ALIBABA CLOUD

# Alibaba Cloud

## 机器学习PAI

## PAI-DSW Notebook建模

文档版本：20210607

阿里云

# 法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

# 通用约定

| 格式 | 说明 | 样例 |
| --- | --- | --- |
| 危险 | 该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 危险<br>重置操作将丢失用户配置数据。 |
| 警告 | 该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 警告<br>重启操作将导致业务中断，恢复业务时间约十分钟。 |
| 注意 | 用于警示信息、补充说明等，是用户必须了解的内容。 | 注意<br>权重设置为0，该服务器不会再接受新请求。 |
| 说明 | 用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。 | 说明<br>您也可以通过按Ctrl+A选中全部文件。 |
| > | 多级菜单递进。 | 单击**设置**> **网络**> **设置网络类型**。 |
| **粗体** | 表示按键、菜单、页面名称等UI元素。 | 在**结果确认**页面，单击**确定**。 |
| Courier字体 | 命令或代码。 | 执行 `cd /d C:/window` 命令，进入Windows系统文件夹。 |
| *斜体* | 表示参数、变量。 | `bae log list --instanceid`<br>*Instance_ID* |
| [] 或者 [a\|b] | 表示可选项，至多选择一个。 | `ipconfig [-all\|-t]` |
| {} 或者 {a\|b} | 表示必选项，至多选择一个。 | `switch {active\|stand}` |

# 目录

# 1.概述

PAI-DSW是一款云端机器学习开发IDE，为您提供交互式编程环境，适用于不同水平的开发者。本文介绍个人版、GPU特价版及探索者版的功能特点、实例规格及可用区。

PAI-DSW集成了开源JupyterLab，并以插件化的形式进行深度定制化开发。您无需任何运维配置，即可进行Notebook编写、调试及运行Python代码。同时，PAI-DSW提供丰富的计算资源，且对接多种数据源。通过EASCMD的方式，可以将PAI-DSW获得的训练模型部署为RESTful接口，对外提供模型服务，从而实现一站式机器学习。

## 功能特性

- 支持资源实时监控。算法开发时，可以显示CPU或GPU的使用情况。
- 支持多源数据接入，包括MaxCompute、OSS及NAS。
- 支持编写和运行SQL语句。
- 支持多种资源型号，包括纯CPU及多种GPU算力卡。
- 支持灵活切换各类资源，有效降低使用成本。
- 预置常用大数据开发包和算法库，且支持自定义安装第三方库。

## 版本介绍

PAI-DSW支持个人版、GPU特价版及探索者版，各版本的区别如下表所示。

| 功能 | 个人版 | GPU特价版 | 探索者版 |
| --- | --- | --- | --- |
| 支持GPU | 是 | 是 | 是 |
| 按量付费（后付费） | 是 | 否 | 免费使用 |
| 包年包月（预付费） | 否 | 是 | 免费使用 |
| 内存及CPU核数 | 自行选择，无上限。 | 自行选择，无上限。 | 2 vCPU+4 GB |
| 实例存储空间限制 | 自行选择，无上限。 | 自行选择，无上限。 | 5 GB |
| 网络访问 | 无限制 | 不能访问外网 | CPU型的无限制，GPU型的不能访问外网。 |
| Root权限 | 是 | 否 | 否 |
| 运行环境Image选择 | 是 | 否 | 否 |

## 个人版

个人版由原V2专业版改进，基于阿里云Docker和Kubernetes等云原生技术，为您提供灵活且开放的AI开发环境。该版本的功能特点、实例规格及可用区、支持的镜像列表如下：

- 版本特点
  - 对比原V2专业版，实例的创建时间降低了65%。同时，不额外收取EIP和SLB费用。
  - 支持实例随时停止和启动、镜像一键保存及开发环境恢复。

- 提供集成式AI开发环境：
  - 预装常用大数据开发包和算法包，且开放Sudo权限，从而允许安装第三方库。
  - 预装JupyterLab插件，可以提高开发效率。例如Git及TensorBoard。
  - 提供多种官方镜像，可以覆盖多版本主流计算框架。例如TensorFlow及PyTorch。
  - 嵌入WebIDE，可以安装任意插件。
- 预置PAI的基础能力，包括视觉类算法工具EasyVision、自动调参工具AutoML、编译优化及读取MaxCompute表的CommonIO组件。
- 支持Root权限。

- 实例规格及可用区

  CPU类型的实例规格及可用区如下表所示。

| 规格 | vCPU数量 | 内存（GiB） | 带宽（Gbps） | 系统盘（GB） | 地域 |
| --- | --- | --- | --- | --- | --- |
| ecs.c6.large | 2 | 4 | 1 | 128 | ○ 华北2（北京）<br>○ 华东2（上海）<br>○ 华东1（杭州）<br>○ 华南1（深圳） |
| ecs.g6.large | 2 | 8 | 1 | 128 | |
| ecs.g6.xlarge | 4 | 16 | 1,5 | 256 | |
| ecs.g6.2xlarge | 8 | 32 | 2.5 | 500 | |
| ecs.g6.4xlarge | 16 | 64 | 5 | 500 | |
| ecs.g6.8xlarge | 32 | 128 | 10 | 500 | |

  GPU类型的实例规格及可用区如下表所示。

| 规格名称 | vCPU数量 | 内存（GiB） | GPU | 带宽（Gbps） | 系统盘（GB） | 地域 |
| --- | --- | --- | --- | --- | --- | --- |
| ecs.gn5-c4g1.xlarge | 4 vCPU | 30 GiB | 1 * NVIDIA P100 | 3 Gbps | 256 G | ○ 华北2（北京）<br>○ 华东2（上海）<br>○ 华东1（杭州）<br>○ 华南1（深圳） |
| ecs.gn6v-c8g1.2xlarge | 8 vCPU | 32 GiB | 1 * NVIDIA V100 | 2.5 Gbps | 500 G | |
| ecs.gn5-c8g1.2xlarge | 8 vCPU | 60 GiB | 1 * NVIDIA P100 | 3 Gbps | 500 G | |
| ecs.gn5-c8g1.4xlarge | 16 vCPU | 120 GiB | 2 * NVIDIA P100 | 5 Gbps | 500 G | |
| ecs.gn5-c28g1.7xlarge | 28 vCPU | 112 GiB | 1 * NVIDIA P100 | 5 Gbps | 500 G | |

| 规格名称 | vCPU数量 | 内存（GiB） | GPU | 带宽（Gbps） | 系统盘（GB） | 地域 |
| --- | --- | --- | --- | --- | --- | --- |
| ecs.gn6v-c8g1.8xlarge | 32 vCPU | 128 GiB | 4 * NVIDIA V100 | 10 Gbps | 500 G | |
| ecs.gn6e-c12g1.12xlarge | 48 vCPU | 368 GiB | 4 * NVIDIA V100 | 16 Gbps | 500 G | |

- 镜像列表

| 镜像名称 | 描述 |
| --- | --- |
| py27_cuda90_tf1.12_ubuntu | 支持TensorFlow 1.12（GPU）版本 |
| py27_cpu_tf1.12_ubuntu | 支持TensorFlow 1.12（CPU）版本 |
| py36_cuda101_tf2.1_torch1.4_ubuntu | 支持TensorFlow 2.1和PyTorch 1.4（GPU）版本 |
| py36_cpu_tf2.1_torch1.4_ubuntu | 支持TensorFlow 2.1和PyTorch 1.4（CPU）版本 |
| py36_cuda100_paitf1.12_alios | 支持PAI-TensorFlow 1.12（GPU）版本 |
| py36_cpu_paitf1.12_alios | 支持PAI-TensorFlow 1.12（CPU）版本 |
| py36_cuda100_tf1.15_ubuntu | 支持TensorFlow 1.15（GPU）版本 |
| py36_cpu_tf1.15_ubuntu | 支持TensorFlow 1.15（CPU）版本 |

## GPU特价版

GPU特价版由原V1入门版改进，基于阿里云飞天大数据平台构建，大幅优化运行成本。但是该版本不支持访问外网和Root权限，无法进行Sudo操作，请谨慎选择。GPU特价版的功能特点、实例规格及可用区如下：

- 版本特点
  - 支持资源实时监控。算法开发时，可以显示CPU或GPU的使用情况。
  - 支持多源数据接入，包括MaxCompute、OSS及NAS。
  - 支持编写和运行SQL语句。
  - 支持多种资源型号，包括纯CPU及多种GPU算力卡。
  - 支持灵活切换各类资源，有效降低使用成本。
  - 预置常用大数据开发包和算法库，且支持自定义安装第三方库。
- 实例规格及可用区

| 规格 | 地域 |
| --- | --- |
| P100 | 华北2（北京） |
| M40 | 华东2（上海） |

## 探索者版

索者版由原V2专业版改进，基于阿里云Docker和Kubernetes等云原生技术，为您提供灵活且开放的AI开发环境。探索者版处于免费状态，登录探索者版即可使用，详细的使用文档请参见天池实验室DSW新手使用手册。

