
mox.file.copy_parallel(src_obs_dir=input_storage,'obs://dylov8/yolov5_test/yolov5-7.0/datasets'),
```

**mox**这个函数怎么定义以变量的形式填写OBS路径？

### 解决方案

变量定义参考如下示例：

```
input_storage = './test.py'  
import moxing as mox  
mox.file.copy_parallel(input_storage,'obs://dyyolov8/yolov5_test/yolov5-7.0/datasets')
```

## 7.7 在 ModelArts 上如何创建引用第三方依赖包的训练作业?

ModelArts支持训练模型过程中安装第三方依赖包。在训练代码目录下放置“pip-requirements.txt”文件后，在训练启动文件被执行前系统会执行如下命令，以安装用户指定的Python Packages。

```
pip install -r pip-requirements.txt
```

仅使用**预置框架**创建的训练作业支持在训练模型时引用依赖包。

### 📖 说明

pip-requirements.txt文件命名支持以下4种格式，文档中以pip-requirements为例说明。

- pip-requirement.txt
- pip-requirements.txt
- requirement.txt
- requirements.txt
- 代码目录位置请参考[在代码目录下提供安装文件](#)。
- pip-requirements文件写法请参考[安装文件规范](#)。

### 在代码目录下提供安装文件

- 如果使用“我的算法”创建训练作业，则在创建算法时，可以把相关文件放置在配置的“代码目录”下，算法的“启动方式”必须选择“预置框架”。
- 如果使用“自定义算法”创建训练作业，则可以把相关文件放置在配置的“代码目录”下，“启动方式”必须选择“预置框架”。

需要在创建训练作业前将相关文件上传至OBS路径下，文件打包要求请参见[安装文件规范](#)。

### 安装文件规范

请根据依赖包的类型，在代码目录下放置对应文件：

- **依赖包为开源安装包时**

#### 📖 说明

暂时不支持直接从github的源码中安装。

在“代码目录”中创建一个命名为“pip-requirements.txt”的文件，并且在文件中写明依赖包的包名及其版本号，格式为“包名==版本号”。

例如，“代码目录”对应的OBS路径下，包含模型文件，同时还存在“pip-requirements.txt”文件。“代码目录”的结构如下所示：

```
|--模型启动文件所在OBS文件夹  
|---model.py #模型启动文件。  
|---pip-requirements.txt #定义的配置文件，用于指定依赖包的包名及版本号。
```

“pip-requirements.txt”文件内容如下所示：

```
alembic==0.8.6  
bleach==1.4.3  
click==6.6
```

- **依赖包为whl包时**

如果训练后台不支持下载开源安装包或者使用用户编译的whl包时，由于系统无法自动下载并安装，因此需要在“代码目录”放置此whl包，同时创建一个命名为“pip-requirements.txt”的文件，并且在文件中指定此whl包的包名。依赖包必须为“.whl”格式的文件。

例如，“代码目录”对应的OBS路径下，包含模型文件、whl包，同时还存在“pip-requirements.txt”文件。“代码目录”的结构如下所示：

```
|---模型启动文件所在OBS文件夹  
|---model.py      #模型启动文件。  
|---XXX.whl       #依赖包。依赖多个时，此处放置多个。  
|---pip-requirements.txt #定义的配置文件，用于指定依赖包的包名。
```

“pip-requirements.txt”文件内容如下所示：

```
numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl  
tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
```

## 7.8 在 ModelArts 训练时如何安装 C++ 的依赖库？

在训练作业的过程中，会使用到第三方库。以C++为例，请参考如下操作步骤进行安装：

1. 将源码下载至本地并上传到OBS。使用OBS客户端上传文件的操作请参见[上传文件](#)。
2. 将上传到OBS的源码使用Moxing复制到开发环境Notebook中。

以下为使用EVS挂载的开发环境，将数据复制至notebook中的代码示例：

```
import moxing as mox  
mox.file.make_dirs('/home/ma-user/work/data')  
mox.file.copy_parallel('obs://bucket-name/data', '/home/ma-user/work/data')
```

3. 在Jupyter页面的“Files”页签下，单击“New”，打开“Terminal”。执行如下命令进入目标路径，确认源码已下载，即“data”文件是否存在。

```
cd /home/ma-user/work  
ls
```

4. 在“Terminal”环境进行编译，具体编译方式请您根据业务需求进行。
5. 将编译结果使用Moxing复制至OBS中。代码示例如下：

```
import moxing as mox  
mox.file.make_dirs('/home/ma-user/work/data')  
mox.file.copy_parallel('/home/ma-user/work/data', 'obs://bucket-name/file')
```

6. 在训练时，将OBS中的编译结果使用Moxing复制到容器中使用。代码示例如下：

```
import moxing as mox  
mox.file.make_dirs('/cache/data')  
mox.file.copy_parallel('obs://bucket-name/data', '/cache/data')
```

## 7.9 在 ModelArts 训练作业中如何判断文件夹是否复制完毕？

您可以在训练作业启动文件的脚本中，通过如下方式获取复制和被复制文件夹大小，根据结果判断是否复制完毕：

```
import moxing as mox  
mox.file.get_size('obs://bucket_name/obs_file',recursive=True)
```

其中，“get\_size”为获取文件或文件夹的大小。“recursive=True”表示类型为文件夹，“True”表示是文件夹，“False”为文件。

如果输出结果为一致，表示文件夹复制已完结。如果输出结果不一致，表示复制未结束。

## 7.10 如何在 ModelArts 训练作业中加载部分训练好的参数？

在训练作业时，需要从预训练的模型中加载部分参数，初始化当前模型。请您通过如下方式加载：

1. 通过如下代码，您可以查看所有的参数。  

```
from moxing.tensorflow.utils.hyper_param_flags import mox_flags
print(mox_flags.get_help())
```
2. 通过如下方式控制载入模型时需要恢复的参数名。其中，“checkpoint\_include\_patterns”为需要恢复的参数，“checkpoint\_exclude\_patterns”为不需要恢复的参数。  

```
checkpoint_include_patterns: Variables names patterns to include when restoring checkpoint. Such as: conv2d/weights.
checkpoint_exclude_patterns: Variables names patterns to include when restoring checkpoint. Such as: conv2d/weights.
```
3. 通过以下方式控制需要训练的参数列表。其中，“trainable\_include\_patterns”为需要训练的参数列表，“trainable\_exclude\_patterns”为不需要训练的参数列表。  

```
--trainable_exclude_patterns: Variables names patterns to exclude for trainable variables. Such as: conv1,conv2.
--trainable_include_patterns: Variables names patterns to include for trainable variables. Such as: logits.
```

## 7.11 ModelArts 训练时使用 os.system('cd xxx')无法进入文件夹怎么办？

当在训练作业的启动脚本中使用os.system('cd xxx')无法进入相应的文件夹时，建议使用如下方法：

```
import os
os.chdir('/home/work/user-job-dir/xxx')
```

## 7.12 在 ModelArts 训练代码中，如何获取依赖文件所在的路径？

由于用户本地开发的代码需要上传至ModelArts后台，训练代码中涉及到依赖文件的路径时，用户设置有误的场景较多。因此推荐通用的解决方案：使用os接口得到依赖文件的绝对路径，避免报错。

以下示例展示如何通过os接口获得其他文件夹下的依赖文件路径。

文件目录结构：

```
project_root      #代码根目录
├── bootfile.py   #启动文件
├── otherfileDirectory #其他依赖文件所在的目录
│   └── otherfile.py #其他依赖文件
```

在启动文件代码中，建议用户参考以下方式获取其他依赖文件所在路径，即示例中的“otherfile\_path”。

```
import os
current_path = os.path.dirname(os.path.realpath(__file__)) # 获得启动文件bootfile.py的路径
project_root = os.path.dirname(current_path) # 通过启动文件路径获得工程的根目录，对应ModelArts训练控制台上设置的代码目录
otherfile_path = os.path.join(project_root, "otherfileDirectory", "otherfile.py") # 通过工程的根目录得到依赖文件路径
```

## 7.13 自如何获取 ModelArts 训练容器中的文件实际路径？

如果容器中的文件实际路径不清楚，可以使用Python获取当前文件路径的方法获取。

