ModelArts

快速入门（中国站）

文档版本 01

发布日期 2025-08-21

华为云计算技术有限公司

商标声明


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

1 ModelArts 入门指引

本文旨在帮助您了解ModelArts的基本使用流程以及相关的常见问题，帮助您快速上手ModelArts服务。

面向不同AI基础的开发者，本文档提供了相应的入门教程，帮助用户更快速地了解ModelArts的功能，您可以根据经验选择相应的教程。

面向AI开发零基础的用户，您可以使用ModelArts在AI Gallery中预置的模型、算法、数据、Notebook等资产，零代码完成AI建模和应用。

- 如果您想了解如何使用ModelArts Standard一键部署现有的模型，并在线使用模型进行预测，您可以参考**使用ModelArts Standard一键完成商超商品识别模型部署**。

面向AI工程师，熟悉代码编写和调测，您可以使用ModelArts Standard提供的在线代码开发环境，编写训练代码进行AI模型的开发。

- 如果您想了解如何在ModelArts Standard提供的Notebook开发环境中，完成AI开发全流程，您可以参考**基于Codelab使用Standard Notebook实例进行AI开发**。
- 如果您有自己的算法，想改造适配后迁移到ModelArts Standard平台上进行训练和推理，您可以参考**使用自定义算法构建模型（手写数字识别）**。

更多入门实践，请参考《**ModelArts入门实践**》章节。如果您有其他疑问，您也可以通过**华为云社区问答频道**来与我们联系探讨。

2 使用 ModelArts Studio（MaaS）的 DeepSeek-R1 模型框架实现对话问答

场景描述

本案例用于指导用户使用ModelArts Studio大模型即服务平台（下面简称为MaaS）的DeepSeek-R1模型框架，快速实现对话问答。更多MaaS服务的使用指导，请参见**用户指南**。

前提条件

已注册华为账号并开通华为云，进行了实名认证，且在使用ModelArts前检查账号状态，账号不能处于欠费或冻结状态。具体操作，请参见**注册华为账号并开通华为云**和**实名认证**。

步骤一：配置委托访问授权

ModelArts使用过程中涉及到与OBS、SWR等服务交互，首次使用ModelArts需要用户配置委托授权，允许访问这些依赖服务。

1. 登录**ModelArts Studio（MaaS）控制台**，在弹出的免责声明对话框，勾选“我已阅读并同意《ModelArts Studio免责声明》”，然后单击“确定”。
2. 在弹出的“服务授权提醒”对话框，单击“此处”跳转至“添加授权”页面，配置相关信息，勾选“我已经详细阅读并同意《ModelArts服务声明》”，然后单击“创建”。

表 2-1 添加授权配置说明

参数	说明
授权对象类型	选择“IAM子用户”，您也可以根据实际情况进行选择。
授权对象	选择指定的IAM子用户，给指定的IAM子用户配置委托授权。
委托选择	选择“新增委托”。
委托名称	ModelArts会自动生成委托名称，您也可以按实际情况进行修改。

参数	说明
权限配置	选择“普通模式”，在服务列表右侧勾选“全选”。

步骤二：领取免费额度并体验 DeepSeek-R1 模型

1. 在**ModelArts Studio（MaaS）控制台**左侧导航栏，单击“在线推理”。
2. 在“预置服务”页签，单击“免费服务”页签，在DeepSeek-R1服务右侧的“操作”列，单击“领取额度”。

   免费配额将会立即到账，到账后您可进行体验或调用。
3. 在DeepSeek-R1服务右侧的“操作”列，单击“在线体验”，跳转到“文本对话”页面，即可开始问答体验。更多信息，请参见**免费体验MaaS预置服务**。
4. （可选）在DeepSeek-R1服务右侧的“操作”列，单击“调用说明”，获取调用实例代码，修改接口信息和API Key，快速实现端外调用。具体操作，请参见**调用MaaS部署的模型服务**。

步骤三：部署模型服务

如果免费Token额度用完后，还要继续使用该模型，可以付费部署为我的服务使用或者在MaaS预置服务中开通商用服务。下文以部署为我的服务为例进行说明。

1. 登录**ModelArts Studio（MaaS）控制台**，在左侧导航栏单击“在线推理”。
2. 在“在线推理”页面，单击“我的服务”页签，在右上角单击“部署模型服务”。
3. 在“部署模型服务”页面，完成创建配置。

表 2-2 部署模型服务

参数		说明	取值样例
服务设置	服务名称	自定义模型服务的名称。	service-1122
	描述	自定义部署模型服务的简介。	-
模型设置	部署模型	单击选择模型，在“模型广场”页签选择DeepSeek-R1模型，单击“确定”。 **说明** 如果您需要自定义模型的相关参数，可以在“我的模型”页面创建模型，然后在“在线推理”页面的“我的模型”页签进行部署。具体操作，请参见**在MaaS中创建模型**和**使用MaaS部署模型服务**。	DeepSeek-R1

参数		说明	取值样例
资源设置	资源池类型	资源池分为公共资源池与专属资源池。 ● 公共资源池供所有租户共享使用。 ● 专属资源池需单独创建，不与其他租户共享。 如果不支持公共资源池，“公共资源池”按钮会置灰，鼠标悬停时，会提示：该模型版本暂不支持公共资源池部署；如果专属资源池不匹配，勾选按钮会置灰，鼠标悬停时，会出现相关提示，请按照提示进行相关操作。	公共资源池
	实例规格	选择实例规格，规格中描述了服务器类型、型号等信息。	Ascend: 1*ascend-snt9b2 | ARM: 24 vCPUs 96000MB
	流量限制（QPS）	设置待部署模型的流量限制QPS。	1
	实例数	设置服务器个数。	1
更多选项	内容审核	选择是否打开内容审核，默认启用。 ● 开关打开（默认打开），内容审核可以阻止在线推理中的输入输出中出现不合规的内容，但可能会对接口性能产生较大影响。 ● 开关关闭，停用内容审核服务，将不会审核在线推理中的输入输出，模型服务可能会有违规风险，请谨慎关闭。 关闭“内容审核”开关，需要在弹窗中确认是否停用内容审核服务，勾选后，单击“确定”关闭。	打开

参数		说明	取值样例
	事件通知	选择是否打开“事件通知”开关。 • 开关关闭（默认关闭）：表示不启用消息通知服务。 • 开关打开：表示订阅消息通知服务，当任务发生特定事件（如任务状态变化或疑似卡死）时会发送通知。此时必须配置“主题名”和“事件”。 – “主题名”：事件通知的主题名称。单击“创建主题”，前往消息通知服务中创建主题。 – “事件”：选择要订阅的事件类型。例如“创建中”、“已完成”、“运行失败”等。	关闭
	自动停止	当使用付费资源时，可以选择是否打开“自动停止”开关。 • 开关关闭（默认关闭）：表示任务将一直运行。 • 开关打开：表示启用自动停止功能，此时必须配置自动停止时间，支持设置为“1小时”、“2小时”、“4小时”、6小时或“自定义”。启用该参数并设置时间后，运行时长到期后将会自动终止任务，准备排队等状态不扣除运行时长。	关闭