# 2.授权

本文为您介绍如何为RAM用户和PAI-DSW关联角色授权。

## 背景信息

授权策略语言的结构和语法请参见权限策略语法和结构。

## 主账号授权PAI-DSW服务角色

为了确保PAI-DSW能够正常提供服务，需要确认当前阿里云主账号拥有AliyunPAIDSWDefaultRole这一服务角色。

1. 登录RAM控制台。
2. 在左侧导航栏，单击**RAM角色管理**。
3. 在**RAM角色管理**页面的搜索中，输入AliyunPAIDSWDefaultRole，进行搜索。
   - 如果搜索到了该角色，则表示已经授权了PAI-DSW服务角色。
   - 如果没有搜索到该角色，则进行授权（授权）。

## RAM用户授权

阿里云主账号可以授权RAM用户管理PAI-DSW实例，包括创建、启动、停止及删除实例。

1. 登录RAM控制台。
2. 创建自定义权限策略。
   i. 在左侧导航栏，选择**权限管理 > 权限管理策略**。
   ii. 在**权限管理策略**页面，单击**创建权限策略**。
   iii. 在**新建自定义权限策略**页面，配置参数。

| 参数 | 描述 |
| --- | --- |
| **策略名称** | 输入DSW_Notebook_Access。 |
| **备注** | 输入PAI-DSW访问策略。 |
| **配置模式** | 单击**脚本配置**。 |

| 参数 | 描述 |
| --- | --- |
| 策略内容 | 将**策略内容**修改为如下内容。<br>`{ "Statement": [ { "Action": [ "notebook:CreateInstance", "notebook:StartInstance", "notebook:StopInstance", "notebook:EditInstance", "notebook:ListInstance" ], "Effect": "Allow", "Resource": "*" } ], "Version": "1" }`<br>其中Action表示赋予的操作权限，可以包括以下权限：<br>■ notebook:CreateInstance：创建PAI-DSW实例。<br>■ notebook:StartInstance：开启PAI-DSW实例。<br>■ notebook:StopInstance：停止PAI-DSW实例。<br>■ notebook:EditInstance：编辑PAI-DSW实例。<br>■ notebook:ListInstance：查看所有PAI-DSW实例。<br>Resource表示资源权限，配置方式包括：<br>■ 指定实例的地域权限<br>`"Resource": "acs:notebook:cn-beijing:*:notebook/*"`<br>■ 为特定实例（例如hhdemo）赋予PAI-DSW的使用权限<br>`"Resource": "acs:notebook:*:*:notebook/hhdemo"`<br>■ 为所有实例赋予PAI-DSW的使用权限<br>`"Resource": "*"`<br>如果需要配置更多权限，请参见权限策略基本元素。 |

iv. 单击**确定**。

3. 为RAM用户授权。
   i. 在左侧导航栏，选择**人员管理 > 用户**。
   ii. 在**用户**页面，单击**操作**列下的**添加权限**。
   iii. （可选）在**添加权限**面板，单击**自定义策略**。
   iv. 在**选择权限**下的文本框，输入**DSW_Notebook_Access**。
   v. 单击**权限策略名称**下的**DSW_Notebook_Access**，使其显示在**已选择**列表中。
   vi. 单击**确定**。

## 为关联角色授权

首次使用PAI-DSW，需要对相关资源进行访问授权。

1. 进入**Notebook建模服务**页面。
   i. 登录Machine Learning Platform for AI console。
   ii. 在PAI控制台首页，选择**模型开发和训练 > DSW-Notebook建模**。
2. 单击**创建实例**。
3. 在**角色授权**对话框，单击**去授权**。
4. 在**云资源访问授权**页面，单击**同意授权**。在**云资源访问授权**页面，系统自动配置PAI-DSW需要的关联角色，无需手动配置。

# 3.使用说明

## 3.1. 创建实例

使用PAI-DSW进行Notebook建模前，您需要创建PAI-DSW实例。本文为您介绍如何创建PAI-DSW实例。

### 前提条件

首次使用PAI-DSW，需要对相关资源进行访问授权，详情请参见为关联角色授权。

如果RAM用户创建实例，则需要主账号为其授权，详情请参见RAM用户授权。

### 操作步骤

1. 登录PAI控制台。
2. 在左侧导航栏，选择**模型开发和训练 > 交互式建模（DSW）**。
3. 在页面左上方，选择目标地域。
4. 在**Notebook建模服务**页面，单击**创建实例**。
5. 在**配置实例**向导页面，配置参数。

| 参数 | 描述 |
| --- | --- |
| **实例名称** | 只能包含英文字母、数字及下划线（_），长度不能超过27个字符。 |
| **实例版本** | 支持以下版本：<br>○ **DSW个人版**：由原V2专业版改进，基于阿里云Docker和Kubernetes等云原生技术，为您提供灵活且开放的AI开发环境。<br>○ **DSW特价版**：由原V1入门版改进，基于阿里云飞天大数据平台构建，大幅优化运行成本。但是该版本不支持访问外网和Root权限，无法sudo操作，请谨慎选择。 |
| **地域及可用区** | 在不同地域的实例之间，网络不相通。选择距离较近的地域，可以降低网络时延，从而提高访问速度。 |
| **付费模式** | **DSW个人版**仅支持**按量付费**模式，**DSW特价版**仅支持**包年包月**模式。 |
| **网络配置** | 仅**DSW特价版**需要选择。默认为**经典网络**，不支持修改。 |
| **实例资源** | 仅**DSW个人版**需要选择。支持CPU实例和GPU实例，详细的实例规格请参见个人版。 |
| **资源类型** | 仅**DSW特价版**需要选择。如果剩余的GPU卡数无法满足需求，可以单击**资源类型**后的**购买页面**进行购买。 |

| 参数 | 描述 |
| --- | --- |
| 存储 | 实例自带系统盘存储为临时存储，停止或删除实例后，该存储清空。如果需要永久化存储，需要选择已创建的NAS文件系统进行挂载。创建NAS文件系统请参见创建文件系统。<br>说明 一旦挂载NAS文件系统，PAI-DSW将默认使用该NAS存储数据，不再使用临时存储。 |
| 实例镜像 | 支持的镜像覆盖Python、TensorFlow及PyTorch多个版本，详细的镜像列表请参见个人版。 |
| 专有网络 | 仅DSW个人版需要选择，支持用户在VPC内使用PAI-DSW。您必须同时配置专有网络、交换机及安全组。<br>您可以直接选择已经创建的专有网络进行挂载，或单击专有网络后的创建专有网络进行创建。 |
| 交换机 | 如果配置了专有网络，则必须同时配置交换机和安全组。<br>您可以直接选择已经创建的交换机和安全组进行挂载，或单击创建交换机和创建安全组进行创建。 |
| 安全组 | |

6. 单击**确认订单**。
7. 核对订单信息，选中**《机器学习PAI DSW服务条款》**复选框，并单击**创建实例**。

# 3.2. 管理实例

本文为您介绍如何管理PAI-DSW的实例，包括停止、启动及删除实例。

## 前提条件

- 如果使用RAM用户管理实例，则需要主账号对其授权，详情请参见授权。
- 创建PAI-DSW实例，详情请参见创建实例。

## 进入交互式建模（DSW）

1. 登录PAI控制台。
2. 在左侧导航栏，选择**模型开发和训练 > 交互式建模（DSW）**。

## 停止实例

1. 进入**交互式建模（DSW）**。
2. 在**Notebook建模服务**页面，单击待停止实例**操作**列下的**停止**。
3. 在执行停止操作的确认对话框，选择停止方式。PAI-DSW实例支持两种停止方式：
   - **保存环境然后停止**：如果您对默认环境进行了修改（例如安装了软件包或pip包），建议选择该方式。
   - **直接停止**：如果未修改默认环境，通常选择**直接停止**。

   实例停止后，其**状态**变为**停止**，此时对于后付费实例，系统停止计费。退出PAI-DSW时，确保您的实例

处于**停止**状态，否则可能产生不必要的费用。

### 启动实例

如果实例处于停止状态，则可以手动启动。

1. 进入**交互式建模（DSW）**。
2. 在**Notebook建模服务**页面，单击待启动实例**操作**列下的**启动**。
   实例启动后，其**状态**变为**运行中**，此时对于后付费实例，系统开始计费。完成训练后，建议您及时停止实例，以免产生不必要的费用。

### 删除实例

如果不再进行训练，则可以删除实例。实例删除后，其数据无法恢复。

1. 进入**交互式建模（DSW）**。
2. 在**Notebook建模服务**页面，单击待删除实例**操作**列下的**删除**。
3. 在**删除**对话框，单击**确定**。