```
os.getcwd() #获取文件当前工作目录路径（绝对路径）
os.path.realpath(__file__) #获得文件所在的路径（绝对路径）
```

也可在搜索引擎寻找其他获取文件路径的方式，使用获取到的路径进行文件读写。

## 7.14 ModelArts 训练中不同规格资源“/cache”目录的大小是多少？

在创建训练作业时可以根据训练作业的大小选择资源。

ModelArts会挂载硬盘至“/cache”目录，用户可以使用此目录来储存临时文件。“/cache”与代码目录共用资源，不同资源规格有不同的容量。

### 📖 说明

- k8s磁盘的驱逐策略是90%，所以可以正常使用的磁盘大小应该是“cache目录容量 x 0.9”。
- 裸机的本地磁盘为物理磁盘，无法扩容，如果存储的数据量大，建议使用SFS存放数据，SFS支持扩容。
- GPU规格的资源

表 7-1 GPU cache 目录容量

GPU规格	cache目录容量
GP Vnt1	800G
8*GP Vnt1	3T
GP Pnt1	800G

- CPU规格的资源

表 7-2 CPU cache 目录容量

CPU规格	cache目录容量
2 核 8GiB	50G
8 核 32GiB	50G

- Ascend规格的资源

表 7-3 Ascend cache 目录容量

Ascend规格	cache目录容量
Ascend	3T

## 7.15 ModelArts 训练作业为什么存在/work 和/ma-user 两种超参目录?

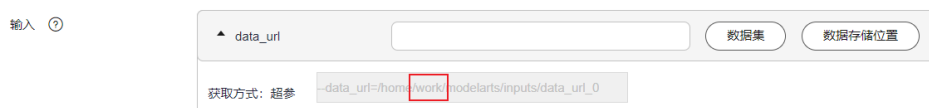
### 问题描述

创建训练作业时，输入输出参数的超参目录有的是/work，有的是/ma-user。

图 7-1 目录是/ma-user



图 7-2 目录是/work

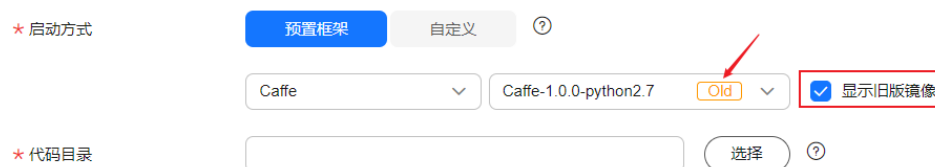


### 解决方案

这是创建训练作业选用的算法有差异导致的。

- 如果选择的算法是使用旧版镜像创建的，那么创建训练作业时输入输出参数的超参目录就是/work。

图 7-3 创建算法



- 如果选择的算法不是使用旧版镜像创建的，那么创建训练作业时输入输出参数的超参目录就是/ma-user。

## 7.16 如何查看 ModelArts 训练作业资源占用情况？

在ModelArts管理控制台，选择“模型训练>训练作业”，进入训练作业列表页面。在训练作业列表中，单击目标作业名称，查看该作业的详情。您可以在“资源占用情况”页签查看到如下指标信息。

- CPU: CPU使用率 (cpuUsage) 百分比 (Percent)。
- MEM: 物理内存使用率 (memUsage) 百分比 (Percent)。
- GPU: GPU使用率 (gpuUtil) 百分比 (Percent)。
- GPU\_MEM: 显存使用率 (gpuMemUsage) 百分比 (Percent)。

## 7.17 如何将将在 ModelArts 中训练好的模型下载或迁移到其他账号？

通过训练作业训练好的模型可以下载，然后将下载的模型上传存储至其他账号对应区域的OBS中。

### 获取模型下载路径

1. 登录ModelArts管理控制台，在左侧导航栏中选择“模型训练 > 训练作业”，进入“训练作业”列表。
2. 在训练作业列表中，单击目标训练作业名称，查看该作业的详情。
3. 在左侧获取“输出位置”下的路径，即为训练模型的下载路径。

### 模型迁移到其他账号

您可以通过如下两种方式将训练的模型迁移到其他账号。

- 将训练好的模型下载至本地后，上传至目标账号对应区域的OBS桶中。
- 通过对模型存储的目标文件夹或者目标桶配置策略，授权其他账号进行读写操作。详情请参见[配置高级桶策略](#)。

# 8 Standard 推理部署

## 8.1 如何将 Keras 的.h5 格式的模型导入到 ModelArts 中?

ModelArts不支持直接导入“.h5”格式的模型。您可以先将Keras的“.h5”格式转换为TensorFlow的格式，然后再导入ModelArts中。

从Keras转TensorFlow操作指导请参见其[官网指导](#)。

## 8.2 ModelArts 导入模型时，如何编写模型配置文件中的安装包依赖参数?

### 问题描述

从OBS中或者从容器镜像中导入模型时，开发者需要编写模型配置文件。模型配置文件描述模型用途、模型计算框架、模型精度、推理代码依赖包以及模型对外API接口。配置文件为JSON格式。配置文件中的“dependencies”，表示配置模型推理代码需要的依赖包，需要提供依赖包名、安装方式和版本约束的信息，详细参数见[模型配置文件编写说明](#)。导入模型时，模型配置文件中的安装包依赖参数“dependencies”如何编写?

### 解决方案

安装包存在前后依赖关系。例如您在安装“mmcv-full”之前，需要完成“Cython”、“pytest-runner”、“pytest”的安装，在配置文件中，您需要把“Cython”、“pytest-runner”、“pytest”写在“mmcv-full”的前面。

示例如下：

```
"dependencies": [
  {
    "installer": "pip",
    "packages": [
      {
        "package_name": "Cython"
      },
      {
        "package_name": "pytest-runner"
      },
      {
        "package_name": "pytest"
      }
    ]
  }
]
```



```
{
  "package_name": "pytest"
},
{
  "restraint": "ATLEAST",
  "package_version": "5.0.0",
  "package_name": "Pillow"
},
{
  "restraint": "ATLEAST",
  "package_version": "1.4.0",
  "package_name": "torch"
},
{
  "restraint": "ATLEAST",
  "package_version": "1.19.1",
  "package_name": "numpy"
},
{
  "package_name": "mmcv-full"
}
]
}
```

当"mmcv-full"安装失败，原因可能是基础镜像中没有安装gcc，无法编译导致安装失败，此时需要用户使用线下wheel包安装。

示例如下：

```
"dependencies": [
  {
    "installer": "pip",
    "packages": [
      {
        "package_name": "Cython"
      },
      {
        "package_name": "pytest-runner"
      },
      {
        "package_name": "pytest"
      },
      {
        "restraint": "ATLEAST",
        "package_version": "5.0.0",
        "package_name": "Pillow"
      },
      {
        "restraint": "ATLEAST",
        "package_version": "1.4.0",
        "package_name": "torch"
      },
      {
        "restraint": "ATLEAST",
        "package_version": "1.19.1",
        "package_name": "numpy"
      },
      {
        "package_name": "mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl"
      }
    ]
  }
]
```

模型配置文件的“dependencies”支持多个“dependency”结构数组以list形式填入。

示例如下:

```
"dependencies": [
  {
    "installer": "pip",
    "packages": [
      {
        "package_name": "Cython"
      },
      {
        "package_name": "pytest-runner"
      },
      {
        "package_name": "pytest"
      },
      {
        "package_name": "mmcv_full-1.3.9-cp37-cp37m-manylinux1_x86_64.whl"
      }
    ]
  },
  {
    "installer": "pip",
    "packages": [
      {
        "restraint": "ATLEAST",
        "package_version": "5.0.0",
        "package_name": "Pillow"
      },
      {
        "restraint": "ATLEAST",
        "package_version": "1.4.0",
        "package_name": "torch"
      },
      {
        "restraint": "ATLEAST",
        "package_version": "1.19.1",
        "package_name": "numpy"
      }
    ]
  }
]
```

## 8.3 在 ModelArts 中使用自定义镜像创建在线服务，如何修改端口？

当模型配置文件中定义了具体的端口号，例如：8443，创建模型没有配置端口，或者配置了其他端口号，均会导致服务部署失败。您需要把模型中的端口号配置为8443，才能保证服务部署成功。

修改默认端口号，具体操作如下：

1. 登录ModelArts控制台，左侧菜单选择“模型管理”；
2. 单击“创建”，进入创建模型界面，元模型选择“从容器镜像中选择”，选择自定义镜像；
3. 配置“容器调用接口”和端口号，端口号与模型配置文件中的端口保持一致；
4. 设置完成后，单击“立即创建”，等待模型状态变为“正常”；
5. 重新部署在线服务。

## 8.4 ModelArts 平台是否支持多模型导入？

ModelArts平台从对象存储服务（OBS）中导入模型包适用于单模型场景。

如果有多模型复合场景，推荐使用自定义镜像方式，通过从容器镜像（SWR）中选择元模型的方式创建模型部署服务。

制作自定义镜像请参考[从0-1制作自定义镜像并创建AI应用](#)。

## 8.5 在 ModelArts 中导入模型对于镜像大小有什么限制？

ModelArts部署使用的是容器化部署，容器运行时有空间大小限制，当用户的模型文件或者其他自定义文件，系统文件超过容器引擎空间大小时，会提示镜像内空间不足。

当前，公共资源池容器引擎空间的大小最大支持50G，专属资源池容器引擎空间的默认为50G，专属资源池容器引擎空间可在创建资源池时自定义设置，设置专属资源池容器引擎空间不会造成额外费用增加。

如果使用的是OBS导入或者训练导入，则包含基础镜像、模型文件、代码、数据文件和下载安装软件包的大小总和。

如果使用的是自定义镜像导入，则包含解压后镜像和镜像下载文件的大小总和。

## 8.6 ModelArts 在线服务和批量服务有什么区别？

- 在线服务  
将模型部署为一个Web服务，您可以通过管理控制台或者API接口访问在线服务。
- 批量服务  
批量服务可对批量数据进行推理，完成数据处理后自动停止。

批量服务一次性推理批量数据，处理完服务结束。在线服务提供API接口，供用户调用推理。

## 8.7 ModelArts 在线服务和边缘服务有什么区别？

- 在线服务  
将模型部署为一个Web服务，您可以通过管理控制台或者API接口访问在线服务。
- 边缘服务

云端服务是集中化的离终端设备较远，对于实时性要求高的计算需求，把计算放在云上会引起网络延时变长、网络拥塞、服务质量下降等问题。而终端设备通常计算能力不足，无法与云端相比。在此情况下，通过在靠近终端设备的地方建立边缘节点，将云端计算能力延伸到靠近终端设备的边缘节点，从而解决上述问题。

智能边缘平台（Intelligent EdgeFabric）通过纳管您的边缘节点，提供将云上应用延伸到边缘的能力，联动边缘和云端的数据，满足客户对边缘计算资源的远程管控、数据处理、分析决策、智能化的诉求。

ModelArts支持将模型通过智能边缘平台IEF，在边缘节点将模型部署为一个Web服务。您可以通过API接口访问边缘服务。

## 8.8 在 ModelArts 中部署模型时，为什么无法选择 Ascend Snt3 资源？

由于Ascend Snt3资源有限，当资源售罄后，您在部署上线时，无法选择Ascend Snt3资源（公共资源池）进行推理，即在部署页面中，“Ascend: 1\* Snt3 (8GB) | ARM: 3核 6GB”资源为灰色，无法选择。

### 解决方案：

- 方法1：如果您希望使用公共资源池下的Ascend Snt3，可以等待其他用户释放，即其他使用Ascend Snt3芯片的服务停止，您即可选择此资源进行部署上线。
- 方法2：如果专属资源池还有Ascend Snt3资源，您可以创建一个Ascend Snt3专属资源池使用。
- 方法3：如果专属资源池的Ascend Snt3资源也已售罄，则需等待其他用户删除Ascend Snt3实例后，您才可以创建Ascend Snt3的专属资源池进行使用。

## 8.9 ModelArts 线上训练得到的模型是否支持离线部署在本地？

通过ModelArts预置算法训练得到的模型是保存在OBS桶里的，模型支持下载到本地。

1. 在训练作业列表找到需要下载模型的训练作业，单击名称进入详情页，获取训练输出路径。

图 8-1 获取训练输出位置

训练作业 / test-  
作业ID cedba060-3f0f-482b-934f-  
状态 ✔ 已完成  
创建时间 2021/10/19 10:57:27  
运行时间 00:02:27  
-----  
算法名称 物体检测  
AI引擎 PyTorch
计算节点个数 1  
规格 GPU: 1\*NVIDIA-V100(32GB) | CPU: 8 核 64GB  
-----  
训练输入

输入路径	训练参数...	本地路径 (训练参数值)
/test-modelart:	data_url	/home/work/modelarts/in...

训练输出

输出路径	训练参数...	本地路径 (训练参数值)
/test-modelarts-	train_url	/home/work/modelarts/ou...

2. 单击“输出路径”，跳转至OBS对象路径，下载训练得到的模型。
3. 在本地环境进行离线部署。

具体请参见[模型调试](#)章节在本地导入模型，参见[服务调试](#)章节，将模型离线部署在本地并使用。

## 8.10 ModelArts 在线服务预测请求体大小限制是多少？

服务部署完成且服务处于运行中后，可以往该服务发送推理的请求，请求的内容根据模型的不同可以是文本，图片，语音，视频等内容。

当使用调用指南页签中显示的调用地址（华为云APIG网关服务的地址）预测时，对请求体的大小限制是12MB，超过12MB时，请求会被拦截。

如果是从ModelArts console的预测页签进行的预测，由于console的网络链路的不同，此时要求请求体的大小不超过8MB。

因此，尽量避免请求体大小超限。如果有高并发的大流量推理请求，请提工单联系专业服务支持。

## 8.11 ModelArts 部署在线服务时，如何避免自定义预测脚本python 依赖包出现冲突？

导入模型时，需同时将对应的推理代码及配置文件放置在模型文件夹下。使用Python编码过程中，推荐采用相对导入方式（Python import）导入自定义包。

如果ModelArts推理框架代码内部存在同名包，而又未采用相对导入，将会出现冲突，导致部署或预测失败。

## 8.12 ModelArts 在线服务预测时，如何提高预测速度？

- 部署在线服务时，您可以选择性能更好的“实例规格”提高预测速度。例如使用GPU资源代替CPU资源。
- 部署在线服务时，您可以增加“实例数”。  
如果实例数设置为1，表示后台的计算模式是单机模式；如果实例数设置大于1，表示后台的计算模式为分布式的。您可以根据实际需求进行选择。
- 推理速度与模型复杂度强相关，您可以尝试优化模型提高预测速度。  
ModelArts中提供了模型版本管理的功能，方便溯源和模型反复调优。

## 8.13 在 ModelArts 中调整模型后，部署新版本模型能否保持原 API 接口不变？

ModelArts提供多版本支持和灵活的流量策略，您可以通过使用灰度发布，实现模型版本的平滑过渡升级。修改服务部署新版本模型或者切换模型版本时，原服务预测API不会变化。

调整模型版本的操作可以参考如下的步骤。

### 前提条件

- 已存在部署完成的服务。
- 已完成模型调整，创建模型。

## 操作步骤

1. 登录ModelArts管理控制台，在左侧导航栏中选择“部署上线 > 在线服务”，默认进入“在线服务”列表。
2. 在部署完成的目标服务中，单击操作列的“修改”，进入“修改服务”页面。
3. 在选择模型及配置中，单击“增加模型版本进行灰度发布”添加新版本。

图 8-2 灰度发布



4. 您可以设置两个版本的流量占比，服务调用请求根据该比例分配。其他设置可参考[参数说明](#)。完成设置后，单击下一步。
5. 确认信息无误后，单击“提交”部署在线服务。

## 8.14 ModelArts 在线服务的 API 接口组成规则是什么？

模型部署成在线服务后，用户可以获取API接口用于访问推理。

API接口组成规则如下：

`https://域名/版本/infer/服务ID`

示例如下：

`https://6ac81cdfac4f4a30be95xxxbb682.apig.xxx.xxx.com/v1/infers/468d146d-278a-4ca2-8830-0b6fb37d3b72`

图 8-3 API 接口



## 8.15 ModelArts 在线服务处于运行中时，如何填写 request header 和 request body?

### 问题现象

部署在线服务完成且在线服务处于“运行中”状态时，通过ModelArts console的调用指南tab页签可以获取到推理请求的地址，但是不知道如何填写推理请求的header及body。

### 原因分析

在线服务部署完成且服务处于运行中状态后，可以通过调用指南页签的调用地址对模型发起预测请求，出于安全考虑，ModelArts会通过相关的认证鉴权机制避免在线服务被无关人员非法调用。所以在预测请求的header信息中包含的是调用者的身份信息，在body部分是需要进行预测的内容。

header的部分需要按照华为云的相关机制进行认证，body部分需要根据模型的要求如前处理脚本的要求，如自定义镜像的要求进行输入。

### 处理方法

- Header:
  - 在调用指南页签上最多可以获取到两个api地址，分别是支持IAM/AKSK认证的地址以及支持APP认证的地址，对于支持不同认证方式的地址，对header的组织也不同，具体如下：
    - IAM/AKSK认证方式：需要在header的X-Auth-Token字段上填入该租户在该region的domain级别的token。具体指导参见连接：[获取IAM用户Token](#)。
    - APP认证的方式：APP认证方式又可以细分为AppCode认证和APP签名认证。
      - AppCode认证需要在header的X-Apig-AppCode字段上填入绑定给该在线服务的APP的AppCode。
      - APP签名认证需要在header的X-Sdk-Date和Authorization字段中填入通过sdk或者工具使用该在线服务绑定的APP的AppKey和AppSecret所生产的这两个字段的值，以完成对该请求的签名认证。具体指导参见链接：[访问在线服务（APP认证）](#)。
- Body:
  - body的组装和模型强相关，不同来源的模型body的组装方式不同。
    - 模型为从容器镜像中导入的：需要按照自定义镜像的要求组织，请咨询该镜像的制作人。
    - 模型为从对象存储(OBS)导入的：此时对body的要求会在推理代码中体现，具体在推理代码的\_preprocess方法中，该方法将输入的http body转换成模型期望的输入，具体的指导可以查看文档：[模型推理代码编写说明](#)。
    - 模型从AI Gallery中获取的：请查看AI Gallery中的调用说明或者咨询该模型的提供方。



## 建议与总结

无

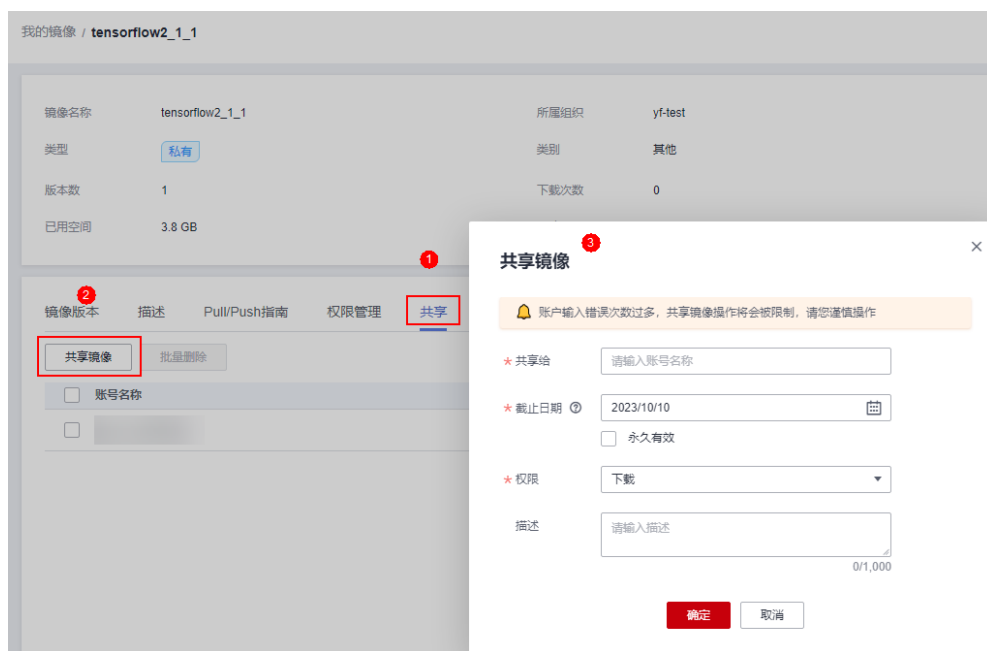
# 9 Standard 镜像相关

## 9.1 不在同一个主账号下，如何使用他人的自定义镜像创建 Notebook？

不是同一个主账号，用户A需要使用用户B的自定义镜像创建Notebook，此时需要用户B将此镜像共享给用户A，用户A将此共享镜像Pull下来注册后方可在Notebook中使用。详细操作如下：

**用户B的操作：**

1. 登录容器镜像服务控制台，进入“我的镜像”页面。
2. 单击需要共享的镜像名称，进入镜像详情页。
3. 在共享页签，单击“共享镜像”，在新窗口中输入共享的账号名称等，单击“确定”。

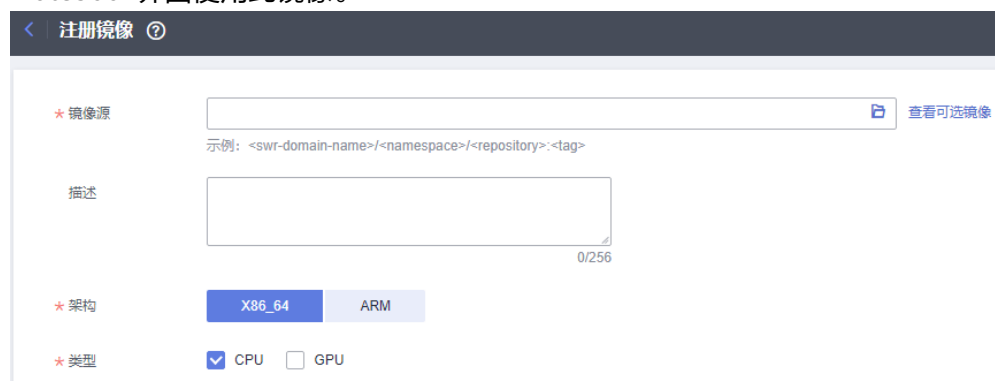


**用户A的操作：**

1. 登录容器镜像服务控制台，在“我的镜像>他人共享”页签下，查看用户B共享的镜像，单击镜像名称进入镜像详情。
2. 按照“Pull/Push指南”页签提供的操作方法，将用户B共享的镜像Pull下来，即作为自有镜像。



3. 进入ModelArts控制台，选择Pull下来的镜像进行镜像注册，注册成功后即可在Notebook界面使用此镜像。



## 9.2 如何登录并上传镜像到 SWR?

本章节介绍如何上传镜像到容器镜像服务SWR。

## Step1 登录 SWR

1. 登录容器镜像服务控制台，选择区域。
2. 单击右上角“创建组织”，输入组织名称完成组织创建。您可以自定义组织名称，本示例使用“deep-learning”，实际操作时请重新命名一个组织名称。后续所有命令中使用到组织名称deep-learning时，均需要替换为此处实际创建的组织名称。
3. 单击右上角“登录指令”，获取登录访问指令。
4. 以root用户登录ECS环境，输入登录指令。

图 9-1 在 ECS 中执行登录指令