4. 参数配置完成后，单击“提交”。

   “资源池类型”选择“公共资源池”时，会出现“计费提醒”对话框，请您仔细阅读预估费用信息，然后单击“确定”，创建部署任务。模型部署会基于资源占用时长进行计费。服务状态为运行中时会产生费用，最终实际费用以账单为准。

   在“我的服务”列表中，当模型部署服务的“状态”变成“运行中”时，表示模型部署完成。

步骤四：在模型体验使用模型服务

1. 在**ModelArts Studio（MaaS）控制台**左侧导航栏中，选择“在线推理”进入服务列表。
2. 在“在线推理”页面，单击“我的服务”页签，然后在服务列表选择目标模型服务，单击操作列“更多 > 在线体验”。
3. 在“文本对话”页面右上角，单击“参数设置”，拖动或直接输入数值配置推理参数。

   您也可以单击“恢复默认”，将参数值调回默认值。

图 2-1 设置推理参数

表 2-3 参数设置

参数	说明	取值样例
温度/Temperature	设置推理温度。 ● 数值较高，输出结果更加随机。 ● 数值较低，输出结果更加集中和确定。	0.7
核采样/top_p	设置推理核采样。调整输出文本的多样性，数值越大，生成文本的多样性就越高。	1
top_k	选择在模型的输出结果中选择概率最高的前K个结果。	20

4. 在对话框中输入问题，查看返回结果，在线体验对话问答。

后续操作

如果不再需要使用此模型服务，建议清除相关资源，避免产生不必要的费用。

- 在MaaS服务的“在线推理”页面，选择“我的服务”页签，在服务列表选择目标模型服务，单击操作列的“更多 > 删除”，在弹窗中输入“DELETE”，单击“确定”，删除服务。
- 在MaaS服务的“我的模型”页面，单击目标模型对应的“操作”列的“更多 > 删除”，在弹窗中输入“DELETE”，单击“确定”，删除模型。
- 进入OBS控制台，删除本示例使用的OBS桶及文件夹。

3 使用 ModelArts Studio（MaaS）的 Qwen2-7B 模型框架实现对话问答

场景描述

本案例用于指导用户使用ModelArts Studio大模型即服务平台（下面简称为MaaS）的Qwen2-7B模型框架，创建并部署一个模型服务，实现对话问答。通过学习本案例，您可以快速了解如何在MaaS服务上创建和部署模型。更多MaaS服务的使用指导请参见**用户指南**。

操作流程

开始使用如下样例前，请务必按**准备工作**指导完成必要操作。

1. **步骤一：创建我的模型**：使用基础模型创建自定义模型。
2. **步骤二 ： 部署模型服务**：使用创建成功的自定义模型部署模型服务。
3. **步骤三：在模型体验使用模型服务**：在“文本对话”页面，体验部署的模型服务，进行对话问答。

准备工作

- 已注册华为账号并开通华为云，进行了实名认证，且在使用ModelArts前检查账号状态，账号不能处于欠费或冻结状态。具体操作，请参见**注册华为账号并开通华为云**和**实名认证**。
- 配置委托访问授权

  ModelArts使用过程中涉及到与OBS、SWR等服务交互，首次使用ModelArts需要用户配置委托授权，允许访问这些依赖服务。

  a. 使用华为云账号登录**ModelArts管理控制台**，按照版本选择以下操作。
     - 新版本：在左侧导航栏选择“系统管理 > 权限管理”。
     - 旧版本：在左侧导航栏选择“全局配置”。

  b. 单击“添加授权”，配置相关参数。

     下文以IAM子用户为例进行说明，您可以按需修改。

图 3-1 添加授权示例

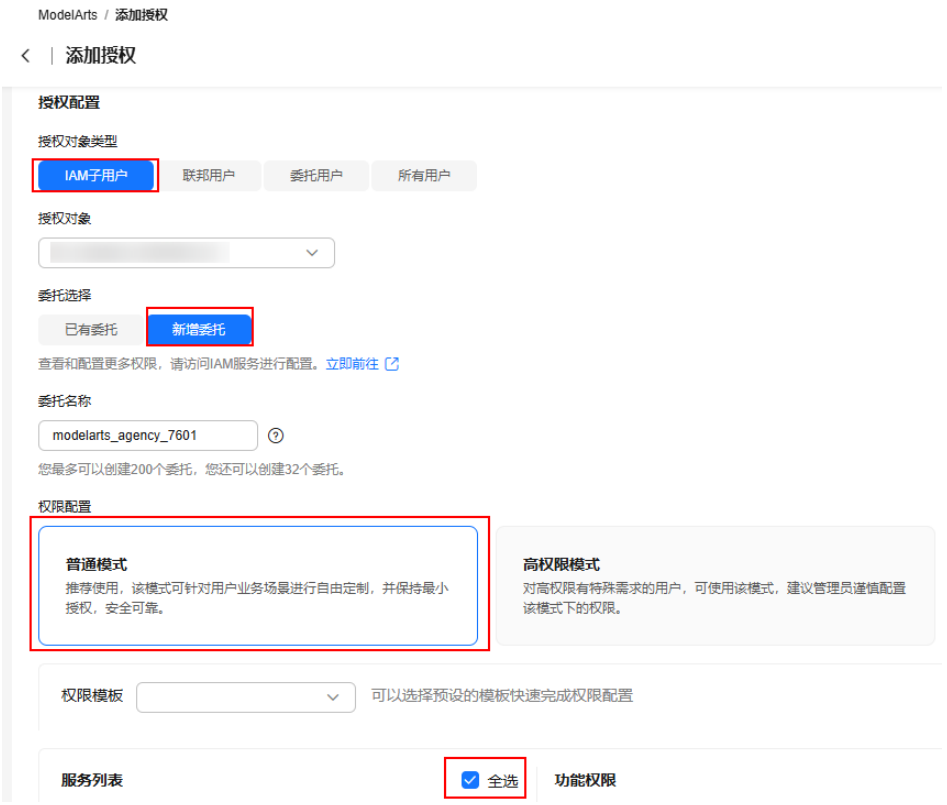


表 3-1 参数说明

参数	说明
“授权对象类型”	选择“IAM子用户”。
“授权对象”	选择指定的IAM子用户，给指定的IAM子用户配置委托授权。
“委托选择”	选择“新增委托”。
“ 委托名称”	系统自动创建委托名称，用户可以手动修改。
“权限配置 ”	选择“普通模式”，在服务列表右侧勾选“全选”。

c. 勾选“我已经阅读并同意《ModelArts服务声明》”，单击“创建”，即可完成委托配置。

步骤一：创建我的模型

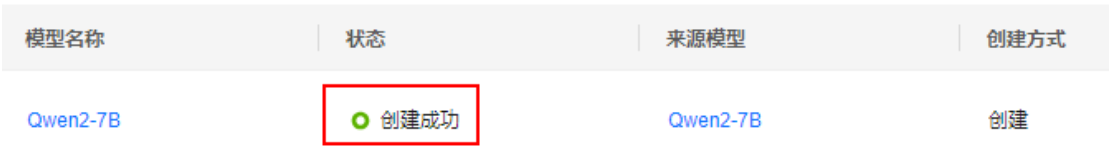
1. 登录**ModelArts Studio（MaaS）控制台**，在顶部导航栏选择目标区域。
2. 在左侧导航栏，选择“我的模型”。
3. 在“我的模型”页面右上角，单击“创建模型”。
4. 在“创建模型”页面，配置相关参数。