## 3.3. 使用开发环境

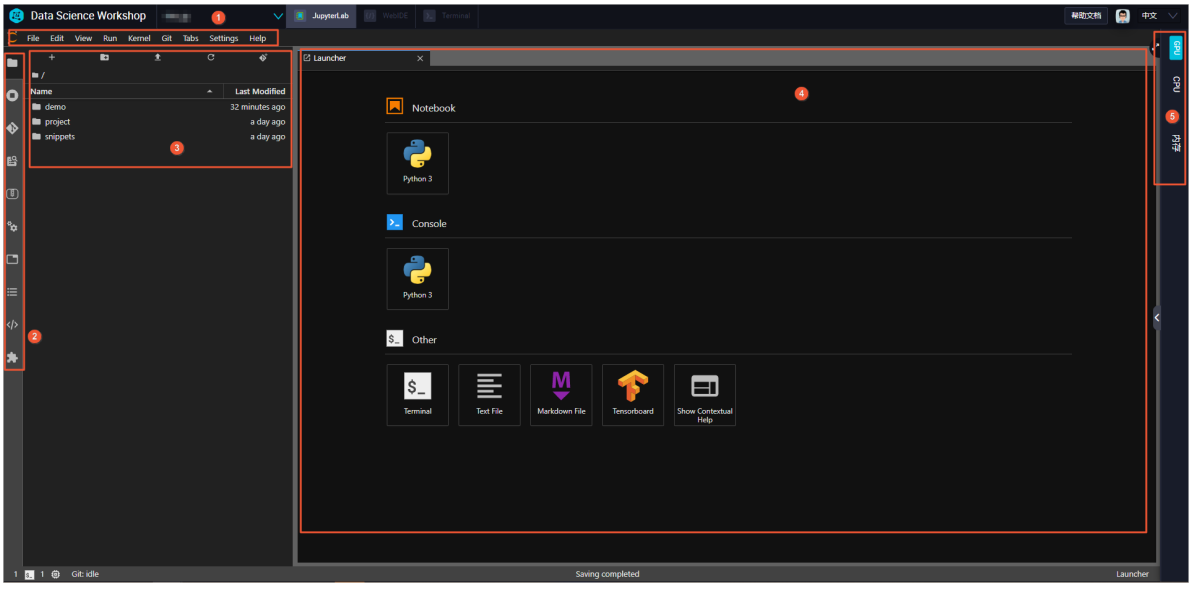
本文为您介绍如何使用PAI-DSW的开发环境，包括界面介绍、使用Notebook预置案例、上传数据及管理第三方库。

### 前提条件

创建PAI-DSW实例，详情请参见创建实例。

### 界面介绍

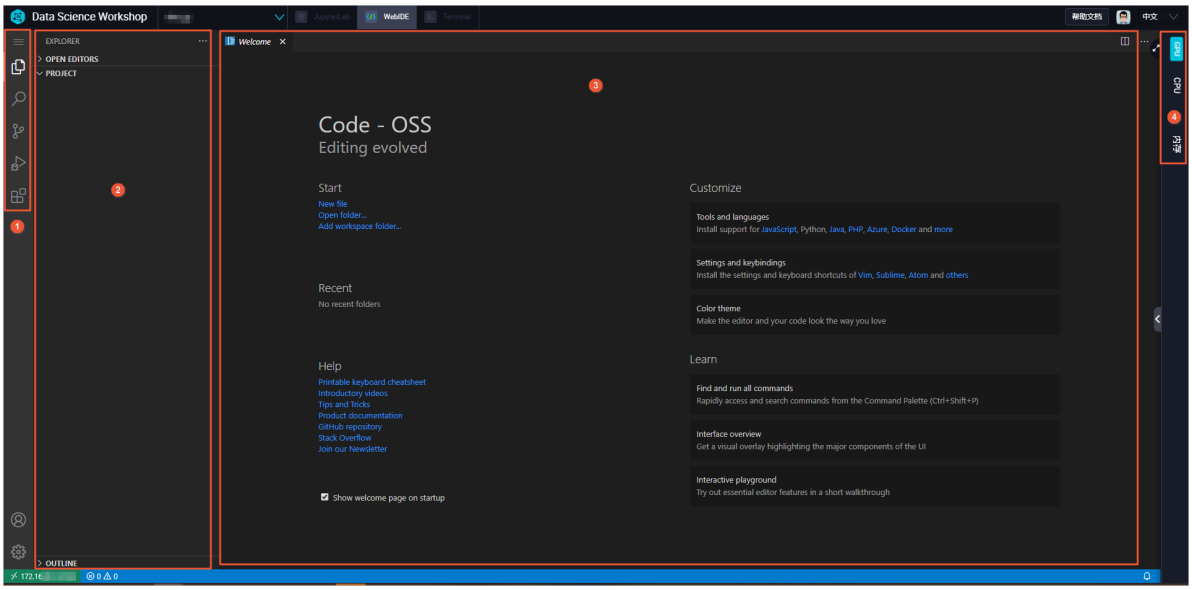
- Jupyterlab




| 功能区编号 | 描述 |
| --- | --- |
| ① | 顶部菜单栏 |

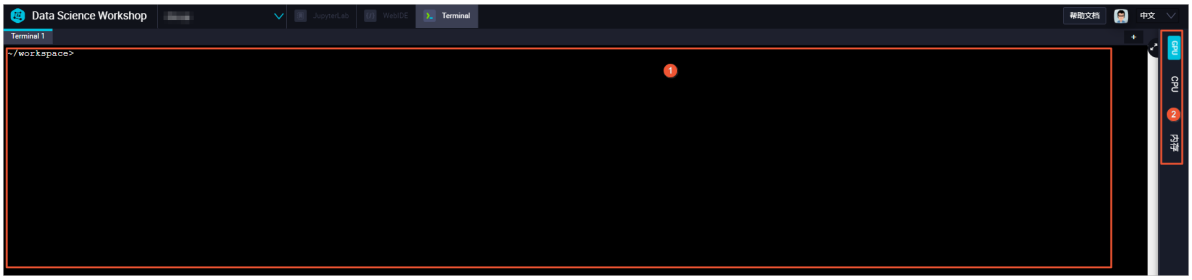
| 功能区编号 | 描述 |
| --- | --- |
| ② | 左侧工具栏 |
| ③ | 工具内容 |
| ④ | 主工作区 |
| ⑤ | 资源水位 |

- WebIDE




| 功能区编号 | 描述 |
| --- | --- |
| ① | 左侧工具栏 |
| ② | 工具内容 |
| ③ | 主工作区 |
| ④ | 资源水位 |

- Terminal



| 功能区编号 | 描述 |
| --- | --- |
| ① | 主工作区 |
| ② | 资源水位 |

## 使用预置的Notebook案例

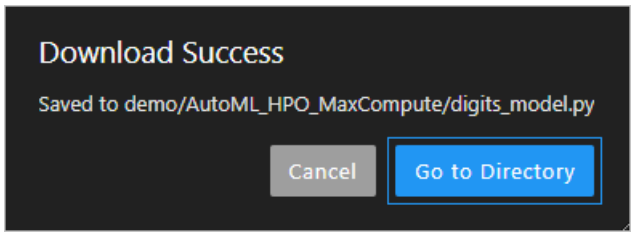
如果您初次使用PAI-DSW，推荐通过预置案例，熟悉产品功能。

1. 进入PAI-DSW开发环境。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型开发和训练 > 交互式建模（DSW）**。
   iii. 在页面左上方，选择使用服务的地域。
   iv. （可选）在**Notebook建模服务**页面的搜索框，输入实例名称或关键字，搜索实例。
   v. 单击需要打开的实例**操作**列下的**打开**。
2. 下载预置案例。
   i. 在PAI-DSW开发环境左侧的工具栏，单击图标。
   ii. 单击待下载案例（例如**AutoML_HPO_101**）后的图标。


3. 打开案例文件（以**AutoML_HPO_101**为例）。

i. 进入案例存储路径。下载完成的**AutoML_HPO_101**案例存储在*/demo/AutoML_HPO_101/*下，您可以通过以下任何一种方式进入案例存储路径：

- 下载案例后，在**Download Success**对话框中，单击**Go to Directory**。




- 通过PAI-DSW开发环境左侧的工具栏进入：

a. 在PAI-DSW开发环境左侧的工具栏，单击图标。

b. 双击**Name**列下的**demo**。

c. 双击**Name**列下的**AutoML_HPO_101**。

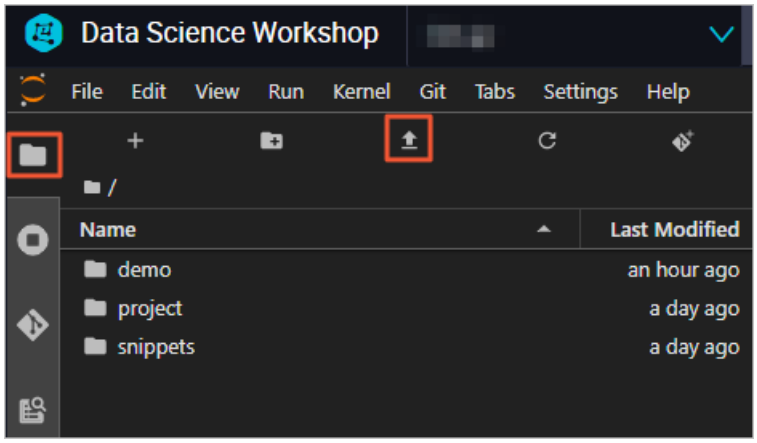
ii. 在存储路径下，双击**Name**列下的**AutoML_HPO_101.ipynb**，打开案例文件。