```
root@diy-ubuntu1804-cpu-4:~# docker login -u [REDACTED] -p [REDACTED] swr.[REDACTED].com
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

## Step2 上传镜像到 SWR

此小节介绍如何上传镜像至容器镜像服务SWR的镜像仓库。

1. 登录SWR后，使用docker tag命令给上传镜像打标签。下面命令中的组织名称deep-learning，请替换为Step1中实际创建的组织名称，以下所有命令中的deep-learning都需要替换。  
`sudo docker tag tf-1.13.2:latest swr.example.com/deep-learning/tf-1.13.2:latest`

2. 使用docker push命令上传镜像。  
`sudo docker push swr.example.com/deep-learning/tf-1.13.2:latest`

图 9-2 上传镜像

```
root@ecs-7918:~# sudo docker tag tf-1.13.2:latest swr.[REDACTED].com/deep-learning/tf-1.13.2:latest
root@ecs-7918:~# sudo docker push swr.[REDACTED].com/deep-learning/tf-1.13.2:latest
The push refers to repository [swr.[REDACTED].com/deep-learning/tf-1.13.2]
c554b77e0db: Pushed
902871f33e88: Pushed
83ade5a612e2: Pushed
20cfaf8c1ab8: Pushed
bf7955efefcb: Pushed
af6a5fe577ce: Pushed
f4c051ffa5f2: Pushed
184f790bb501: Pushed
dfbea9e01449: Pushed
f0193b2fb026: Pushed
f98177ec269a: Pushed
81e535525773: Pushed
582ab80c9f26: Pushed
4e3516398cef: Pushed
52ad947270f1: Pushed
dd841c774a30: Pushed
37b9a4b22186: Pushed
e0b3afb09dc3: Pushed
6c01b5a53aac: Pushed
2c6ac8e5063e: Pushed
cc967c529ced: Pushed
latest: digest: sha256:b19dad3d95e931eb1a87e5d010f0b987357e618eedb14fbbb3701f3144c106f7 size: 4735
```

3. 完成镜像上传后，在“容器镜像服务控制台>我的镜像”页面可查看已上传的自定义镜像。  
“swr.example.com/deep-learning/tf-1.13.2:latest”即为此自定义镜像的“SWR\_URL”。

## 9.3 在 Dockerfile 中如何给镜像设置环境变量？

在Dockerfile中，可使用ENV指令来设置环境变量，具体信息请参考[Dockerfile指导](#)。

## 9.4 如何通过 docker 镜像启动容器？

Notebook保存后的镜像有Entrypoint参数，如图9-3。Entrypoint参数中指定的可执行文件或命令会覆盖镜像的默认启动命令，Entrypoint中指定的执行命令内容不在镜像中预置，在本地环境通过docker run启动通过Notebook保存的镜像，报错创建容器任务失败，启动文件或目录不存在，如图9-4。

因此需要设置--entrypoint参数，覆盖Entrypoint中指定的程序。使用--entrypoint参数指定的启动文件或命令启动镜像。命令示例如下：

```
docker run -it -d --entrypoint /bin/bash image:tag
```

图 9-3 Entrypoint 参数

```
    ],
    "Cmd": null,
    "Healthcheck": {
      "Test": [
        "NONE"
      ]
    },
    "Image": "sha256:...",
    "Volumes": null,
    "WorkingDir": "/home/ma-user",
    "Entrypoint": [
      "/modelarts/authoring/script/entrypoint/deps/tini/bin/tini",
      "-g",
      "--",
      "/modelarts/authoring/script/entrypoint/notebook/boot/start.sh"
    ],
  ],
```

图 9-4 启动镜像报错

```
root@...:~# docker inspect -f {{.Config.Entrypoint}} swr.cn-north-4.myhuaweicloud.com/hwstaff_public_..._point-test:ul
["/modelarts/authoring/script/entrypoint/deps/tini/bin/tini -g -- /modelarts/authoring/script/entrypoint/notebook/boot/start.sh"]
root@...:~# docker run -it -d swr.cn-north-4.myhuaweicloud.com/hwstaff_public_..._point-test:ul
5cc42a90026b5e0fc42d1115a629e6534d14a5b50b9496d584d3f825991b248d
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed:
runc create failed: unable to start container process: exec: "/modelarts/authoring/script/entrypoint/deps/tini/bin/tini": stat /
modelarts/authoring/script/entrypoint/deps/tini/bin/tini: no such file or directory: unknown.
```

## 9.5 如何在 ModelArts 的 Notebook 中配置 Conda 源？

用户可以在Notebook开发环境中自行安装开发依赖包，方便使用。常见的依赖安装支持pip和Conda，pip源已经配置好，可以直接使用安装，Conda源需要多一步配置。

本章节介绍如何在Notebook开发环境中配置Conda源。

### 配置 Conda 源

Conda软件已经预置在镜像中，具体操作可以参见<https://mirror.tuna.tsinghua.edu.cn/help/anaconda/>。

### 常用 Conda 命令

全部Conda命令建议参考[Conda官方文档](#)。这里仅对常用命令做简要说明。

表 9-1 常用 Conda 命令

命令说明	命令
获取帮助	conda --help conda update --help #获取某一命令的帮助，如update
查看 conda 版本	conda -V
更新 conda	conda update conda #更新 conda conda update anaconda #更新 anaconda
环境管理	conda env list #显示所有的虚拟环境 conda info -e #显示所有的虚拟环境 conda create -n myenv python=3.7 #创建一个名为myenv环境，指定Python版本是3.7 conda activate myenv #激活名为myenv的环境 conda deactivate #关闭当前环境 conda remove -n myenv --all #删除一个名为myenv的环境 conda create -n newname --clone oldname #克隆oldname环境为newname环境
package 管理	conda list #查看当前环境下已安装的package conda list -n myenv #指定myenv环境下安装的package conda search numpy #查找名为numpy的package的所有信息 conda search numpy=1.12.0 --info #查看版本为1.12.0的numpy的信息 conda install numpy pandas #安装numpy和pandas两个package，此命令可同时安装一个或多个包 conda install numpy=1.12.0 #安装指定版本的numpy #install, update及remove命令使用-n指定环境，install及update命令使用-c指定源地址 conda install -n myenv numpy #在myenv的环境中安装名字为numpy的package conda install -c https://conda.anaconda.org/anaconda numpy #使用源 https://conda.anaconda.org/anaconda 安装numpy conda update numpy pandas #更新numpy和pandas两个package，此命令可同时更新一个或多个包 conda remove numpy pandas #卸载numpy和pandas两个package，此命令可同时卸载一个或多个包 conda update --all #更新当前环境下所有的package
清理 conda	conda clean -p # 删除无用的包 conda clean -t # 删除压缩包 conda clean -y --all # 删除所有的安装包及cache

## 安装完外部库后保存镜像环境

ModelArts的新版Notebook提供了镜像保存功能。支持一键将运行中的Notebook实例保存为镜像，将准备好的环境保存下来，可以作为自定义镜像，方便后续使用。保存镜像，安装的依赖包不会丢失。安装完依赖包后，推荐保存镜像，避免安装的依赖包丢失。具体操作请参见[保存Notebook镜像环境](#)。

## 9.6 ModelArts 的自定义镜像软件版本匹配有哪些注意事项？

如果您的自定义镜像涉及NCCL、CUDA、OFED等软件库，当您制作自定义镜像时，您需要确保镜像中的软件库和ModelArts的软件库相匹配。您镜像中的软件版本需要满足以下要求：

- NCCL版本 ≥ 2.7.8。

- OFED版本  $\geq$  MLNX\_OFED\_LINUX-5.4-3.1.0.0。
- CUDA版本需要参考专属资源池的GPU驱动版本，自主进行适配，GPU驱动版本可在专属资源池详情页面查看。

## 9.7 镜像在 SWR 上显示只有 13G，安装少量的包，然后镜像保存过程会提示超过 35G 大小保存失败，为什么？

### 问题现象

我的镜像在SWR侧看，只有13G左右，在开发环境Notebook镜像管理注册，启动Notebook实例后，安装一些包后，镜像保存过程会提示超过35G大小，保存失败？

### 原因分析

SWR侧看到的大小是镜像压缩后的大小，解压后实际大小一般是压缩后的2.5~3倍，所以才安装少量的包后，镜像大小超过35G。

## 9.8 如何保证自定义镜像能不因为超过 35G 而保存失败？

可以从如下几方面考虑：

1. 请选择较小的基础镜像创建Notebook实例，这样在实例中可操作的空间才会大，可自由安装的包才能更多，一般建议原始的启动Notebook的基础镜像在SWR侧查看大小不要超过6G。
2. 镜像保存主要保存在/home/ma-user路径下除挂载路径/home/ma-user/work以外的目录，请将数据集等放到work路径下，不要放到非work路径下。
3. 请不要将实例频繁保存镜像，建议一次将需要的安装包安装好，然后执行镜像保存，避免频繁执行镜像保存的动作，保存次数越多镜像越大，且多次保存后的镜像过大问题无法通过清理磁盘方式减少镜像的大小（ Docker保存原理机制）。