表 3-2 创建模型

参数	说明	取值样例
来源模型	单击“选择基础模型”，在弹窗中选择模型，单击“确定”。	Qwen2-7B
模型名称	自定义模型名称。	Qwen2-7B
描述	自定义模型简介。	-
权重设置与词表	默认选择“自定义权重”。	自定义权重
自定义权重存储路径	1. 将权重文件存储到OBS桶中，且权重文件必须满足对应模型的文件格式要求。 权重文件指的是模型的参数集合。 OBS桶必须和MaaS服务在同一个Region下。 • 关于如何获取权重文件，请参见**Hugging Face官网**。 **说明** 如果Hugging Face网站打不开，请在互联网上搜索解决方案。 • 关于权重文件的格式要求，请参见**约束限制**。 • 关于如何将权重文件存储到OBS桶，请参见**上传概述**。 2. 单击“自定义权重存储路径”右侧的文件图标，选择存放模型权重文件的OBS路径（必须选择到模型文件夹），然后单击“确定”。 单次上传本地文件到OBS的总大小不能超过5GB，详情请参见**如何上传超过5GB的大对象**。	/3003****/79abed0d-2622-4cd0-80fc-2065ebe****/

5. 参数配置完成后，单击“创建”，创建自定义模型。
6. 在模型列表，单击模型名称可以进入详情页查看模型详细信息和任务。
   当模型“状态”变成“创建成功”时，表示模型创建完成。

图 3-2 查看我的模型状态



步骤二： 部署模型服务

1. 模型创建成功后，在“我的模型”页面，单击目标模型右侧操作列的“部署”。
2. 在“部署模型服务”页面，完成创建配置。

表 3-3 部署模型服务

参数		说明	取值样例
服务设置	服务名称	自定义模型服务的名称。	service-1122
	描述	自定义部署模型服务的简介。	-
模型设置	部署模型	当从“我的模型”进入部署模型服务页面时，此处默认呈现选择的模型。	Qwen2-7B
资源设置	资源池类型	资源池分为公共资源池与专属资源池。 ● 公共资源池供所有租户共享使用。 ● 专属资源池需单独创建，不与其他租户共享。	公共资源池
	实例规格	选择实例规格，规格中描述了服务器类型、型号等信息。 **说明** 公共资源池暂未完全公开，如需申请使用，请联系与您对接的销售人员或拨打4000-955-988获得支持，您也可以在线提交**售前咨询**。	Ascend: 1*ascend-snt9b2 | ARM: 24 vCPUs 96000MB
	流量限制（QPS）	设置待部署模型的流量限制QPS。	3
	实例数	设置服务器个数。	1
更多选项	内容审核	选择是否打开内容审核，默认启用。 ● 开关打开（默认打开），内容审核可以阻止在线推理中的输入输出中出现不合规的内容，但可能会对接口性能产生较大影响。 ● 开关关闭，停用内容审核服务，将不会审核在线推理中的输入输出，模型服务可能会有违规风险，请谨慎关闭。 关闭“内容审核”开关，需要在弹窗中确认是否停用内容审核服务，勾选后，单击“确定”关闭。	打开

参数		说明	取值样例
	事件通知	选择是否打开“事件通知”开关。 • 开关关闭（默认关闭）：表示不启用消息通知服务。 • 开关打开：表示订阅消息通知服务，当任务发生特定事件（如任务状态变化或疑似卡死）时会发送通知。此时必须配置“主题名”和“事件”。 – “主题名”：事件通知的主题名称。单击“创建主题”，前往消息通知服务中创建主题。 – “事件”：选择要订阅的事件类型。例如“创建中”、“已完成”、“运行失败”等。	关闭
	自动停止	当使用付费资源时，可以选择是否打开“自动停止”开关。 • 开关关闭（默认关闭）：表示任务将一直运行。 • 开关打开：表示启用自动停止功能，此时必须配置自动停止时间，支持设置为“1小时”、“2小时”、“4小时”、6小时或“自定义”。启用该参数并设置时间后，运行时长到期后将会自动终止任务，准备排队等状态不扣除运行时长。	关闭

3. 参数配置完成后，单击“提交”。

   “资源池类型”选择“公共资源池”时，会出现“计费提醒”对话框，请您仔细阅读预估费用信息，然后单击“确定”，创建部署任务。模型部署会基于资源占用时长进行计费。服务状态为运行中时会产生费用，最终实际费用以账单为准。

   在“我的服务”列表中，当模型部署服务的“状态”变成“运行中”时，表示模型部署完成。

步骤三：在模型体验使用模型服务

1. 在**ModelArts Studio（MaaS）控制台**左侧导航栏中，选择“在线推理”。
2. 在“在线推理”页面，单击“我的服务”页签，在目标模型服务右侧，单击操作列“更多 > 在线体验”，进入“文本对话”页面。
3. 在“文本对话”页面右上角，单击“参数设置”，按需拖动或直接输入数值配置推理参数。单击“恢复默认”可以将参数值调回默认值。

图 3-3 设置推理参数

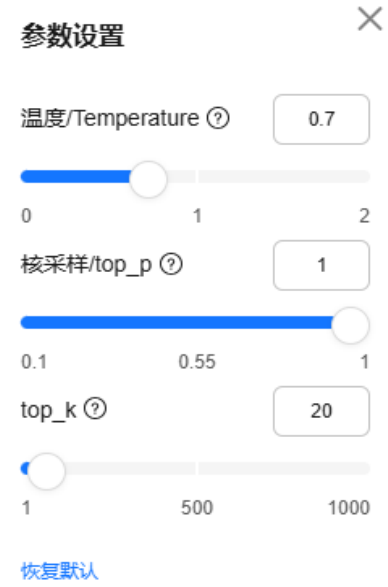


表 3-4 参数设置

参数	说明	取值样例
温度/Temperature	设置推理温度。 ● 数值较高，输出结果更加随机。 ● 数值较低，输出结果更加集中和确定。	0.7
核采样/top_p	设置推理核采样。调整输出文本的多样性，数值越大，生成文本的多样性就越高。	1
top_k	选择在模型的输出结果中选择概率最高的前K个结果。	20

4. 在对话框中输入问题，查看返回结果，在线体验对话问答。

图 3-4 体验模型服务

后续操作

如果不再需要使用此模型服务，建议清除相关资源，避免产生不必要的费用。

- 在ModelArts Studio控制台的“在线推理”页面，选择“我的服务”页签，在需要删除的服务右侧，单击操作列的“更多 > 删除”，在弹窗中输入“DELETE”，单击“确定”，删除服务。
- 在ModelArts Studio控制台的“我的模型”页面，在“Qwen2-7B”模型右侧，单击操作列的“更多 > 删除”，在弹窗中输入“DELETE”，单击“确定”，删除模型。
- 进入OBS控制台，删除本示例使用的OBS桶及文件夹。