4. 在**AutoML_HPO_101.ipynb**文件中，可以查看案例的使用原理，您可以根据该信息执行任务。

## 上传数据

1. 在PAI-DSW Jupyterlab Notebook编程环境中，单击左侧工具栏中的图标。

2. 单击快捷工具栏中的图标，即可上传文件，且支持断点续传。




## 管理第三方库

如果使用Python开发环境，您可以在Terminal中，对第三方库进行以下操作：

- 安装

```
pip install --user <yourLibraryName>
```

需要将 `<yourLibraryName>` 替换为待安装的第三方库名称。例如，使用 `pip install --user sklearn` 命令，安装sklearn库。

- 查看

```
pip list
```

查看所有已安装的第三方库。

- 卸载

```
pip uninstall <yourLibraryName>
```

需要将 <yourLibraryName> 替换为已安装的第三方库名称。

> **说明** 只能卸载自己安装的第三方库。

因为tensoflow-gpu不支持卸载，所以只能使用更新命令安装固定版本的tensoflow-gpu，且新版本必须与CUDA版本（预付费实例的CUDA版本为10，后付费实例的CUDA版本为9）兼容。

```
pip install --upgrade --user tensorflow-gpu=<versionNumber>
```

需要将 <versionNumber> 替换为待安装的tensoflow-gpu版本号。

> **注意** 不要升级系统pip，否则可能导致无法安装。

PAI-DSW提供的开发环境包括Python2、Python3、PyTorch及TensorFlow2.0。安装第三方库时，默认安装至Python3，如果需要安装至其他环境，则必须手动切换环境后，再进行安装。

```
#安装至Python2环境。
source activate python2
pip install --user <yourLibraryName>
#安装至TensorFlow2.0环境。
source activate tf2
pip install --user <yourLibraryName>
```

需要将 <yourLibraryName> 替换为待安装的第三方库名称。

# 3.4. 读写OSS数据

本文为您介绍如何使用OSS Python SDK、TensorFlow OSS IO及PyTorch OSS API读写OSS数据。

## 背景信息

对象存储OSS（Object Storage Service）是阿里云提供的海量、安全、低成本及高可靠性的云存储服务。PAI-DSW不仅预置NAS文件系统，而且对接OSS存储。

## OSS Python SDK

通常，您可以直接使用OSS的Python API读写OSS中的数据，详情请参见。PAI-DSW已预装OSS2 Python包，您可以参见如下方法读写OSS数据。

1. 鉴权及初始化。

```
import oss2
auth = oss2.Auth('<your_AccessKey_ID>', '<your_AccessKey_Secret>')
bucket = oss2.Bucket(auth, 'http://oss-cn-beijing-internal.aliyuncs.com', '<your_bucket_name>')
```

需要根据实际需要修改以下参数。

| 参数 | 描述 |
| --- | --- |
| <your_AccessKey_ID> | 阿里云的AccessKey ID。 |
| <your_AccessKey_Secret> | 阿里云的AccessKey Secret。 |
| http://oss-cn-beijing-internal.aliyuncs.com | OSS域名。需要根据实例的地域选择对应的OSS域名：<br>○ 华北2（北京）后付费实例：oss-cn-beijing.aliyuncs.com<br>○ 华北2（北京）预付费实例：oss-cn-beijing-internal.aliyuncs.com<br>○ 华东2（上海）GPU P100实例或CPU实例：oss-cn-shanghai.aliyuncs.com<br>○ 华东2（上海）GPU M40实例：oss-cn-shanghai-internal.aliyuncs.com |
| <your_bucket_name> | Bucket名称，且开头不带oss://。 |

2. 读写OSS数据。

```
#读取一个完整文件。
result = bucket.get_object('<your_file_path/your_file>')
print(result.read())
#按Range读取数据。
result = bucket.get_object('<your_file_path/your_file>', byte_range=(0, 99))
#写数据至OSS。
bucket.put_object('<your_file_path/your_file>', '<your_object_content>')
#对文件进行Append。
result = bucket.append_object('<your_file_path/your_file>', 0, '<your_object_content>')
result = bucket.append_object('<your_file_path/your_file>', result.next_position, '<your_object_content>')
```

其中 <your_file_path/your_file> 表示待读写的文件路径， <your_object_content> 表示待Append的内容，需要根据实际情况修改。

## TensorFlow OSS IO

PAI-DSW提供tensorflow_io模块，TensorFlow用户可以使用其直接读取OSS数据。在执行TensorFlow训练任务的过程中，无需频繁拷贝数据文件或模型文件。

1. 导入tensorflow_io包，并拼接OSS Bucket URL。

```
import tensorflow as tf
import tensorflow_io
access_id="<your_Access_Key_ID>"
access_key="<your_Access_Key_Secret>"
host = "oss-cn-beijing-internal.aliyuncs.com"
bucket="oss://<your_bucket_name>"
oss_bucket_root="{}\x01id={}\x02key={}\x02host={}/".format(bucket, access_id, access_key, host)
```

需要根据实际需要修改以下参数。

| 参数 | 描述 |
| --- | --- |
| <your_AccessKey_ID> | 阿里云的AccessKey ID。 |
| <your_AccessKey_Secret> | 阿里云的AccessKey Secret。 |

| 参数 | 描述 |
| --- | --- |
| oss-cn-beijing-internal.aliyuncs.com | OSS域名。需要根据实例的地域选择对应的OSS域名：<br>○ 华北2（北京）后付费实例：oss-cn-beijing.aliyuncs.com<br>○ 华北2（北京）预付费实例：oss-cn-beijing-internal.aliyuncs.com<br>○ 华东2（上海）GPU P100实例或CPU实例：oss-cn-shanghai.aliyuncs.com<br>○ 华东2（上海）GPU M40实例：oss-cn-shanghai-internal.aliyuncs.com |
| <your_bucket_name> | Bucket名称，且开头不带oss://。 |

2. 您可以使用以下任何一种方式读取OSS数据（以TensorFlow 1.0 API为例）：
   - 使用 `GFile` 读写OSS文本文件。

     ```
     oss_file = oss_bucket_root + "test.txt"
     with tf.gfile.GFile(oss_file, "w") as f:
      f.write("<your_context>")
     with tf.gfile.GFile(oss_file, "r") as f:
      print(f.read())
     ```

     其中 `<your_context>` 表示写入的内容，需要根据实际情况修改。
   - 使用 `TextLineDataset` 读取OSS数据。

     ```
     #Test textline reader op.
     oss_file = oss_bucket_root + "test.txt"
     dataset = tf.data.TextLineDataset([oss_file])
     iterator = dataset.make_initializable_iterator()
     a = iterator.get_next()
     with tf.Session() as sess:
       tf.global_variables_initializer().run()
       sess.run(iterator.initializer)
       print(sess.run(a))
     ```

## OSS Python API

对于PyTorch用户，PAI-DSW提供OSS Python API，用于直接读写OSS数据。

您可以在OSS存储训练数据、日志或模型：

- 加载训练数据

  您可以将数据存放在一个OSS Bucket中，且将数据路径和对应的Label存储在同一个OSS Bucket的索引文件中。通过自定义DataSet，在PyTorch中使用 `DataLoader` API多进程并行读取数据，示例如下。

```
import io
import oss2
import PIL
import torch
class OSSDataset(torch.utils.data.dataset.Dataset):
  def __init__(self, endpoint, bucket, auth, index_file):
    self._bucket = oss2.Bucket(auth, endpoint, bucket)
    self._indices = self._bucket.get_object(index_file).read().split(',')
  def __len__(self):
    return len(self._indices)
  def __getitem__(self, index):
    img_path, label = self._indices(index).strip().split(':')
    img_str = self._bucket.get_object(img_path)
    img_buf = io.BytesIO()
    img_buf.write(img_str.read())
    img_buf.seek(0)
    img = Image.open(img_buf).convert('RGB')
    img_buf.close()
    return img, label
dataset = OSSDataset(endpoint, bucket, index_file)
data_loader = torch.utils.data.DataLoader(
  dataset,
  batch_size=batch_size,
  num_workers=num_loaders,
  pin_memory=True)
```

其中 `endpoint` 为OSS域名， `bucket` 为Bucket名称， `auth` 为鉴权对象， `index_file` 为索引文件的路径，都需要根据实际情况修改。