## 9.9 如何减小本地或 ECS 构建镜像的目的镜像的大小？

减小目的镜像大小的最直接的解决办法就是选择尽可能小且符合自己诉求的镜像，比如您需要制作一个PyTorch2.1+Cuda12.2的镜像，官方如果没有提供对应的PyTorch或者Cuda版本的镜像，优选一个没有PyTorch环境或没有安装Cuda的镜像，而不是选择一个PyTorch引擎和Cuda都不满足的镜像，如MindSpore+Cuda11.X，这样基础镜像就会很大，同样的操作最终目的镜像就很大。

此外下面举出几种常见的减少镜像大小的方式。

- 减少目的镜像层数

举例：假设需要安装两个pip包six, numpy，将安装放到同一层，而不是放到不同层：

正确方式：

```
RUN pip install six &&\
  pip install numpy
```

不宜方式：

```
RUN pip install six
RUN pip install numpy
```

镜像层数越多，镜像越大。

- 安装和卸载放在同一层，不要跨层删除。

举例：假设从官网下载了一个SCC包，安装后卸载：

正确方式：

```
RUN mkdir -p /tmp/scc && \  
  cd /tmp/scc && \  
  wget http://100.95.151.167:6868/aarch64/euler/dls-release/euleros-arm/compiled-software/  
seccomponent-1.1.0-release.aarch64.rpm && \  
  rpm -ivh /tmp/scc/seccomponent-1.1.0-release.aarch64.rpm --force --nodeps && \  
  rm -rf /tmp/scc
```

不宜方式：

```
RUN mkdir -p /tmp/scc && \  
  cd /tmp/scc && \  
  wget http://100.95.151.167:6868/aarch64/euler/dls-release/euleros-arm/compiled-software/  
seccomponent-1.1.0-release.aarch64.rpm && \  
  rpm -ivh /tmp/scc/seccomponent-1.1.0-release.aarch64.rpm --force --nodeps  
RUN rm -rf /tmp/scc
```

## 9.10 镜像过大，卸载原来的包重新打包镜像，最终镜像会变小吗？

不会，反而会变大。因为Docker镜像的层原因，当前的镜像是基于原来的镜像制作，而原来的镜像层数是无法改变的，层不变的情况下，大小是不变的，卸载包或者删除数据集，会新增镜像层，镜像反而会变大，这和传统概念的存储不一样。

## 9.11 在 ModelArts 镜像管理注册镜像报错 ModelArts.6787 怎么处理？

### 问题现象

在“镜像管理”界面注册镜像时报错“ModelArts.6787:镜像\*\*\*无法使用，在SWR路径下\*\*\*无法找到指定镜像，请在SWR控制台检查镜像及访问权限配置，或使用其他镜像并重试”。

### 原因分析

报错主要有如下原因：

- 该镜像为主账号注册的private镜像，子账号在使用，而主账号没有给予子账号赋SWR权限，子账号从SWR Console界面看不到该镜像，需要主账号给予子账号在SWR侧赋予SWR权限，使得子账号可以看到该SWR镜像地址，否则该镜像子账号不可使用。
- 该镜像不属于该租户（包括主账号和子账号），是其他人共享的public镜像，而这个镜像又被镜像所有者删除，导致不可使用，用户需要联系对应的SWR镜像负责人，确认镜像是否存在。
- 该镜像不属于该租户（包括主账号和子账号），是其他人共享的public镜像，而这个镜像又被镜像所有者设置成private，导致不可使用，用户需要联系对应的SWR镜像负责人，确认镜像的属性。



## 解决方案

按照原因分析分别解决。

### 9.12 用户如何设置默认的 kernel?

用户希望打开Notebook默认的kernel为自己自定义的kernel。

#### 解决方式:

1. 在Terminal里执行如下命令在镜像里指定环境变量。  

```
# python-3.7.10这里指用户想设置的kernel名称  
export KG_DEFAULT_KERNEL_NAME=python-3.7.10
```
2. 单击操作列的“更多>保存镜像”，保存成功后然后重新启动Notebook。

# 10 Standard 专属资源池

## 10.1 ModelArts 支持使用 ECS 创建专属资源池吗？

不支持。创建资源池时，只能选择界面提供的“未售罄”节点规格进行创建。专属资源池的节点规格后台是对应的ECS资源，但是无法使用账号下购买的ECS，作为ModelArts专属资源池。

## 10.2 在 ModelArts 中 1 个节点的专属资源池，能否部署多个服务？

支持。

在部署服务时，选择专属资源池，在选择“计算节点规格”时选择“自定义规格”，设置小一些或者选择小规格的服务节点规格，当资源池节点可以容纳多个服务节点规格时，就可以部署多个服务。如果使用此方式进行部署推理，选择的规格务必满足模型的要求，当设置的规格过小，无法满足模型的最小推理要求时，则会出现部署失败或预测失败的情况。

图 10-1 设置自定义规格



## 10.3 在 ModelArts 中公共资源池和专属资源池的区别是什么？

- 共享池是所有ModelArts共享的一个资源池，当使用人数比较多的时候，可能造成资源紧张而产生排队。
- 专属池是专属于您的资源池，不会因为资源紧张而产生排队，同时专属资源池支持打通自己的VPC，能和自己的资源网络互通。

## 10.4 ModelArts 中的作业为什么一直处于等待中？

当前训练任务排队的逻辑是先进先出，前面的任务没运行完后面的任务不会运行，有可能会造成小任务被“饿死”，需要用户注意。

### 📖 说明

饿死指的是前面的任务被一个大的任务堵着（例如是64卡），需要等空闲64卡这个任务才能运行，64卡的任务后面跟着1卡的。即使现在空出来30卡，这个1卡的任务也排不上。

- 如果是公共资源池，一般是由于其他用户占用资源导致。有以下方法可以尝试：
  - 如果使用的是免费规格，可以换成收费规格，免费规格资源较少，排队概率高。
  - 规格选择卡数尽量少，如可以选择1卡，相比于选择8卡排队几率大大降低。
  - 可以尝试使用其他Region（如北京四切换为上海一）。
  - 如果有长期的资源使用诉求，可以购买独占使用的专属资源池。
- 如果是专属资源池，建议您进行以下排查：
  - a. 排查专属资源池中是否存在其他作业（包括推理作业、训练作业、开发环境作业等）。

可通过总览页面，快速判断是否有其他模块的作业或实例在运行中，并进入到相关作业或实例上，判断是否使用了专属资源池。如判断相关作业或实例可停止，则可以停止，释放出更多的资源。
  - b. 单击进入专属资源池详情页面，查看作业列表。

观察队头是否有其他作业在排队，如果已有作业在排队，则新建的作业需要继续等待。
  - c. 如果通过排查计算，发现资源确实足够，则考虑可能由于资源碎片化导致的。

例如，集群共2个节点，每个节点都空闲了4张卡，总剩余卡数为8张卡，但用户的作业要求为1节点8张卡，因此无法调度上。

## 10.5 ModelArts 控制台为什么能看到创建失败被删除的专属资源池？

在控制台页面操作删除专属资源池后，后端服务需要进行资源实例释放。在资源实例释放过程中，用户依然可以查询到资源池。如果需要创建专属资源池，建议等待5min后再创建，且不要使用已创建过的专属资源池名称来命名新建的专属资源池。如果做UI自动化测试，建议用例用随机串替代。

## 10.6 ModelArts 训练专属资源池如何与 SFS 弹性文件系统配置对等链接？

配置训练专属资源池与SFS弹性文件系统的对等链接，需要资源池打通VPC，使得资源池与SFS弹性文件系统所配置的VPC相同。配置完成后，在创建训练作业时，就可以看到SFS的配置选项。

打通VPC步骤请参考[打通VPC](#)。

# 11 Edge

## 11.1 在 ModelArts 中使用边缘节点部署边缘服务时能否使用 http 接口协议？

系统默认使用https。如果您想使用http，可以采取以下两种方式：

- 方式一：在部署边缘服务时添加如下环境变量：  
MODELARTS\_SSL\_ENABLED = false

图 11-1 添加环境变量



- 方式二：在使用自定义镜像导入模型时，创建模型页面中“容器调用接口”设置为“http”，再部署边缘服务。

# 12 API/SDK

## 12.1 ModelArts SDK、OBS SDK 和 MoXing 的区别是什么？

### ModelArts SDK

ModelArts服务提供的SDK，可调用ModelArts功能。您可以下载SDK至本地调用接口，也可以在ModelArts Notebook中直接调用。

ModelArts SDK提供了OBS管理、训练管理、模型管理、服务管理等几个模块功能。目前，仅提供了Python语言的ModelArts SDK接口。

详细指导文档：《[ModelArts SDK参考](#)》

### OBS SDK

OBS服务提供的SDK，对OBS进行操作。由于ModelArts较多功能需使用OBS中存储的数据，用户可使用OBS SDK进行调用，使用OBS存储您的数据。

OBS提供了多种语言SDK供选择，开发者可根据使用习惯下载OBS SDK进行调用。使用OBS SDK前，需下载OBS SDK包，然后在本地开发环境中安装使用。

详细指导：《[OBS SDK参考](#)》

### MoXing

MoXing是ModelArts自研的组件，是一种轻型的分布式框架，构建于TensorFlow、PyTorch、MXNet、MindSpore等深度学习引擎之上，使得这些计算引擎分布式性能更高，同时易用性更好。MoXing包含很多组件，其中MoXing Framework模块是一个基础公共组件，可用于访问OBS服务，和具体的AI引擎解耦，在ModelArts支持的所有AI引擎(TensorFlow、MXNet、PyTorch、MindSpore等)下均可以使用。

MoXing Framework模块提供了OBS中常见的数据文件操作，如读写、列举、创建文件夹、查询、移动、复制、删除等。

在ModelArts Notebook中使用MoXing接口时，可直接调用接口，无需下载或安装SDK，使用限制比ModelArts SDK和OBS SDK少，非常便捷。

详细指导：《[MoXing开发指南](#)》

## 12.2 ModelArts 的 API 或 SDK 支持模型下载到本地吗？

ModelArts的API和SDK不支持模型下载到本地，但训练作业输出的模型是存放在对象存储服务（OBS）里面的，您可以通过OBS的API或SDK下载存储在OBS中的文件，具体请参见[从OBS下载文件](#)。