4 使用 ModelArts Standard 自定义算法实现手写数字识别

本文为用户提供如何将本地的自定义算法通过简单的代码适配，实现在ModelArts上进行模型训练与部署的全流程指导。

场景描述

本案例用于指导用户使用PyTorch1.8实现手写数字图像识别，示例采用的数据集为MNIST官方数据集。

通过学习本案例，您可以了解如何在ModelArts平台上训练作业、部署推理模型并预测的完整流程。

操作流程

开始使用如下样例前，请务必按**准备工作**指导完成必要操作。

1. **Step1 准备训练数据**：下载MNIST数据集。
2. **Step2 准备训练文件和推理文件**：编写训练与推理代码。
3. **Step3 创建OBS桶并上传文件**：创建OBS桶和文件夹，并将数据集和训练脚本，推理脚本，推理配置文件上传到OBS中。
4. **Step4 创建训练作业**：进行模型训练。
5. **Step5 推理部署**：训练结束后，将生成的模型导入ModelArts用于创建模型，并将模型部署为在线服务。
6. **Step6 预测结果**：上传一张手写数字图片，发起预测请求获取预测结果。
7. **Step7 清除资源**：运行完成后，停止服务并删除OBS中的数据，避免不必要的扣费。

准备工作

- 已注册华为账号并开通华为云，且在使用ModelArts前检查账号状态，账号不能处于欠费或冻结状态。
- 配置委托访问授权

  ModelArts使用过程中涉及到OBS、SWR、IEF等服务交互，首次使用ModelArts需要用户配置委托授权，允许访问这些依赖服务。

a. 使用华为云账号登录**ModelArts管理控制台**，在左侧导航栏单击“权限管理”，进入“权限管理”页面，单击“添加授权”。

b. 在弹出的“添加授权”窗口中，选择：

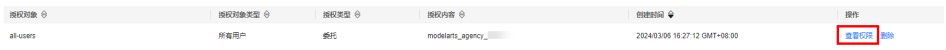
- **授权对象类型：所有用户**
- **委托选择：新增委托**
- **权限配置：普通用户**

选择完成后勾选“我已经详细阅读并同意《ModelArts服务声明》”，然后单击“创建”。

图 4-1 配置委托访问授权

c. 完成配置后，在ModelArts控制台的权限管理列表，可查看到此账号的委托配置信息。

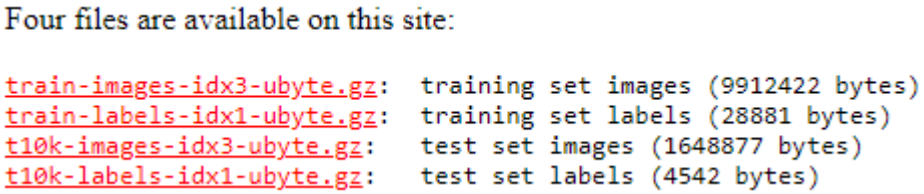
图 4-2 查看委托配置信息



Step1 准备训练数据

本案例使用的数据是MNIST数据集，您可以在浏览器中搜索“MNIST数据集”下载如**图4-3**所示的4个文件。

图 4-3 MNIST 数据集



- “train-images-idx3-ubyte.gz”：训练集的压缩包文件，共包含60000个样本。
- “train-labels-idx1-ubyte.gz”：训练集标签的压缩包文件，共包含60000个样本的类别标签。
- “t10k-images-idx3-ubyte.gz”：验证集的压缩包文件，共包含10000个样本。

- “t10k-labels-idx1-ubyte.gz”：验证集标签的压缩包文件，共包含10000个样本的类别标签。

Step2 准备训练文件和推理文件

针对此案例，ModelArts提供了需使用的训练脚本、推理脚本和推理配置文件。请参考如下文件内容。

说明

粘贴“.py”文件代码时，请直接新建“.py”文件，否则会可能出现“SyntaxError: 'gbk' codec can't decode byte 0xa4 in position 324: illegal multibyte sequence”报错。