> **说明** 示例中，索引文件格式为每条样本使用英文逗号（,）分隔，样本路径与Label之间使用英文冒号（:）分隔。

- 写日志

  您可以编写一个StreamHandler，用于封装Logging日志的输出。

  > **说明** 多个进程不能同时写一个日志文件。

```
import oss2
import logging
class OSSLoggingHandler(logging.StreamHandler):
  def __init__(self, endpoint, bucket, auth, log_file):
    OSSLoggingHandler.__init__(self)
    self._bucket = oss2.Bucket(auth, endpoint, bucket)
    self._log_file = log_file
    self._pos = self._bucket.append_object(self._log_file, 0, '')
  def emit(self, record):
    msg = self.format(record)
    self._pos = self._bucket.append_object(self._log_file, self._pos.next_position, msg)
oss_handler = OSSLoggingHandler(endpoint, bucket, log_file)
logging.basicConfig(
  stream=oss_handler,
  format='[%(asctime)s] [%(levelname)s] [%(process)d#%(threadName)s] ' +
      '[%(filename)s:%(lineno)d] %(message)s',
  level=logging.INFO)
```

其中 `endpoint` 为OSS域名， `bucket` 为OSS Bucket名称， `auth` 为鉴权对象， `log_file` 为日志文件路径，都需要根据实际情况修改。

- Save或Load模型

  您可以使用OSS2 Python API Save或Load PyTorch模型（关于PyTorch如何Save或Load模型，详情请参见PyTorch），示例如下：

  - Save模型

    ```
    from io import BytesIO
    import torch
    import oss2
    # bucket_name
    bucket_name = "<your_bucket_name>"
    bucket = oss2.Bucket(auth, endpoint, bucket_name)
    buffer = BytesIO()
    torch.save(model.state_dict(), buffer)
    bucket.put_object("<your_model_path>", buffer.getvalue())
    ```

    其中 `endpoint` 为OSS域名， `<vour_bucket_name>` 为OSS Bucket名称，且开头不带oss://， `auth` 为鉴权对象， `<your_model_path>` 为模型路径，都需要根据实际情况修改。

  - Load模型

    ```
    from io import BytesIO
    import torch
    import oss2
    bucket_name = "<your_bucket_name>"
    bucket = oss2.Bucket(auth, endpoint, bucket_name)
    buffer = BytesIO(bucket.get_object("<your_model_path>").read())
    model.load_state_dict(torch.load(buffer))
    ```

    其中 `endpoint` 为OSS域名， `<vour_bucket_name>` 为OSS Bucket名称，且开头不带oss://， `auth` 为鉴权对象， `<your_model_path>` 为模型路径，都需要根据实际情况修改。

# 3.5. 读写MaxCompute表

本文介绍如何使用PyODPS读写MaxCompute表数据。

## PyODPS

您可以使用PyODPS与MaxCompute或PAI-Studio中的数据进行通信。PyODPS是阿里云提供的Python SDK，详情请参见PyODPS开发文档。

1. 安装PyODPS。在PAI-DSW的Terminal中，执行如下命令。

```
pip install --user pyodps
```

2. 读取MaxCompute表数据（以MaxCompute特定项目下的某表前10行数据为例）。

```
from odps import ODPS
from odps.df import DataFrame
o = ODPS('<your_AccessKey_ID>', '<your_AccessKey_Secret>',project='<your_MaxCompute_project>',
endpoint='http://service-all.ext.odps.aliyun-inc.com/api')
users = DataFrame(o.get_table('<your_table_name>'))
print(users.head(10))
```

需要根据实际情况修改以下参数。

| 参数 | 描述 |
| --- | --- |
| <your_AccessKey_ID> | 阿里云的AccessKey ID。 |
| <your_AccessKey_Secret> | 阿里云的AccessKey Secret。 |
| <your_MaxCompute_project> | MaxCompute项目名称。 |
| http://service-all.ext.odps.aliyun-inc.com/api | 华东2（上海）GPU M40实例和华北2（北京）预付费P100实例的Endpoint。其他地域的Endpoint为http://service.cn.maxcompute.aliyun.com/api。 |
| <your_table_name> | MaxCompute表名。 |

# 3.6. 导出Notebook文件

您可以将Notebook文件导出为多种形式，以便本地查看或分享。

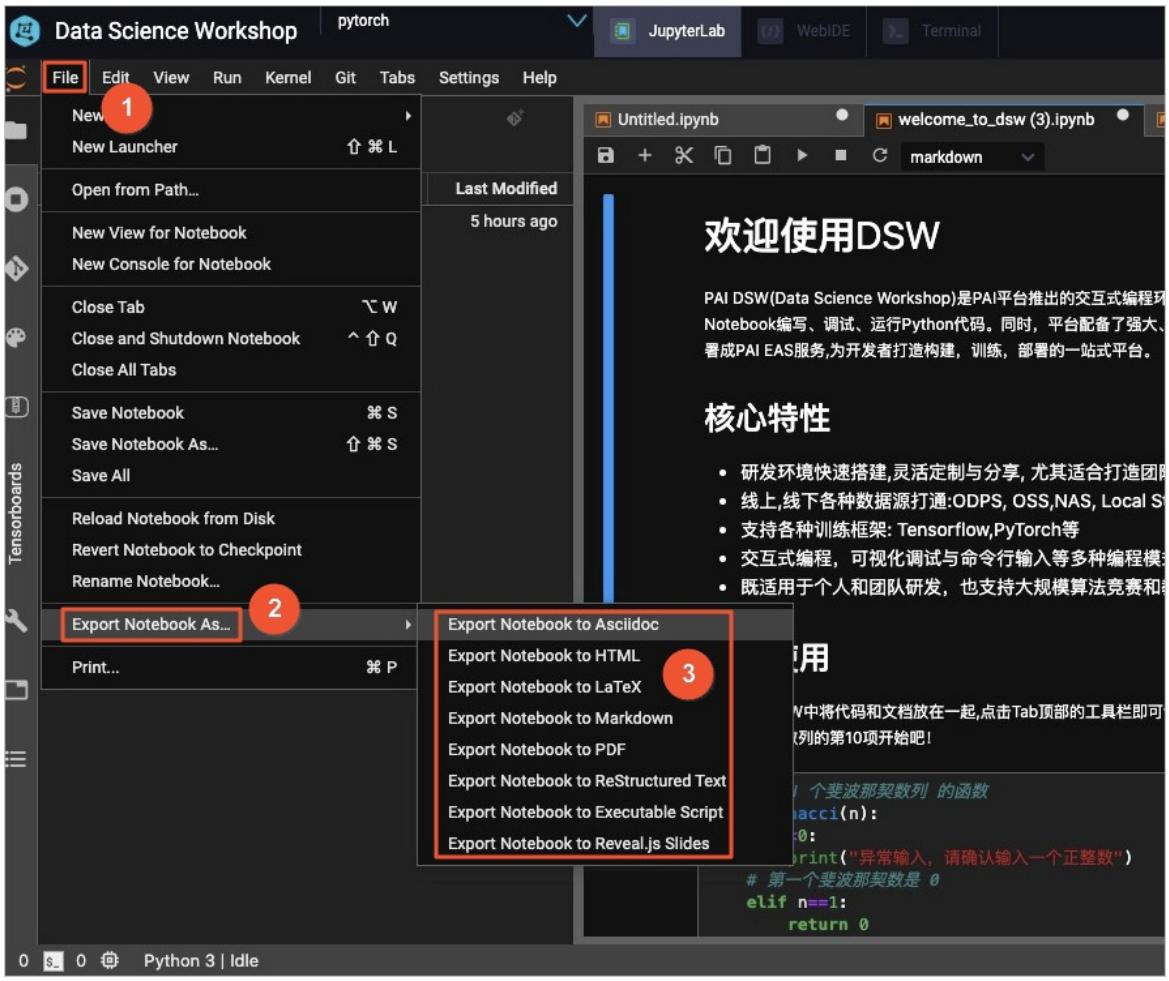
## 背景信息

PAI-DSW支持将Notebook文件导出为以下几种形式：

- Asciidoc：.asciidoc文件
- HTML：.html文件
- Latex：.tex文件
- Markdown：.md文件
- PDF：.pdf文件
- ReStructured Text：.rst文件
- Executable Script：.py文件
- Reveal.js Slides：.html文件

## 操作步骤

1. 进入PAI-DSW开发环境。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型开发和训练 > DSW-Notebook建模**。
   iii. 在页面左上方，选择使用服务的地域。
   iv. 在**Notebook建模服务**页面，单击待打开的实例**操作**列下的**打开**。
2. 打开待导出的Notebook文件。
3. 在顶部菜单栏，选择**File > Export Notebook As... > 目标格式**，即可将该文件导出为所选的**目标格式**文件。