## 12.3 ModelArts 通过 OBS 的 API 访问 OBS 中的文件，属于内网还是公网访问？

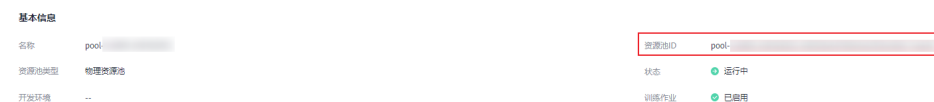
在同一区域，ModelArts通过OBS的API访问OBS中的文件属于内网通信，不消耗公网流量费。

如果是通过互联网从OBS下载数据到本地，这时候会产生OBS公网流量费。OBS的详细计费说明可以参见[计费项](#)。

## 12.4 调用 ModelArts API 接口创建训练作业和部署服务时，如何填写资源池的参数？

- 调用API接口创建训练作业时，“pool\_id”为“资源池ID”。
- 调用API接口部署在线服务时，“pool\_name”为“资源池ID”。

图 12-1 资源池 ID



# 13 Lite Server

## 13.1 GPU A 系列裸金属服务器如何进行 RoCE 性能带宽测试?

### 场景描述

本文主要指导如何在GPU A系列裸金属服务器上测试RoCE性能带宽。

### 前提条件

GPU A系列裸金属服务器已经安装了IB驱动。（网卡设备名称可以使用ibstatus或者ibstat获取。华为云Ant8裸金属服务器使用Ubuntu20.04操作系统默认已经安装IB驱动。）

### 操作步骤

#### 方法1：使用mlx硬件计数器，估算ROCE网卡收发流量

统计300s内流量，统计脚本如下：

```
x=$(cat /sys/class/infiniband/mlx5_2/ports/1/counters/port_rcv_data)
sleep 300
y=$(cat /sys/class/infiniband/mlx5_2/ports/1/counters/port_rcv_data)
res=$((y-x))
echo $res
```

上述获取的值\*4/300，即为当前网卡的接收速率，单位Byte/s。

#### 方法2：使用ib\_write\_bw测试RDMA的读写处理确定带宽

服务器A：服务端从mlx4\_0网卡接收数据

```
ib_write_bw -a -d mlx5_0
```

服务器B：客户端向服务端mlx4\_0网卡发送数据。

```
ib_write_bw -a -F 服务器A的IP -d mlx5_0 --report_gbits
```



图 13-1 服务器 A 执行结果

```
(base) root@devserver-gpu-...-roce-2:~# ib_write_bw -a -d mlx5_0
*****
* Waiting for client to connect... *
*****
-----
RDMA_Write BW Test
Dual-port      : OFF      Device       : mlx5_0
Number of qps  : 1        Transport type : IB
Connection type : RC      Using SRQ    : OFF
PCIe relax order: ON
ibv_wr* API    : ON
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : Ethernet
GID index      : 3
Max inline data : 0[B]
rdma_cm QPs    : OFF
Data ex. method : Ethernet
-----
local address: LID 0000 QPN 0x00be PSN 0x101d60 RKey 0x189535 VAddr 0x007fc0683d9000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:29:31:55:128
remote address: LID 0000 QPN 0x00be PSN 0x342cce RKey 0x189535 VAddr 0x007fb1d0f2f000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:29:31:53:22
-----
#bytes    #iterations    BW peak[MB/sec]    BW average[MB/sec]    MsgRate[Mpps]
8388608   5000            73.96              64.82                  0.000966
-----
```

图 13-2 服务器 B 执行结果

```
(base) root@devserver-gpu-...-roce-3:~# ib_write_bw -a -F 192.168.102.236 -d mlx5_0 --report_gbits
-----
RDMA_Write BW Test
Dual-port      : OFF      Device       : mlx5_0
Number of qps  : 1        Transport type : IB
Connection type : RC      Using SRQ    : OFF
PCIe relax order: ON
ibv_wr* API    : ON
TX depth       : 128
CQ Moderation  : 100
Mtu            : 4096[B]
Link type      : Ethernet
GID index      : 3
Max inline data : 0[B]
rdma_cm QPs    : OFF
Data ex. method : Ethernet
-----
local address: LID 0000 QPN 0x00be PSN 0x342cce RKey 0x189535 VAddr 0x007fb1d0f2f000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:29:31:53:22
remote address: LID 0000 QPN 0x00be PSN 0x101d60 RKey 0x189535 VAddr 0x007fc0683d9000
GID: 00:00:00:00:00:00:00:00:00:00:255:255:29:31:55:128
-----
#bytes    #iterations    BW peak[Gb/sec]    BW average[Gb/sec]    MsgRate[Mpps]
2         5000            0.042403           0.020440              1.277489
4         5000            0.14                0.14                  4.327294
8         5000            0.27                0.20                  3.184282
16        5000            0.54                0.50                  3.925457
32        5000            1.11                1.10                  4.296528
64        5000            1.97                0.76                  1.483763
128       5000            4.43                3.70                  3.615876
256       5000            8.81                5.88                  2.872231
512       5000            17.19               17.17                 4.192633
1024      5000            34.66               34.62                 4.225802
2048      5000            62.22               25.17                 1.536477
4096      5000            91.45               91.37                 2.788505
8192      5000            95.12               39.66                 0.605207
16384     5000            96.05               56.11                 0.428078
32768     5000            95.14               52.60                 0.200663
65536     5000            94.60               44.86                 0.085564
131072    5000            64.14               49.90                 0.047587
262144    5000            81.49               56.76                 0.027066
524288    5000            90.90               59.10                 0.014090
1048576   5000            88.51               65.10                 0.007760
2097152   5000            91.43               88.60                 0.005281
4194304   5000            90.91               86.47                 0.002577
8388608   5000            73.96               64.82                 0.000966
-----
```

## 13.2 GPU A 系列裸金属服务器节点内如何进行 NVLINK 带宽性能测试方法？

### 场景描述

本文指导如何进行节点内NVLINK带宽性能测试，适用的环境为：Ant8或者Ant1 GPU裸金属服务器，且服务器中已经安装相关GPU驱动软件，以及Pytorch2.0。

GPU A系列裸金属服务器，单台服务器GPU间是走NVLINK，可以通过相关命令查询GPU拓扑模式：

```
nvidia-smi topo -m
```

图 13-3 查询 GPU 拓扑模式

Affinity	GPU0	GPU1	GPU2	GPU3	GPU4	GPU5	GPU6	GPU7	NIC0	NIC1	NIC2	NIC3	NIC4	NIC5	NIC6	NIC7
GPU0	X	NV8	NV8	NV8	NV8	NV8	NV8	NV8	PXB	PXB	NODE	NODE	SYS	SYS	SYS	SYS
GPU1	NV8	X	NV8	NV8	NV8	NV8	NV8	NV8	PXB	PXB	NODE	NODE	SYS	SYS	SYS	SYS
GPU2	NV8	NV8	X	NV8	NV8	NV8	NV8	NV8	NODE	NODE	PXB	PXB	SYS	SYS	SYS	SYS
GPU3	NV8	NV8	NV8	X	NV8	NV8	NV8	NV8	NODE	NODE	PXB	PXB	SYS	SYS	SYS	SYS
GPU4	NV8	NV8	NV8	NV8	X	NV8	NV8	NV8	SYS	SYS	SYS	SYS	PXB	PXB	NODE	NODE
GPU5	NV8	NV8	NV8	NV8	NV8	X	NV8	NV8	SYS	SYS	SYS	SYS	PXB	PXB	NODE	NODE
GPU6	NV8	NV8	NV8	NV8	NV8	NV8	X	NV8	SYS	SYS	SYS	SYS	NODE	NODE	PXB	PXB
GPU7	NV8	NV8	NV8	NV8	NV8	NV8	NV8	X	SYS	SYS	SYS	SYS	NODE	NODE	PXB	PXB
NIC0	PXB	PXB	NODE	NODE	SYS	SYS	SYS	SYS	X	PIX	NODE	NODE	SYS	SYS	SYS	SYS
NIC1	PXB	PXB	NODE	NODE	SYS	SYS	SYS	SYS	PIX	X	NODE	NODE	SYS	SYS	SYS	SYS
NIC2	NODE	NODE	PXB	PXB	SYS	SYS	SYS	SYS	NODE	NODE	X	PIX	SYS	SYS	SYS	SYS
NIC3	NODE	NODE	PXB	PXB	SYS	SYS	SYS	SYS	NODE	NODE	PIX	X	SYS	SYS	SYS	SYS
NIC4	SYS	SYS	SYS	SYS	PXB	PXB	NODE	NODE	SYS	SYS	SYS	SYS	X	PIX	NODE	NODE
NIC5	SYS	SYS	SYS	SYS	PXB	PXB	NODE	NODE	SYS	SYS	SYS	SYS	PIX	X	NODE	NODE
NIC6	SYS	SYS	SYS	SYS	NODE	NODE	PXB	PXB	SYS	SYS	SYS	SYS	NODE	NODE	X	PIX
NIC7	SYS	SYS	SYS	SYS	NODE	NODE	PXB	PXB	SYS	SYS	SYS	SYS	NODE	NODE	PIX	X

### 操作步骤

#### 步骤1 使用以下脚本测得GPU服务器内NVLINK带宽性能。

```
import torch
import numpy as np

device = torch.device("cuda")

n_gpus = 8
data_size = 1024 * 1024 * 1024 # 1 GB

speed_matrix = np.zeros((n_gpus, n_gpus))

for i in range(n_gpus):
    for j in range(i + 1, n_gpus):
        print(f"Testing communication between GPU {i} and GPU {j}...")
        with torch.cuda.device(i):
            data = torch.randn(data_size, device=device)
            torch.cuda.synchronize()
        with torch.cuda.device(j):
            result = torch.randn(data_size, device=device)
            torch.cuda.synchronize()
        with torch.cuda.device(i):
            start = torch.cuda.Event(enable_timing=True)
            end = torch.cuda.Event(enable_timing=True)
            start.record()
            result.copy_(data)
            end.record()
            torch.cuda.synchronize()
            elapsed_time_ms = start.elapsed_time(end)
            transfer_rate = data_size / elapsed_time_ms * 1000 * 8 / 1e9
            speed_matrix[i][j] = transfer_rate
            speed_matrix[j][i] = transfer_rate

print(speed_matrix)
```