粘贴完代码后，建议检查代码文件是否出现中文注释变为乱码的情况，如果出现该情况请将编辑器改为utf-8格式后再粘贴代码。

在本地电脑中创建训练脚本“train.py”，内容如下：

```
# base on https://github.com/pytorch/examples/blob/main/mnist/main.py

from __future__ import print_function

import os
import gzip
import codecs
import argparse
from typing import IO, Union

import numpy as np

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
from torch.optim.lr_scheduler import StepLR

import shutil


# 定义网络模型
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output
```

```
# 模型训练，设置模型为训练模式，加载训练数据，计算损失函数，执行梯度下降
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
            if args.dry_run:
                break


# 模型验证，设置模型为验证模式，加载验证数据，计算损失函数和准确率
def test(model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))


# 以下为pytorch mnist
# https://github.com/pytorch/vision/blob/v0.9.0/torchvision/datasets/mnist.py
def get_int(b: bytes) -> int:
    return int(codecs.encode(b, 'hex'), 16)


def open_maybe_compressed_file(path: Union[str, IO]) -> Union[IO, gzip.GzipFile]:
    """Return a file object that possibly decompresses 'path' on the fly.
       Decompression occurs when argument `path` is a string and ends with '.gz' or '.xz'.
    """
    if not isinstance(path, torch._six.string_classes):
        return path
    if path.endswith('.gz'):
        return gzip.open(path, 'rb')
    if path.endswith('.xz'):
        return lzma.open(path, 'rb')
    return open(path, 'rb')


SN3_PASCALVINCENT_TYPEMAP = {
    8: (torch.uint8, np.uint8, np.uint8),
    9: (torch.int8, np.int8, np.int8),
    11: (torch.int16, np.dtype('>i2'), 'i2'),
    12: (torch.int32, np.dtype('>i4'), 'i4'),
    13: (torch.float32, np.dtype('>f4'), 'f4'),
    14: (torch.float64, np.dtype('>f8'), 'f8')
}


def read_sn3_pascalvincent_tensor(path: Union[str, IO], strict: bool = True) -> torch.Tensor:
    """Read an SN3 file in "Pascal Vincent" format (Lush file 'libidx/idx-io.lsh').
```

```
    Argument may be a filename, compressed filename, or file object.
    """
    # read
    with open_maybe_compressed_file(path) as f:
        data = f.read()
    # parse
    magic = get_int(data[0:4])
    nd = magic % 256
    ty = magic // 256
    assert 1 <= nd <= 3
    assert 8 <= ty <= 14
    m = SN3_PASCALVINCENT_TYPEMAP[ty]
    s = [get_int(data[4 * (i + 1): 4 * (i + 2)]) for i in range(nd)]
    parsed = np.frombuffer(data, dtype=m[1], offset=(4 * (nd + 1)))
    assert parsed.shape[0] == np.prod(s) or not strict
    return torch.from_numpy(parsed.astype(m[2], copy=False)).view(*s)


def read_label_file(path: str) -> torch.Tensor:
    with open(path, 'rb') as f:
        x = read_sn3_pascalvincent_tensor(f, strict=False)
    assert(x.dtype == torch.uint8)
    assert(x.ndimension() == 1)
    return x.long()


def read_image_file(path: str) -> torch.Tensor:
    with open(path, 'rb') as f:
        x = read_sn3_pascalvincent_tensor(f, strict=False)
    assert(x.dtype == torch.uint8)
    assert(x.ndimension() == 3)
    return x


def extract_archive(from_path, to_path):
    to_path = os.path.join(to_path, os.path.splitext(os.path.basename(from_path))[0])
    with open(to_path, "wb") as out_f, gzip.GzipFile(from_path) as zip_f:
        out_f.write(zip_f.read())
# --- 以上为pytorch mnist
# --- end


# raw mnist 数据处理
def convert_raw_mnist_dataset_to_pytorch_mnist_dataset(data_url):
    """
    raw

    {data_url}/
        train-images-idx3-ubyte.gz
        train-labels-idx1-ubyte.gz
        t10k-images-idx3-ubyte.gz
        t10k-labels-idx1-ubyte.gz

    processed

    {data_url}/
        train-images-idx3-ubyte.gz
        train-labels-idx1-ubyte.gz
        t10k-images-idx3-ubyte.gz
        t10k-labels-idx1-ubyte.gz
        MNIST/raw
            train-images-idx3-ubyte
            train-labels-idx1-ubyte
            t10k-images-idx3-ubyte
            t10k-labels-idx1-ubyte
        MNIST/processed
            training.pt
            test.pt
    """
```

```
    resources = [
        "train-images-idx3-ubyte.gz",
        "train-labels-idx1-ubyte.gz",
        "t10k-images-idx3-ubyte.gz",
        "t10k-labels-idx1-ubyte.gz"
    ]

    pytorch_mnist_dataset = os.path.join(data_url, 'MNIST')

    raw_folder = os.path.join(pytorch_mnist_dataset, 'raw')
    processed_folder = os.path.join(pytorch_mnist_dataset, 'processed')

    os.makedirs(raw_folder, exist_ok=True)
    os.makedirs(processed_folder, exist_ok=True)

    print('Processing...')

    for f in resources:
        extract_archive(os.path.join(data_url, f), raw_folder)

    training_set = (
        read_image_file(os.path.join(raw_folder, 'train-images-idx3-ubyte')),
        read_label_file(os.path.join(raw_folder, 'train-labels-idx1-ubyte'))
    )
    test_set = (
        read_image_file(os.path.join(raw_folder, 't10k-images-idx3-ubyte')),
        read_label_file(os.path.join(raw_folder, 't10k-labels-idx1-ubyte'))
    )
    with open(os.path.join(processed_folder, 'training.pt'), 'wb') as f:
        torch.save(training_set, f)
    with open(os.path.join(processed_folder, 'test.pt'), 'wb') as f:
        torch.save(test_set, f)

    print('Done!')


def main():
    # 定义可以接收的训练作业运行参数
    parser = argparse.ArgumentParser(description='PyTorch MNIST Example')

    parser.add_argument('--data_url', type=str, default=False,
                        help='mnist dataset path')
    parser.add_argument('--train_url', type=str, default=False,
                        help='mnist model path')

    parser.add_argument('--batch-size', type=int, default=64, metavar='N',
                        help='input batch size for training (default: 64)')
    parser.add_argument('--test-batch-size', type=int, default=1000, metavar='N',
                        help='input batch size for testing (default: 1000)')
    parser.add_argument('--epochs', type=int, default=14, metavar='N',
                        help='number of epochs to train (default: 14)')
    parser.add_argument('--lr', type=float, default=1.0, metavar='LR',
                        help='learning rate (default: 1.0)')
    parser.add_argument('--gamma', type=float, default=0.7, metavar='M',
                        help='Learning rate step gamma (default: 0.7)')
    parser.add_argument('--no-cuda', action='store_true', default=False,
                        help='disables CUDA training')
    parser.add_argument('--dry-run', action='store_true', default=False,
                        help='quickly check a single pass')
    parser.add_argument('--seed', type=int, default=1, metavar='S',
                        help='random seed (default: 1)')
    parser.add_argument('--log-interval', type=int, default=10, metavar='N',
                        help='how many batches to wait before logging training status')
    parser.add_argument('--save-model', action='store_true', default=True,
                        help='For Saving the current Model')
    args = parser.parse_args()

    use_cuda = not args.no_cuda and torch.cuda.is_available()
```

```
    torch.manual_seed(args.seed)

    # 设置使用 GPU 还是 CPU 来运行算法
    device = torch.device("cuda" if use_cuda else "cpu")

    train_kwargs = {'batch_size': args.batch_size}
    test_kwargs = {'batch_size': args.test_batch_size}
    if use_cuda:
        cuda_kwargs = {'num_workers': 1,
                       'pin_memory': True,
                       'shuffle': True}
        train_kwargs.update(cuda_kwargs)
        test_kwargs.update(cuda_kwargs)

    # 定义数据预处理方法
    transform=transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))
        ])

    # 将 raw mnist 数据集转换为 pytorch mnist 数据集
    convert_raw_mnist_dataset_to_pytorch_mnist_dataset(args.data_url)

    # 分别创建训练和验证数据集
    dataset1 = datasets.MNIST(args.data_url, train=True, download=False,
                       transform=transform)
    dataset2 = datasets.MNIST(args.data_url, train=False, download=False,
                       transform=transform)

    # 分别构建训练和验证数据迭代器
    train_loader = torch.utils.data.DataLoader(dataset1, **train_kwargs)
    test_loader = torch.utils.data.DataLoader(dataset2, **test_kwargs)

    # 初始化神经网络模型并复制模型到计算设备上
    model = Net().to(device)
    # 定义训练优化器和学习率策略，用于梯度下降计算
    optimizer = optim.Adadelta(model.parameters(), lr=args.lr)
    scheduler = StepLR(optimizer, step_size=1, gamma=args.gamma)

    # 训练神经网络，每一轮进行一次验证
    for epoch in range(1, args.epochs + 1):
        train(args, model, device, train_loader, optimizer, epoch)
        test(model, device, test_loader)
        scheduler.step()

    # 保存模型与适配 ModelArts 推理模型包规范
    if args.save_model:

        # 在 train_url 训练参数对应的路径内创建 model 目录
        model_path = os.path.join(args.train_url, 'model')
        os.makedirs(model_path, exist_ok = True)

        # 按 ModelArts 推理模型包规范，保存模型到 model 目录内
        torch.save(model.state_dict(), os.path.join(model_path, 'mnist_cnn.pt'))

        # 复制推理代码与配置文件到 model 目录内
        the_path_of_current_file = os.path.dirname(__file__)
        shutil.copyfile(os.path.join(the_path_of_current_file, 'infer/customize_service.py'),
os.path.join(model_path, 'customize_service.py'))
        shutil.copyfile(os.path.join(the_path_of_current_file, 'infer/config.json'), os.path.join(model_path,
'config.json'))

if __name__ == '__main__':
    main()
```