> **说明** 导出的可选目标格式取决于nbconvert配置，详情请参见nbconvert官方文档。

# 4.实践案例

## 4.1. 使用WebIDE在线调试代码

在代码开发过程中经常会遇到某段代码的运行结果与预期不符的情况，此时，除多遍阅读代码以外，增加调试级别的日志，并使用在线调试实时查看各变量值和流程跳转，可以帮助您快速定位问题。本文以排查PAI-DSW中提供的Sample Notebook问题为例，介绍如何通过PAI-DSW中的WebIDE，在线调试Notebook中运行的Python代码。

### 背景信息

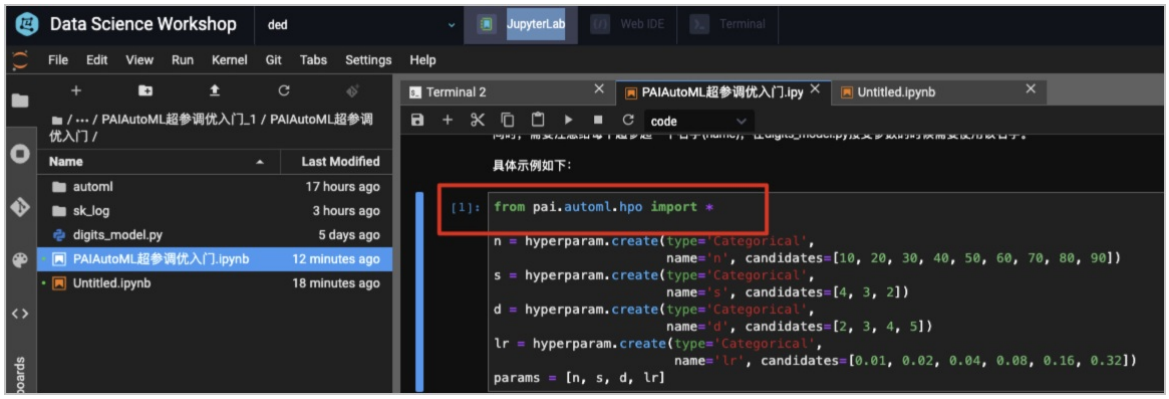
PAI-DSW官方Demo提供了几个PAI AutoML超参调优示例，其中“AutoML超参调优入门”示例在运行到 `tuner.fit()` 时经常卡顿，Notebook单元格输出日志如下图所示。

```
[*]: your_job_id = tuner.fit()
     your_job_id

     2020-04-18,04:03:58.306 [INFO] Missing algorithm, using default setting
     "initial_trials_num": 4, "stop_when_exception": true}}
     2020-04-18,04:03:58.307 [INFO] Missing earlystop, using default setting

[*]: tuner.status()

[*]: tuner.info()

[*]: tuner.log()

[*]: # tuner.kill(your_job_id)

[*]: row = tuner.get_best_config()
     print(row['Hyperparams'])
```

从日志中直接找到问题根源比较困难，可以先查看tuner的实现代码，再进行代码调试。查看tuner实现代码的方法如下：

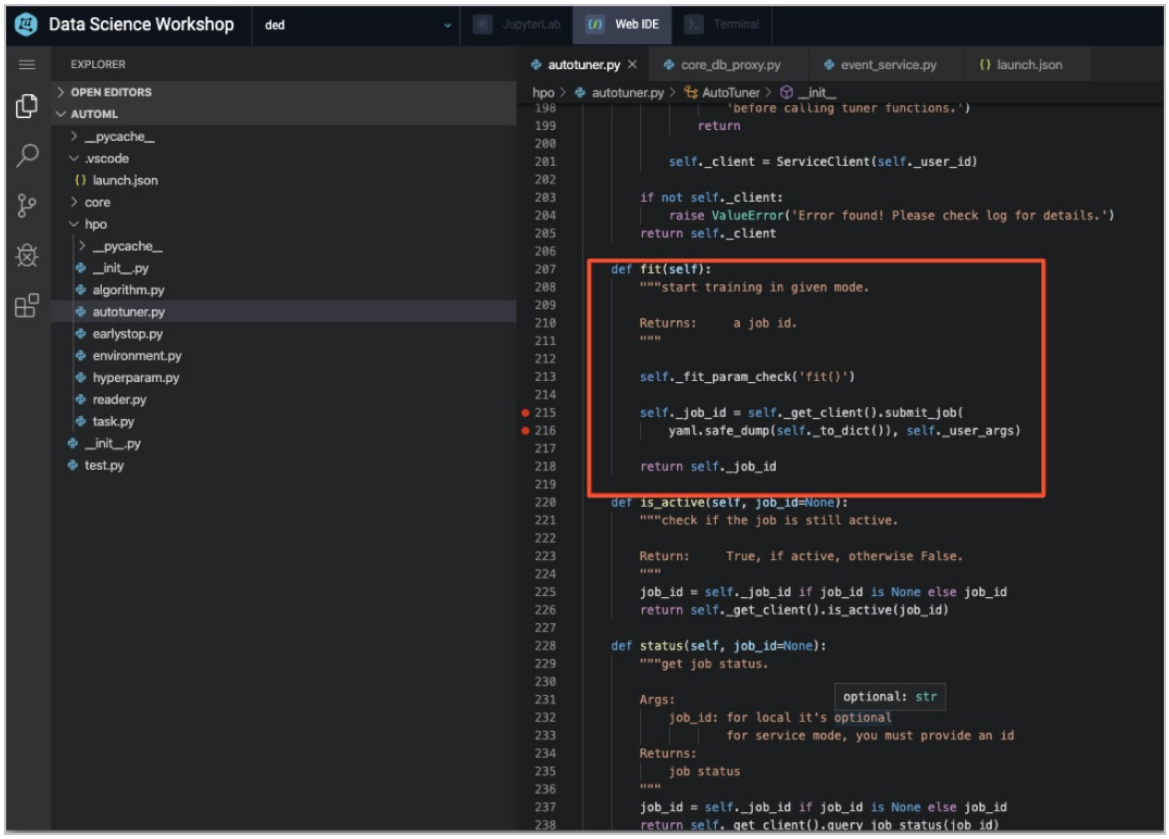
1. 查看tuner对应Class的定义所在位置。

   如下图所示。tuner对应的Class AutoTuner定义在名称为pai.automl.hpo的Python包中。




2. 通过搜索找到代码在PAI-DSW实例中的安装路径（例如*/home/admin/.local/lib/python3.6/site-packa*

*ges/pai/automl*），并将其作为项目在WebIDE中打开，如下图所示。




程序代码比较多，没有时间搞清楚所有逻辑流程，只是希望增加一个debugger并在关键点设置断点，从而快速找到问题所在。PAI-DSW中的WebIDE可以调试本地或远程运行的程序，如果程序在WebIDE之外执行，则需要在待调试程序中增加ptvsd监听代码，暂停执行直到有debugger attach。本文的目标即是在JupyterLab中运行该Sample Notebook时，能够从WebIDE中远程连接并单步跟踪程序运行。

## 操作步骤

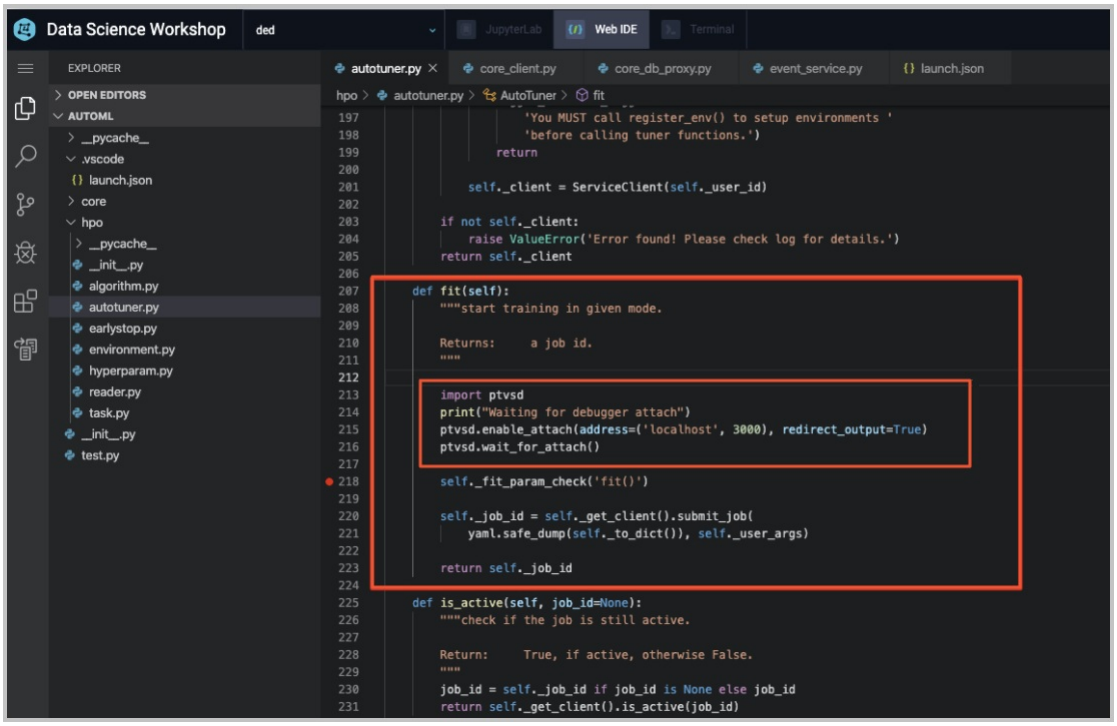
1. 在被调试代码中增加ptvsd instrumentation。
   i. （可选）如果未安装ptvsd，需要执行如下命令进行安装。

```
pip install --upgade ptvsd.
```