**步骤2** 以Ant8 GPU裸金属服务器为例，其理论GPU卡间带宽为：NVIDIA\*NVLink\*Bridge for 2GPU: 400GB/s。使用上述测试脚本测得带宽性能进行如下分析。

- 正常模式-NVLINK全互通，带宽约为370GB。基本符合预期，且证明Ant GPU裸金属服务器内部GPU间确实走NVLINK模式，且完全互联。

图 13-4 正常模式带宽性能

	GPU0	GPU1	GPU2	GPU3	GPU4	GPU5	GPU6	GPU7
GPU0	0.00	307.55	369.55	369.00	369.00	368.83	368.59	369.19
GPU1	307.55	0.00	369.26	370.37	368.07	368.78	369.01	370.36
GPU2	369.55	369.26	0.00	368.15	370.90	370.48	370.84	370.95
GPU3	369.00	370.37	368.15	0.00	368.05	370.16	370.42	370.38
GPU4	369.00	368.07	370.90	368.05	0.00	368.02	370.01	370.97
GPU5	368.83	368.78	370.48	370.16	368.02	0.00	369.75	369.99
GPU6	368.59	369.01	370.84	370.42	370.01	369.75	0.00	367.77
GPU7	369.19	370.36	370.95	370.38	370.97	369.99	367.77	0.00
服务器1, 116.204.125.186								

- 异常模式-NVLINK部分互通，出现带宽波动较大的情况。如下图中GPU0和GPU4之间带宽远低于理论值，存在问题。

图 13-5 异常模式带宽性能

	GPU0	GPU1	GPU2	GPU3	GPU4	GPU5	GPU6	GPU7
GPU0	0.00	367.18	368.41	369.04	<b>24.45</b>	368.85	368.73	368.82
GPU1	367.18	0.00	369.53	370.42	370.34	370.53	370.81	370.95
GPU2	368.41	369.53	0.00	370.37	370.78	370.81	370.63	370.92
GPU3	369.04	370.42	370.37	0.00	370.19	370.47	370.70	370.74
GPU4	<b>24.45</b>	370.34	370.78	370.19	0.00	370.25	370.49	370.53
GPU5	368.85	370.53	370.81	370.47	370.25	0.00	370.36	370.13
GPU6	368.73	370.81	370.63	370.70	370.49	370.36	0.00	368.94
GPU7	368.82	370.95	370.92	370.74	370.53	370.13	368.94	0.00

出现这种现象，可尝试重装nvidia/cuda/nvidia-fabricmanager，重装后再测试又恢复到了正式模式，GPU0和GPU4之间带宽恢复到370GB/s。

可能原因如下，仅供参考：

- 驱动程序问题：可能是由于驱动程序没有正确安装或配置，导致NVLINK带宽受限。重新安装nvidia驱动、CUDA和nvidia-fabricmanager等软件后，驱动程序可能已经正确配置，从而解决了这个问题。
- 硬件问题：如果GPU之间的NVLINK连接存在硬件故障，那么这可能会导致带宽受限。重新安装软件后，重启系统，可能触发了某种硬件自检或修复机制，从而恢复了正常的带宽。
- 系统负载问题：最初测试GPU卡间带宽时，可能存在其他系统负载，如进程、服务等，这些负载会占用一部分网络带宽，从而影响NVLINK带宽的表现。重新安装软件后，这些负载可能被清除，从而使NVLINK带宽恢复正常。

----结束

## 13.3 如何将 Ubuntu20.04 内核版本从低版本升级至 5.4.0-144-generic?

### 场景描述

Ubuntu20.04内核版本从低版本升级至5.4.0-144-generic。

## 操作指导

### 步骤1 检查当前内核版本。

```
uname -r
```

### 步骤2 升级内核

```
apt-get install linux-headers-5.4.0-144-generic linux-image-5.4.0-144-generic  
grub-mkconfig -o /boot/efi/EFI/ubuntu/grub.cfg  
reboot
```

第一条命令为安装Linux内核头文件和内核镜像，其中版本为5.4.0-144-generic。

第二条命令为重新生成GRUB引导程序的配置文件，用于在启动计算机时加载操作系统，命令将使用新安装的内核镜像更新GRUB的配置文件，以便在下次启动时加载新的内核。

---结束

## 13.4 如何禁止 Ubuntu 20.04 内核自动升级？

### 场景描述

在Ubuntu 20.04每次内核升级后，系统需要重新启动以加载新内核。如果您已经安装了自动更新功能，则系统将自动下载和安装可用的更新，这可能导致系统在不经意间被重启，如果使用的软件依赖于特定版本的内核，那么当系统自动更新到新的内核版本时，可能会出现兼容性问题。在使用Ubuntu20.04时，建议手动控制内核的更新。

#### 说明

禁用自动更新可能会导致您的系统变得不安全，因为您需要手动安装重要的安全补丁。在禁用自动更新之前，请确保您已了解其中的风险。

### 操作步骤

在Ubuntu 20.04上禁止内核自动升级，步骤如下：

#### 步骤1 禁用unattended-upgrades。

“unattended-upgrades”是一个用于安装安全更新的软件包。要禁用它，首先打开“/etc/apt/apt.conf.d/20auto-upgrades”文件：

```
vi /etc/apt/apt.conf.d/20auto-upgrades
```

将其中的“Unattended-Upgrade "1";”改为“Unattended-Upgrade "0";”以禁用自动更新，然后保存文件并退出。

#### 步骤2 将当前内核版本锁定。

要禁止特定的内核版本更新，您可以使用“apt-mark”命令将其锁定。

首先，检查当前的内核版本：

```
uname -r
```

例如，如果内核版本是“5.4.0-42-generic”，您需要锁定所有与此版本相关的软件包。可执行以下命令：

```
sudo apt-mark hold linux-image-5.4.0-42-generic linux-headers-5.4.0-42-generic linux-modules-5.4.0-42-generic linux-modules-extra-5.4.0-42-generic
```

#### 步骤3 禁用自动更新

要禁用所有自动更新，首先打开“/etc/apt/apt.conf.d/10periodic”文件：  
vi /etc/apt/apt.conf.d/10periodic

修改文件以将所有选项设置为“0”：  
APT::Periodic::Update-Package-Lists "0";  
APT::Periodic::Download-Upgradeable-Packages "0";  
APT::Periodic::AutocleanInterval "0";  
APT::Periodic::Unattended-Upgrade "0";

保存文件并退出。

执行完以上步骤后，您的Ubuntu 20.04系统将不会自动升级内核。

----结束

## 13.5 哪里可以了解 Atlas800 训练服务器硬件相关内容

### 场景描述

本文提供Atlas800训练服务器硬件相关指南，包括三维视图、备件信息、HCCL常用方法以及网卡配置信息。

### Atlas 800 训练服务器三维视图

Atlas 800 训练服务器（型号9000）是基于华为鲲鹏920+Snt9处理器的AI训练服务器，实现完全自主可控，广泛应用于深度学习模型开发和AI训练服务场景，可单击[此处](#)查看硬件三维视图。

### Atlas 800 训练服务器 HCCN Tool

[Atlas 800 训练服务器 1.0.11 HCCN Tool接口参考](#)主要介绍集群网络工具hccn\_tool对外接口说明，包括配置RoCE网卡的IP、网关，配置网络检测对象IP和查询LLDP信息等。

### Atlas 800 训练服务器备件查询助手

[备件查询助手](#)可以帮助您查询服务器的所有部件、规格描述，数量等详细信息。

打开网站后请输入SN编码“2102313LNR10P5100077”，若失效可以提工单至华为云ModelArts查询。

### Atlas 800 训练服务器的网卡配置问题

#### 1. 机头网卡配置是什么？

有以下两类网卡：

- 四个2\*100GE网卡，为RoCE网卡，插在NPU板。
- 一个4\*25GE/10GE，为Hi1822网卡，插在主板上的。

#### 2. ifconfig能看到的网卡信息吗

能看到主板上的网卡信息，即VPC分配的私有IP。如果要看RoCE网卡的命令需要执行“hccn\_tools”命令查看，参考[Atlas 800 训练服务器 1.0.11 HCCN Tool接口参考](#)中的指导。

#### 3. NPU上的网卡在哪里可以看到，会健康检查吗？

8\*NPU的网卡为机头上配置的四个2\*100GE网卡。华为云有网卡健康状态监控机制。

## 13.6 使用 GPU A 系列裸金属服务器有哪些注意事项？

使用华为云A系列裸金属服务器时有如下注意事项：

1. nvidia-fabricmanager版本号必须和nvidia-driver版本号保持一致，可参考[安装nvidia-fabricmanager](#)方法。
2. NCCL必须和CUDA版本相匹配，可单击[此处](#)可查看配套关系和安装方法。
3. 使用该裸金属服务器制作自定义镜像时，必须清除残留文件，请参考[清理文件](#)。

## 13.7 GPU A 系列裸金属服务器如何更换 NVIDIA 和 CUDA？

### 场景描述

当裸金属服务器预置的NVIDIA版本和业务需求不匹配时，需要更换NVIDIA驱动和CUDA版本。本文介绍华为云A系列GPU裸金属服务器（Ubuntu20.04系统）如何从“NVIDIA 525+CUDA 12.0”更换为“NVIDIA 515+CUDA 11.7”。

### 操作步骤

#### 步骤1 卸载原有版本的NVIDIA和CUDA。

1. 查看使用apt包管理方式安装的nvidia软件包，执行如下命令实现查看和卸载。

```
dpkg -l | grep nvidia
dpkg -l | grep cuda
sudo apt-get autoremove --purge nvidia-*
sudo apt-get autoremove --purge cuda-*
```

以上命令可以卸载nvidia-driver、cuda、nvidia-fabricmanager、nvidia-peer-memory四个软件。

但是如果nvidia和cuda是使用runfile(local)方式安装的，那么需要在下一步中再次卸载。

2. 若使用nvidia run包直接安装的驱动，需要找到对应的卸载命令。

```
sudo /usr/bin/nvidia-uninstall
sudo /usr/local/cuda-11.7/bin/cuda-uninstaller
```