在本地电脑中创建推理脚本“customize_service.py”，内容如下：

```
import os
import log
import json
```

```
import torch.nn.functional as F
import torch.nn as nn
import torch
import torchvision.transforms as transforms

import numpy as np
from PIL import Image

from model_service.pytorch_model_service import PTServingBaseService

logger = log.getLogger(__name__)

# 定义模型预处理
infer_transformation = transforms.Compose([
    transforms.Resize(28),
    transforms.CenterCrop(28),
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

# 模型推理服务
class PTVisionService(PTServingBaseService):

    def __init__(self, model_name, model_path):
        # 调用父类构造方法
        super(PTVisionService, self).__init__(model_name, model_path)

        # 调用自定义函数加载模型
        self.model = Mnist(model_path)

        # 加载标签
        self.label = [0,1,2,3,4,5,6,7,8,9]

    # 接收request数据，并转换为模型可以接受的输入格式
    def _preprocess(self, data):
        preprocessed_data = {}
        for k, v in data.items():
            input_batch = []
            for file_name, file_content in v.items():
                with Image.open(file_content) as image1:
                    # 灰度处理
                    image1 = image1.convert("L")
                    if torch.cuda.is_available():
                        input_batch.append(infer_transformation(image1).cuda())
                    else:
                        input_batch.append(infer_transformation(image1))
            input_batch_var = torch.autograd.Variable(torch.stack(input_batch, dim=0), volatile=True)
            print(input_batch_var.shape)
            preprocessed_data[k] = input_batch_var

        return preprocessed_data

    # 将推理的结果进行后处理，得到预期的输出格式，该结果就是最终的返回值
    def _postprocess(self, data):
        results = []
        for k, v in data.items():
            result = torch.argmax(v[0])
            result = {k: self.label[result]}
            results.append(result)
        return results

    # 对于输入数据进行前向推理，得到推理结果
    def _inference(self, data):

        result = {}
        for k, v in data.items():
            result[k] = self.model(v)
```

```
    return result

# 定义网络
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output


def Mnist(model_path, **kwargs):
    # 生成网络
    model = Net()

    # 加载模型
    if torch.cuda.is_available():
        device = torch.device('cuda')
        model.load_state_dict(torch.load(model_path, map_location="cuda:0"))
    else:
        device = torch.device('cpu')
        model.load_state_dict(torch.load(model_path, map_location=device))

    # CPU 或者 GPU 映射
    model.to(device)

    # 声明为推理模式
    model.eval()

    return model
```

在本地电脑中推理配置文件“config.json”，内容如下：

```
{
    "model_algorithm": "image_classification",
    "model_type": "PyTorch",
    "runtime": "pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64"
}
```

Step3 创建 OBS 桶并上传文件

将上一步中的数据和代码文件、推理代码文件与推理配置文件，从本地上传到OBS桶中。在ModelArts上运行训练作业时，需要从OBS桶中读取数据和代码文件。

1. 登录**OBS管理控制台**，按照如下示例创建OBS桶和文件夹。

```
{OBS桶}                 # OBS对象桶，用户可以自定义名称，例如：test-modelarts-xx
    -{OBS文件夹}          # OBS文件夹，自定义名称，此处举例为pytorch
        - mnist-data     # OBS文件夹，用于存放训练数据集，可以自定义名称，此处举例为mnist-data
        - mnist-code     # OBS文件夹，用于存放训练脚本train.py，可以自定义名称，此处举例为mnist-code
```

```
        - infer       # OBS文件夹，用于存放推理脚本customize_service.py和配置文件config.json
      - mnist-output    # OBS文件夹，用于存放训练输出模型，可以自定义名称，此处举例为mnist-
output
```

注意

- 创建的OBS桶所在区域和后续使用ModelArts必须在同一个区域Region，否则会导致训练时找不到OBS桶。具体操作可参见**查看OBS桶与ModelArts是否在同一区域**。
- 创建OBS桶时，桶的存储类别请勿选择“归档存储”，归档存储的OBS桶会导致模型训练失败。

2. 上传**Step1 准备训练数据**中下载的MNIST数据集压缩包文件到OBS的“mnist-data”文件夹中。

注意

- 上传数据到OBS中时，请不要加密，否则会导致训练失败。
- 文件无需解压，直接上传压缩包至OBS中即可。

3. 上传训练脚本“train.py”到“mnist-code”文件夹中。
4. 上传推理脚本“customize_service.py”和推理配置文件“config.json”到“mnist-code”的“infer”文件中。

Step4 创建训练作业

1. 登录**ModelArts管理控制台**，选择和OBS桶相同的区域。
2. 在“权限管理”中检查当前账号是否已完成访问授权的配置。如未完成，请参考**使用委托授权**。针对之前使用访问密钥授权的用户，建议清空授权，然后使用委托进行授权。
3. 在左侧导航栏选择“模型训练 > 训练作业”进入训练作业页面，单击“创建训练作业”。
4. 填写创建训练作业相关信息。
   - “创建方式”：选择“自定义算法”。
   - “启动方式”：选择“预置框架”，下拉框中选择PyTorch，pytorch_1.8.0-cuda_10.2-py_3.7-ubuntu_18.04-x86_64。
   - “代码目录”：选择已创建的OBS代码目录路径，例如“/test-modelarts-xx/pytorch/mnist-code/”（test-modelarts-xx需替换为您的OBS桶名称）。
   - “启动文件”：选择代码目录下上传的训练脚本“train.py”。
   - “输入”：单击“增加训练输入”，设置训练输入的“参数名称”为“data_url”。设置数据存储位置为您的OBS目录，例如“/test-modelarts-xx/pytorch/mnist-data/”（test-modelarts-xx需替换为您的OBS桶名称）。
   - “输出”：单击“增加训练输出”，设置训练输出的“参数名称”为“train_url”。设置数据存储位置为您的OBS目录，例如“/test-modelarts-xx/pytorch/mnist-output/”（test-modelarts-xx需替换为您的OBS桶名称）。预下载至本地目录选择“不下载”。
   - “资源类型”：选择GPU单卡的规格。

- 其他参数保持默认即可。

  说明

  本样例代码为单机单卡场景，选择多卡规格会导致训练失败。

5. 单击“提交”，确认训练作业的参数信息，确认无误后单击“确定”。

   页面自动返回“训练作业”列表页，当训练作业状态变为“已完成”时，即完成了模型训练过程。

   说明

   本案例的训练作业预计运行十分钟。

6. 单击训练作业名称，进入作业详情界面查看训练作业日志信息，观察日志是否有明显的Error信息，如果有则表示训练失败，请根据日志提示定位原因并解决。
7. 在训练详情页左下方单击训练输出路径，如图4-4所示，跳转到OBS目录，查看是否存在model文件夹，且model文件夹中是否有生成训练模型。如果未生成model文件夹或者训练模型，可能是训练输入数据不完整导致，请检查训练数据上传是否完整，并重新训练。

图 4-4 训练输出路径

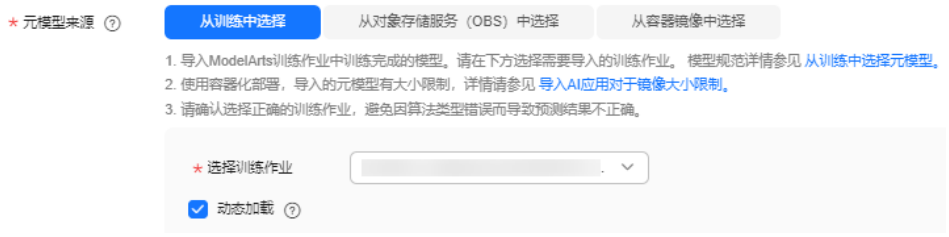
Step5 推理部署

模型训练完成后，可以创建模型，将模型部署为在线服务。

1. 登录ModelArts管理控制台，单击左侧导航栏中的“模型管理”，进入“自定义模型”页面，单击“创建模型”。
2. 在“创建模型”页面，填写相关参数，然后单击“立即创建”。