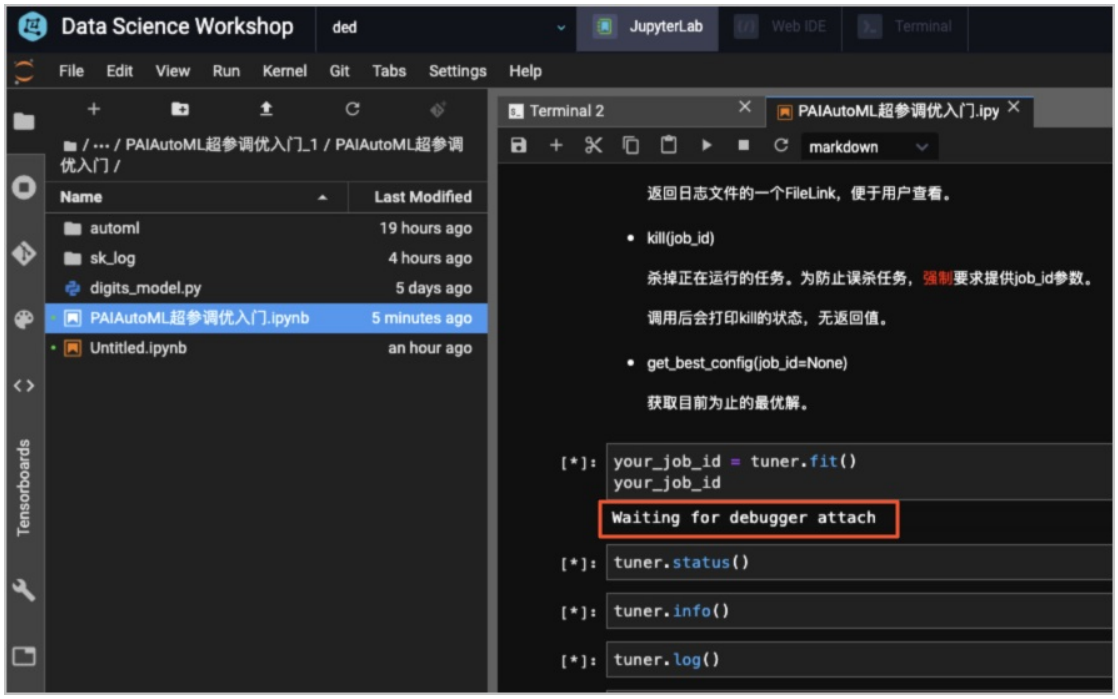
ii. 在待调试的代码中增加如下代码。

```
import ptvsd
print("Waiting for debugger attach")
# Allow other computers to attach to ptvsd at this IP address and port.
ptvsd.enable_attach(address=('192.0.2.1', 3000), redirect_output=True)
# Pause the program until a remote debugger is attached
ptvsd.wait_for_attach()
```

其中 `192.0.2.1` 指代码运行的服务器IP地址， `3000` 指debugger监听端口。针对本文排查的问题，在WebIDE中修改*/home/admin/.local/lib/python3.6/site-packages/pai/automl/hpo/autotuner.py*，改动后的代码如下图所示。



iii. 返回至JupyterLab中，重新运行该Sample Notebook，其运行结果如下图所示。




系统在 tunner.fit() 单元格下方输出 Waiting for debuger attach ，表示远程debugger设置已生效。

2. 在WebIDE中，debugger远程连接ptvsd。ptvsd在远程端等待debugger连接，必须在WebIDE debugger中设置相应的参数才能正常工作。具体来讲，即在项目路径下的*.vscode/launch.json*文件中增加如下配置。

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Remote Attach",
            "type": "python",
            "request": "attach",
            "port": 3000,
            "host": "localhost",
            "justMyCode": false,
            "pathMappings": [
                {
                    "localRoot": "${workspaceFolder}",
                    "remoteRoot": "${workspaceFolder}"
                }
            ]
        }
    ]
}
```

其中的几个关键设置包括：

- host：由于本示例中的debugger与被调试程序在同一台服务器，因此将host设置为localhost。
- port：因为必须与ptvsd等待连接的端口保持一致，所以将port设置为3000。
- justMyCode：必须设置为false，否则断点不生效。

# 4.2. 使用PAI-EasyVision进行目标检测

PAI-EasyVision（视觉智能增强算法包）提供多种模型的训练及预测功能，旨在帮助计算机视觉应用开发者方便快捷地构建视觉模型并应用于生产。本文以目标检测为例，为您介绍如何在PAI-DSW中使用PAI-EasyVision。

## 前提条件

需要准备如下安装环境：

- Python版本：Python 2.7或Python 3.4及其以上版本。
- TensorFlow社区版本：TensorFlow 1.8及其以上版本或PAI-TF。

## 步骤一：准备数据

1. 通过如下任何一种方式，下载Pascal数据集（在当前目录）。

```
#方法一：使用OSSCMD命令下载数据集。
osscmd downloadallobject oss://pai-vision-data-hz/data/voc0712_tfrecord/ data/voc0712_tfrecord --host=oss-cn-zhangjiakou.aliyuncs.com
#方法二：使用ossutil工具下载数据集，需要在配置文件中设置host为oss-cn-zhangjiakou.aliyuncs.com。
ossutil cp -r oss://pai-vision-data-hz/data/voc0712_tfrecord/ data/voc0712_tfrecord
```

2. 下载resnet50预训练模型（在当前目录）。

```
mkdir -p pretrained_models/
ossutil cp -r oss://pai-vision-data-hz/pretrained_models/resnet_v1d_50/ pretrained_models/resnet_v1d_50
```

## 步骤二：启动训练任务（在当前目录）

- 单机模式

```
import easy_vision
easy_vision.train_and_evaluate(easy_vision.RFCN_SAMPLE_CONFIG)
```

  训练过程中，每隔5000轮进行一次评估。

- 多机模式

  在具有至少两张GPU卡的服务器中才能运行，需要启动如下3个子进程：

  - ps：parameter server。
  - master：训练过程，负责写Summary、保存Checkpoint及定期进行Evaluation。
  - worker：训练过程。

  具体脚本如下所示。

```
#-*- encoding:utf-8 -*-
import multiprocessing
import sys
import os
import easy_vision
import json
import logging
import subprocess
import time
```

```
import time
# train config under distributed settings
config=easy_vision.RFCN_DISTRIBUTE_SAMPLE_CONFIG
# cluster spec 集群配置。
TF_CONFIG={'cluster':{
        'ps': ['localhost:12921'],
        'master': ['localhost:12922'],
        'worker': ['localhost:12923']
       }
      }
def job(task, gpu):
 task_name = task['type']
 # redirect python log and tf log to log_file_name
 # [logs/master.log, logs/worker.log, logs/ps.log]
 log_file_name = "logs/%s.log" % task_name
 TF_CONFIG['task'] = task
 os.environ['TF_CONFIG'] = json.dumps(TF_CONFIG)
 os.environ['CUDA_VISIBLE_DEVICES'] = gpu
 train_cmd = 'python -m easy_vision.python.train_eval --pipeline_config_path %s' % config
 logging.info('%s > %s 2>&1 ' % (train_cmd, log_file_name))
 with open(log_file_name, 'w') as lfile:
  return subprocess.Popen(train_cmd.split(' '), stdout= lfile, stderr=subprocess.STDOUT)
if __name__ == '__main__':
 procs = {}
 # start ps job on cpu
 task = {'type':'ps', 'index':0}
 procs['ps'] = job(task, '')
 # start master job on gpu 0
 task = {'type':'master', 'index':0}
 procs['master'] = job(task, '0')
 # start worker job on gpu 1
 task = {'type':'worker', 'index':0}
 procs['worker'] = job(task, '1')
 num_worker = 2
 for k, proc in procs.items():
  logging.info('%s pid: %d' %(k, proc.pid))
 task_failed = None
 task_finish_cnt = 0
 task_has_finished = {k:False for k in procs.keys()}
 while True:
  for k, proc in procs.items():
   if proc.poll() is None:
    if task_failed is not None:
     logging.error('task %s failed, %s quit' % (task_failed, k))
     proc.terminate()
     if k != 'ps':
      task_has_finished[k] = True
      task_finish_cnt += 1
     logging.info('task_finish_cnt %d' % task_finish_cnt)
   else:
    if not task_has_finished[k]:
     #process quit by itself
     if k != 'ps':
      task_finish_cnt += 1
      task_has_finished[k] = True
```