3. 验证是否卸载完成。

```
nvidia-smi
nvcc -V
dpkg -l | grep peer
dpkg -l | grep fabricmanager
dpkg -l | grep nvidia
```

#### 步骤2 卸载nccl相关软件。

由于nccl和cuda是配套关系，当cuda版本从12.0更换为11.7的时候，libnccl和libnccl-dev都需要更换为和cuda11.7匹配的版本。因此必须卸载掉原版本。

```
sudo apt-get autoremove --purge *nccl*
```

#### 步骤3 删除原nccl-test的编译后文件。

由于nccl-test make编译也是基于当前cuda12.0版本的。当cuda版本更换后，需要重新编译，因此删除它。默认该文件在/root/nccl-tests直接删除即可。

**步骤4** 从内核中卸载nvidia相关的所有进程。

在安装nvidia驱动时，必须把内核中加载nvidia相关的进程卸载，否则会失败。具体操作请参考[GPU裸金属服务器更换NVIDIA驱动后执行nvidia-smi提示Failed to initialize NVML](#)。

 **说明**

若遇到加载到内核的nvidia进程循环依赖，无法从内核中卸载nvidia，此时执行reboot命令重启服务器即可。

**步骤5** 安装NVIDIA-515和CUDA-11.7配套软件环境。具体步骤请参考[GP Ant8裸金属服务器Ubuntu 20.04安装NVIDIA 515+CUDA 11.7](#)。

----**结束**

# 14 Lite Cluster

## 14.1 Cluster 资源池如何进行 NCCL Test?

ModelArts提供AI诊断功能，用户可以通过NCCL Test，测试节点GPU状态，并且测试多个节点间的通信速度。

### 操作步骤

**步骤1** 单击资源池名称，进入资源池详情。

**步骤2** 单击左侧“AI组件管理 > AI诊断”。

**步骤3** 单击“诊断”，选择“日志上传路径”和NCCL Test节点，其余参数可保持默认值或根据实际需求修改。

- 测试使用的最大数据：取值范围[1, 1024]，单位可选为“B”、“KB”、“MB”、“GB”“TB”。测试使用的最大数据须大于开始测试使用的最小数据。
- 开始测试使用的最小数据：取值范围[1, 1024]，单位可选为“B”、“KB”、“MB”、“GB”“TB”。
- 日志上传路径：AI诊断日志上传路径。
- 数据增加方式：当前支持乘法方式。
- 乘法系数：数值范围[2, 100]。
- 超过时间：数值范围[150, 3600]。
- NCCL Test节点名称列表：不可为空，且被选择的节点须为可用状态。

**步骤4** 单击“确认”，即可开始诊断。

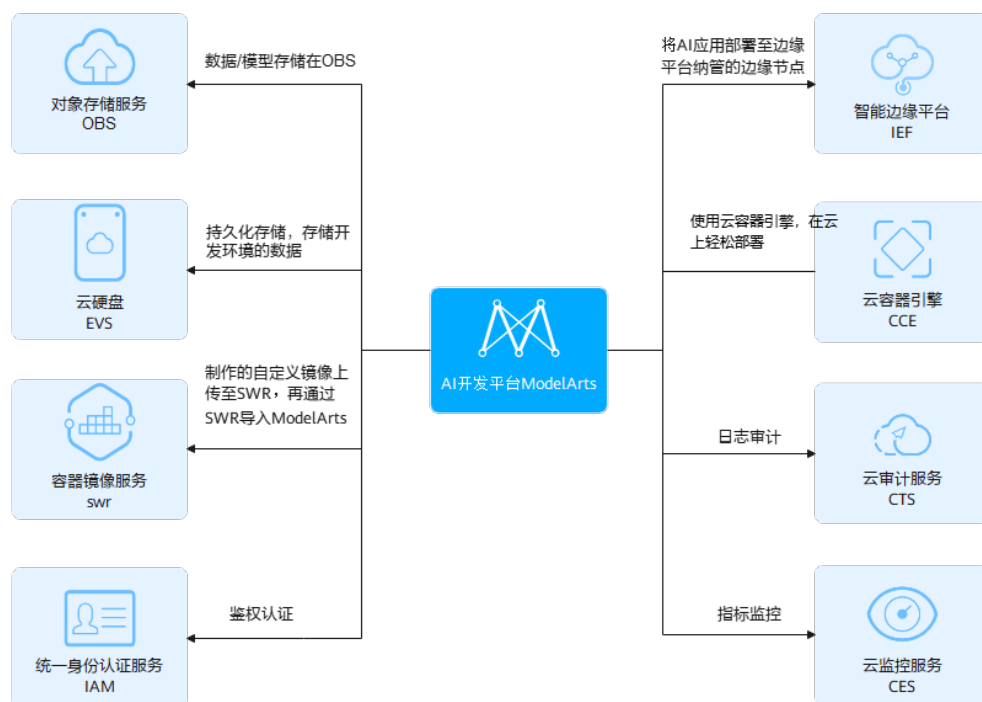
----结束



# 15 历史文档待下线

## 15.1 ModelArts 与其他服务的关系

图 15-1 ModelArts 与其他服务的关系示意图



### 与统一身份认证服务的关系

ModelArts使用统一身份认证服务（Identity and Access Management，简称IAM）实现认证功能。IAM的更多信息请参见《[统一身份认证服务用户指南](#)》。

### 与对象存储服务的关系

ModelArts使用对象存储服务（Object Storage Service，简称OBS）存储数据和模型，实现安全、高可靠和低成本存储需求。OBS的更多信息请参见《[对象存储服务控制台指南](#)》。

表 15-1 ModelArts 各环节与 OBS 的关系

功能	子任务	ModelArts与OBS的关系
自动学习	数据标注	ModelArts标注的数据存储在OBS中。
	自动训练	训练作业结束后，其生成的模型存储在OBS中。
	部署上线	ModelArts将存储在OBS中的模型部署上线为在线服务。
AI全流程开发	数据管理	<ul style="list-style-type: none"> <li>数据集存储在OBS中。</li> <li>数据集的标注信息存储在OBS中。</li> <li>支持从OBS中导入数据。</li> </ul>
	开发环境	Notebook实例中的数据或代码文件存储在OBS中。
	训练模型	<ul style="list-style-type: none"> <li>训练作业使用的数据集存储在OBS中。</li> <li>训练作业的运行脚本存储在OBS中。</li> <li>训练作业输出的模型存储在指定的OBS中。</li> <li>训练作业的过程日志存储在指定的OBS中。</li> </ul>
	AI应用管理	训练作业结束后，其生成的模型存储在OBS中，创建AI应用时，从OBS中导入已有的模型文件。
	部署上线	将存储在OBS中的模型部署上线。
全局配置	-	获取访问授权（使用委托或访问密钥授权），以便ModelArts可以使用OBS存储数据、创建Notebook等操作。

## 与云硬盘的关系

ModelArts使用云硬盘服务（Elastic Volume Service，简称EVS）存储创建的Notebook实例。EVS的更多信息请参见《[云硬盘用户指南](#)》。

## 与云容器引擎的关系

ModelArts使用云容器引擎（Cloud Container Engine，简称CCE）部署模型为在线服务，支持服务的高并发和弹性伸缩需求。CCE的更多信息请参见《[云容器引擎用户指南](#)》。

## 与容器镜像服务的关系

当使用ModelArts不支持的AI框架构建模型时，可通过构建的自定义镜像导入ModelArts进行训练或推理。您可以通过容器镜像服务（Software Repository for Container，简称SWR）制作并上传自定义镜像，然后再通过容器镜像服务导入ModelArts。SWR的更多信息请参见《[容器镜像服务用户指南](#)》。

## 与智能边缘平台的关系

ModelArts可将模型部署至智能边缘平台（Intelligent EdgeFabric，简称IEF）纳管的边缘节点。IEF的更多信息请参见《[智能边缘平台用户指南](#)》。

## 与云监控的关系

ModelArts使用云监控服务（Cloud Eye Service，简称CES）监控在线服务和对应模型负载，执行自动实时监控、告警和通知操作。CES的更多信息请参见《[云监控服务用户指南](#)》。

## 与云审计的关系

ModelArts使用云审计服务（Cloud Trace Service，简称CTS）记录ModelArts相关的操作事件，便于日后的查询、审计和回溯。CTS的更多信息请参见《[云审计服务指南](#)》。

# 15.2 如何上传数据至 OBS？

使用ModelArts进行AI模型开发时，您需要将数据上传至对象存储服务（OBS）桶中。您可以登录OBS管理控制台创建OBS桶，并在您创建的OBS桶中创建文件夹，然后再进行数据的上传，OBS上传数据的详细操作请参见《[对象存储服务快速入门](#)》。

### 📖 说明

- 您在创建OBS桶时，需保证您的OBS桶与ModelArts在同一个区域。如何查看OBS桶与ModelArts的所处区域，请参见[查看OBS桶与ModelArts是否在同一区域](#)。
- 建议根据业务情况及使用习惯，选择OBS使用方法。
  - 如果您的数据量较小（小于100MB）或数据文件少（少于100个），建议您使用控制台上传数据。控制台上传无需工具下载或多余配置，在少量数据上传时，更加便捷高效。
  - 如果您的数据量较大或数据文件较多，建议选择OBS Browser+或obsutil工具上传。OBS Browser+是一个比较常用的图形化工具，支持完善的桶管理和对象管理操作。推荐使用此工具创建桶或上传对象。obsutil是一款用于访问管理OBS的命令行工具，对于熟悉命令行程序的用户，obsutil是执行批量处理、自动化任务的好的选择。
  - 如果您的业务环境需要通过API或SDK执行数据上传操作，或者您习惯于使用API和SDK，推荐选择OBS的API或SDK方法创建桶和上传对象。

上述说明仅罗列OBS常用的使用方式和工具，更多OBS工具说明，请参见《[OBS工具指南](#)》。

## 创建桶

桶是OBS中存储对象的容器，在上传对象前需要先创建桶。OBS提供多种使用方式，您可以根据使用习惯、业务场景选择不同的工具来创建桶。具体参考[OBS文档创建桶章节](#)。

## 上传对象

桶创建成功后，您可以通过以下多种方式将文件上传至桶，OBS最终将这些文件以对象的形式存储在桶中。具体参考[OBS文档上传对象章节](#)。