   在“元模型来源”中，选择“从训练中选择”页签，选择Step4 创建训练作业中完成的训练作业，勾选“动态加载”。AI引擎的值是系统自动写入的，无需设置。

图 4-5 设置元模型来源



3. 在模型列表页面，当模型状态变为“正常”时，表示模型创建成功。单击模型操作列的“部署”，弹出“版本列表”，单击操作列“部署>在线服务”，将模型部署为在线服务。

图 4-6 部署在线服务

4. 在“部署”页面，参考下图填写参数，然后根据界面提示完成在线服务创建。本案例适用于CPU规格，节点规格需选择CPU。如果有免费CPU规格，可选择免费规格进行部署（每名用户限部署一个免费的在线服务，如果您已经部署了一个免费在线服务，需要先将其删除才能部署新的免费在线服务）。

图 4-7 部署模型

完成服务部署后，返回在线服务页面列表页，等待服务部署完成，当服务状态显示为“运行中”，表示服务已部署成功。

Step6 预测结果

1. 在“在线服务”页面，单击在线服务名称，进入服务详情页面。
2. 单击“预测”页签，请求类型选择“multipart/form-data”，请求参数填写“image”，单击“上传”按钮上传示例图片，然后单击“预测”。

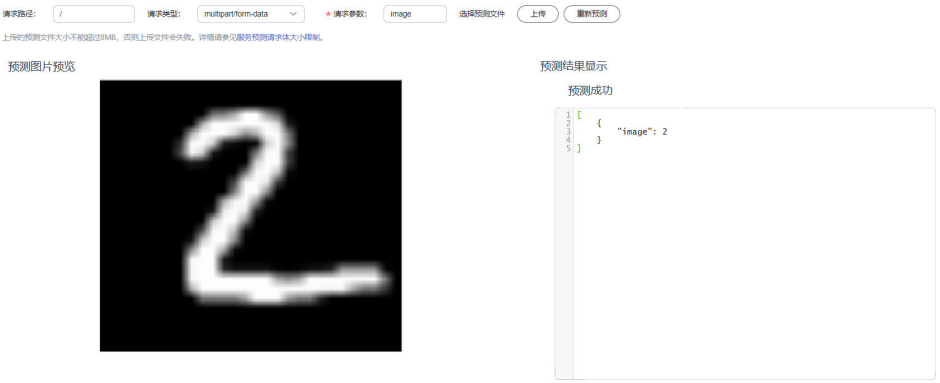
   预测完成后，预测结果显示区域将展示预测结果，根据预测结果内容，可识别出此图片的数字是“2”。

说明

本案例中使用的MNIST是比较简单的用做demo的数据集，配套算法也是比较简单的用于教学的神经网络算法。这样的数据和算法生成的模型仅适用于教学模式，并不能应对复杂的预测场景。即生成的模型对预测图片有一定范围和要求，预测图片必须和训练集中的图片相似（黑底白字）才可能预测准确。

图 4-8 示例图片

图 4-9 预测结果展示



Step7 清除资源

如果不再需要使用此模型及在线服务，建议清除相关资源，避免产生不必要的费用。

- 在“在线服务”页面，“停止”或“删除”刚创建的在线服务。
- 在“自定义模型”页面，“删除”刚创建的模型。
- 在“训练作业”页面，“删除”运行结束的训练作业。
- 进入OBS，删除本示例使用的OBS桶及文件夹，以及文件夹的文件。

常见问题

- 训练作业一直在等待中（排队）？

  训练作业状态一直在等待中状态表示当前所选的资源池规格资源紧张，作业需要进行排队，请耐心等待。请参考**训练作业一直在等待中（排队）？**。
- 在ModelArts中选择OBS路径时，找不到已创建的OBS桶?

  请确保创建的桶和ModelArts服务在同一区域，详细操作请参考**查看OBS桶与ModelArts是否在同一个区域**。

5 使用 ModelArts Standard 一键完成商超商品识别模型部署

ModelArts的AI Gallery中提供了大量免费的模型供用户一键部署，进行AI体验学习。

本文以“商超商品识别”模型为例，完成从AI Gallery订阅模型，到ModelArts一键部署为在线服务的免费体验过程。

“商超商品识别”模型可以识别81类常见超市商品（包括蔬菜、水果和饮品），并给出置信度最高的5类商品的置信度得分。

步骤一：准备工作

- 已注册华为账号并开通华为云，进行了实名认证，且在使用ModelArts前检查账号状态，账号不能处于欠费或冻结状态。
  - **注册华为账号并开通华为云**
  - **进行实名认证**
- 配置委托访问授权

  ModelArts使用过程中涉及到OBS等服务交互，首次使用ModelArts需要用户配置委托授权，允许访问这些依赖服务。具体配置操作请参见**配置ModelArts Standard访问授权**。

步骤二：订阅模型

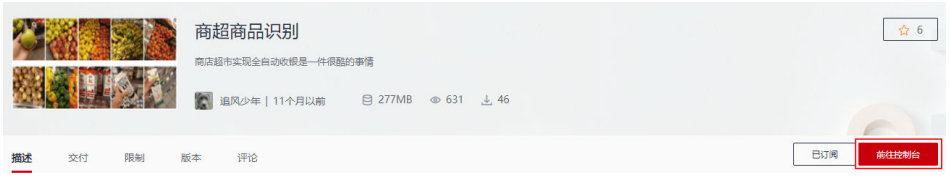
“商超商品识别”的模型共享在AI Gallery中。您可以前往AI Gallery，免费订阅此模型。

1. 单击案例链接**商超商品识别**，进入模型详情页。
2. 完成模型订阅。

   在模型详情页，单击“订阅”，阅读并勾选同意《数据安全与隐私风险承担条款》和《华为云AI Gallery服务协议》，单击“继续订阅”。订阅模型完成后，页面的“订阅”按钮显示为“已订阅”。
3. 从模型详情页进入ModelArts控制台的订阅列表。

   在模型详情页，单击“前往控制台”。在弹出的“选择云服务区域”页面选择ModelArts所在的云服务区域，单击“确定”跳转至ModelArts控制台的“模型管理 > 订阅模型”页面。

图 5-1 前往控制台



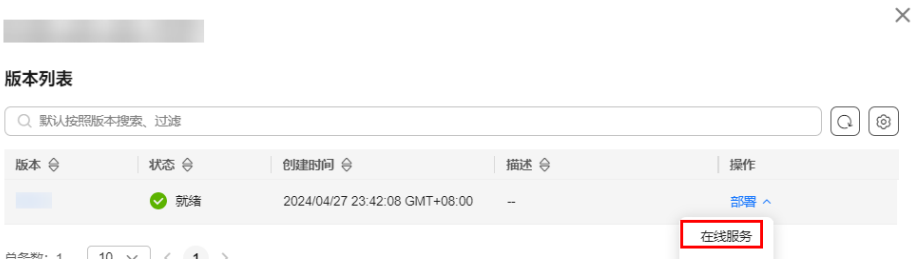
4. 在“订阅模型”列表，单击“版本数量”，在右侧展开版本列表，当订阅模型的版本列表的状态显示为“就绪”时表示模型可以使用。