```
        logging.info('task_finish_cnt %d' % task_finish_cnt)
        if proc.returncode != 0:
          logging.error('%s failed' %k)
          task_failed = k
        else:
          logging.info('%s run successfuly' % k)
    if task_finish_cnt >= num_worker:
      break
    time.sleep(1)
```

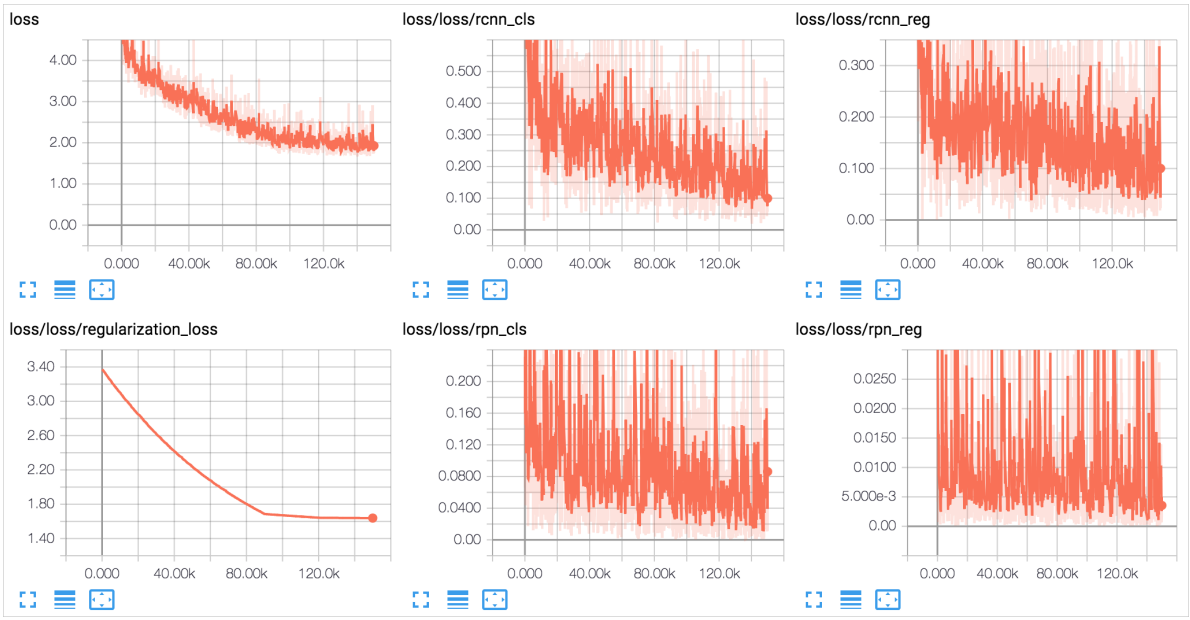
## 步骤三：使用TensorBoard观察训练过程

在pascal_resnet50_rfcn_model下保存了模型的Checkpoint和Event File，通过TensorBoard可以查看loss及mAP等相关信息，示例如下。

```
tensorboard --port 6006 --logdir pascal_resnet50_rfcn_model [ --host 0.0.0.0 ]
```

TensorBoard查看的相关信息如下：

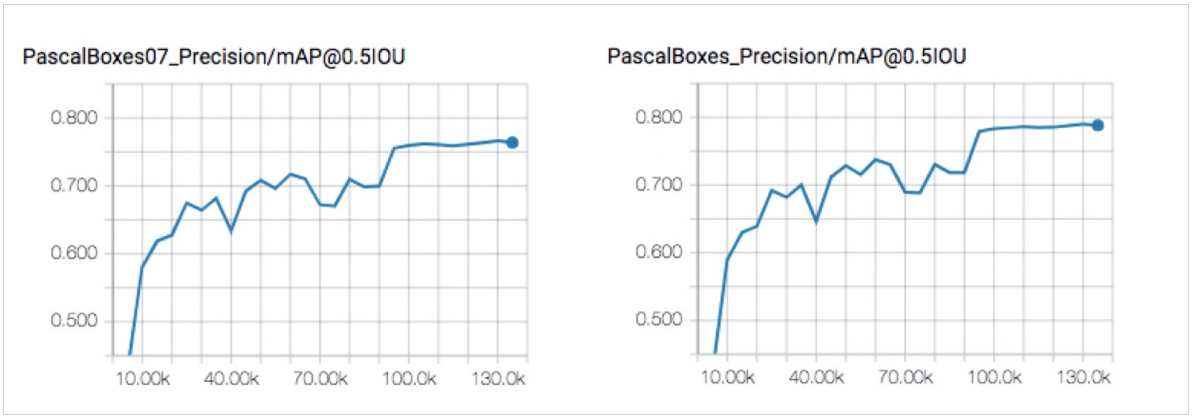
- 训练loss




其中：

  - loss：表示总loss。
  - loss/loss/rcnn_cls：表示分类loss。
  - loss/loss/rcnn_reg：表示回归loss。
  - loss/loss/regularization_loss：表示正则loss。
  - loss/loss/rpn_cls：表示RPN（Region Proposal Network）的分类loss。
  - loss/loss/rpn_reg：表示RPN（Region Proposal Network）的回归loss。

- 测试mAP




上图中分别使用了PascalBoxes07和PascalBoxes作为Metric，其中PascalBoxes07是论文中常用的Metric。

## 步骤四：评估及测试

训练完成后，您可以测试或评估训练结果：

- 在其它数据集上进行测试，得到每张图的检测结果。

```
import easy_vision
test_filelist = 'path/to/filelist.txt' # each line is a image file path
detect_results = easy_vision.predict(easy_vision.RFCN_SAMPLE_CONFIG, test_filelist=test_filelist)
```

其中 `detect_results` 包含了eval_data中每张图片的测试结果，格式为[detection_boxes, box_probability, box_class]，分别指检测到的物体位置、Probability及类别。

- 评估结果。

```
import easy_vision
eval_metrics = easy_vision.evaluate(easy_vision.RFCN_SAMPLE_CONFIG)
```

其中 `eval_metrics` 包含了PascalBoxes07 Metric、PascalBoxes Metric、global_step及训练loss（loss、loss/loss/rcnn_cls、loss/loss/rcnn_reg、 loss/loss/rpn_cls、loss/loss/rpn_reg及loss/loss/total_loss）。具体如下：

  - PascalBoxes07 Metric

```
PascalBoxes07_PerformanceByCategory/AP@0.5IOU/aeroplane = 0.74028647
PascalBoxes07_PerformanceByCategory/AP@0.5IOU/bicycle = 0.77216494
......
PascalBoxes07_PerformanceByCategory/AP@0.5IOU/train = 0.771075
PascalBoxes07_PerformanceByCategory/AP@0.5IOU/tvmonitor = 0.70221454
PascalBoxes07_Precision/mAP@0.5IOU = 0.6975172
```

  - PascalBoxes Metric

```
PascalBoxes_PerformanceByCategory/AP@0.5IOU/aeroplane = 0.7697732
PascalBoxes_PerformanceByCategory/AP@0.5IOU/bicycle = 0.80088705
......
PascalBoxes_PerformanceByCategory/AP@0.5IOU/train = 0.8002225
PascalBoxes_PerformanceByCategory/AP@0.5IOU/tvmonitor = 0.72775906
PascalBoxes_Precision/mAP@0.5IOU = 0.7182514
```

- global_step和loss

```
global_step = 75000
loss = 0.51076376
loss/loss/rcnn_cls = 0.23392382
loss/loss/rcnn_reg = 0.12589474
loss/loss/rpn_cls = 0.13748208
loss/loss/rpn_reg = 0.013463326
loss/loss/total_loss = 0.51076376
```

## 步骤五：导出模型

通过如下脚本将模型导出为SaveModel格式。

```
import easy_vision
easy_vision.export(export_dir, pipeline_config_path, checkpoint_path)
```

该程序会在export_dir路径下，以当前Unix时间戳创建模型目录，并将Checkpoint导出为SaveModel存储在该目录。

## 步骤六：评估SaveModel

评估已导出的SavedModel，Metric会打印在控制台中。

```
from easy_vision.python.main import predictor_evaluate
predictor_evaluate(predictor_eval_config)
```

其中 `predictor_eval_config` 表示Proto文档，详情请参见Protocol Documentation，您也可以参考以下样例：

- 目标检测：detector_eval.config
- 文字检测：text_detector_eval.config
- 文字识别：text_recognizer_eval.config
- 端到端文字识别：text_spotter_eval.config

## 步骤七：搭建服务

将导出的SaveModel模型存放至OSS，并将其部署为服务，详情请参见创建服务。