步骤三：使用订阅模型部署在线服务

模型订阅成功后，可将此模型部署为在线服务

1. 在展开的版本列表中，单击“部署 > 在线服务”跳转至部署页面。

图 5-2 部署模型



2. 在部署页面，参考如下说明填写关键参数。
   - “名称”：自定义一个在线服务的名称，也可以使用默认值，此处以“商超商品识别服务”为例。
   - “资源池”：选择“公共资源池”。
   - “模型来源”和“选择模型及版本”：会自动选择订阅模型。
   - “实例规格”：在下拉框中选择推理使用的计算资源。如果有免费资源，建议选择免费资源。
   - 其他参数可使用默认值。
3. 参数配置完成后，单击“下一步”，确认规格参数后，单击“提交”启动在线服务的部署。
4. 任务提交成功后，单击“查看服务详情”，等待服务状态变为“运行中”时，表示服务部署成功。预计时长4分钟左右。

图 5-3 服务部署成功

步骤四：预测结果

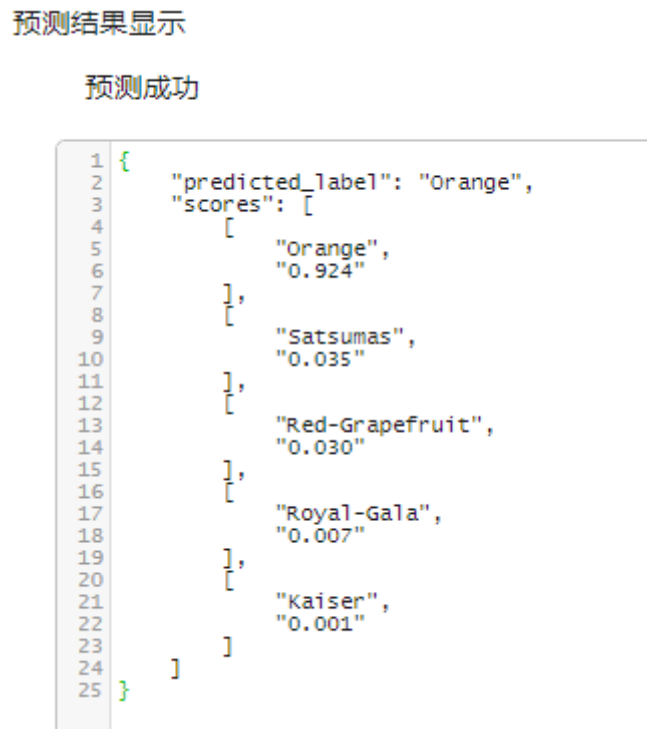
1. 在线服务部署完成后，单击“预测”页签。
2. 在“预测”页签，单击“上传”，上传一个测试图片，单击“预测”查看预测结果。此处提供一个样例图片供预测使用。

说明

本案例中使用的订阅模型可以识别**81类**常见超市商品，模型对预测图片有一定范围和要求，不满足条件的图片会影响预测结果的准确性。

图 5-4 预测样例图

图 5-5 预测结果



后续操作：清理资源

体验结束后，建议暂停或删除服务，避免占用资源，造成资源浪费。

- 停止在线服务：在“在线服务”列表，单击对应服务操作列的“更多 > 停止”。
- 删除在线服务：在“在线服务”列表，单击对应服务操作列的“更多 > 删除”。

常见问题

服务预测失败

6 ModelArts 入门实践

本章节列举了一些常用的实践案例，方便您快速了解并使用ModelArts完成AI开发。

表 6-1 常用最佳实践

分类	实践案例	描述	适用人群
ModelArts Standard模型训练	基于ModelArts Standard上运行GPU训练任务	本案例介绍了如何使用ModelArts Standard专属资源池提供的计算资源，结合SFS和OBS存储，在ModelArts Standard的训练环境中开展单机单卡、单机多卡、多机多卡分布式训练。	面向熟悉代码编写和调测的AI工程师，同时了解SFS和OBS云服务
	从 0 制作自定义镜像并用于训练（PyTorch+CPU/GPU）	本案例介绍如何从0开始制作镜像，并使用该镜像在ModelArts Standard平台上进行训练。镜像中使用的AI引擎是PyTorch，训练使用的资源是CPU或GPU。	面向熟悉代码编写和调测的AI工程师，同时熟悉docker容器知识
	从 0 制作自定义镜像并用于训练（MindSpore+Ascend）	本案例介绍如何从0开始制作镜像，并使用该镜像在ModelArts Standard平台上进行训练。镜像中使用的AI引擎是MindSpore，训练使用的资源是Ascend。	
	主流开源大模型基于Standard适配PyTorch NPU训练指导	本案例基于ModelArts Standard供的昇腾计算资源，指导用户完成Llama、Qwen、ChatGLM、Yi等常见开源大模型的预训练、SFT微调、LoRA微调训练过程。	面向熟悉代码编写和调测的AI工程师

分类	实践案例	描述	适用人群
ModelArts Standard 推理部署	使用Standard一键完成商超商品识别模型部署	本案例以“商超商品识别”模型为例，介绍从AI Gallery订阅模型，一键部署到ModelArts Standard，并进行在线推理预测的体验过程。	面向AI开发零基础的用户
	从0-1制作自定义镜像并创建AI应用	针对ModelArts不支持的AI引擎，您可以构建自定义镜像，并将镜像导入ModelArts，创建为模型。本案例详细介绍如何使用自定义镜像创建模型，并部署成在线服务。	面向熟悉代码编写和调测的AI工程师，同时熟悉docker容器知识
	主流开源大模型基于Standard适配PyTorch NPU推理指导	本案例基于ModelArts Standard供的昇腾计算资源，指导用户完成Llama、Qwen、ChatGLM、Yi等常见开源大模型的推理部署、模型评测、模型量化等功能。	面向熟悉代码编写和调测的AI工程师
ModelArts Standard 开发环境	使用ModelArts VS Code插件调试训练ResNet50图像分类模型	本案例以Ascend Model Zoo为例，介绍如何通过VS Code插件及ModelArts Standard的Notebook进行云端数据调试及模型开发。	面向熟悉代码编写和调测的AI工程师
ModelArts Lite Server	主流开源大模型基于Lite Server适配PyTorch NPU训练指导	本案例基于ModelArts Lite Server供的昇腾计算资源，指导用户完成Llama、Qwen、ChatGLM、Yi等常见开源大模型的预训练、SFT微调、LoRA微调训练过程。	面向熟悉代码编写和调测的AI工程师，同时熟悉Linux和Docker容器基础知识
	主流开源大模型基于Lite Server适配PyTorch NPU推理指导	本案例基于ModelArts Lite Server提供的昇腾计算资源，指导用户完成Llama、Qwen、ChatGLM、Yi等常见开源大模型的推理部署、模型评测、模型量化等功能。	