景行人工智能平台 V6.5

用户手册

北京景行锐创软件有限公司

目　录

1. 入门

1.1 关键词

${APPFORM_TOP}或$APPFORM_TOP：景行应用门户软件安装后的顶级目录，如：/apps/appform；

${APPFORM_CONFDIR}或$APPFORM_CONFDIR：景行应用门户软件安装后的配置文件目录，如：/apps/appform/conf；

${JH_SPOOLER_BASE}或$JH_SPOOLER_BASE：仿真计算软件的数据目录。

1.2 概述与架构

景行人工智能平台专注于为企业级人工智能开发提供全流程业务场景的综合解决方案。我们致力于为用户提供统一算力管理与智能调度服务，涵盖从数据接入、模型开发、模型训练到模型发布的全流程服务。平台支持主流深度学习框架，包括 Tensorflow、Pytorch、Mxnet、PaddlePaddle、MindSpore 等，能够迅速建立人工智能开发环境，全面管理模型训练任务，并提供团队协作、AI 资产共享、数据标注、低代码建模等核心功能，旨在为人工智能用户提供高效易用的开发平台。

景行人工智能平台对 AI 计算集群的 CPU 及多元异构 AI 加速卡资源进行统一的管理、调度及监控，从而有效提升计算资源的利用率和生产效率。我们的目标是为企业提供一体化的人工智能开发环境，助力用户在智能业务领域取得更高效的创新和生产成果。

系统的整体功能架构如下图所示：

系统功能架构图

整个系统主要包括 Web 应用门户、数据集管理、模型管理、方案管理、数据标注、镜像管理和实验管理等模块。

2. 功能介绍

2.1 模型开发

模型开发模块集成了开发环境、方案设计、案例中心和实验管理，实现通过IDE 编辑代码训练模型和通过拖拽组件搭建方案流程训练模型，添加实验管理API 的作业，可在实验管理模块查看训练曲线。案例中心，包含了拖拽式搭建模型的完整示例。

2.1.1 开发环境

开发环境支持用户使用集群资源创建编程开发环境，开发环境在用户进行AI 算法与模型开发时发挥关键作用。用户可通过 WEB 界面按需申请所需资源，迅速便捷地创建个性化的开发环境，在开发环境中进行代码编写和调试工作。系统内置了“JupyterLab”“VSCode”“RStudio”“桌面”和“Web 终端”五种环境类型。

开发环境提供了丰富的功能，包括：

AI 框架集成：集成多种主流 AI 框架，如 TensorFlow、PyTorch 等，并支持用户在开发环境内安装自定义 AI 框架，满足个性化开发需求。

多种算力规格资源选择：提供 CPU、GPU 等多种算力规格，用户可灵活选择适应任务的资源配置。新增 GPU 动态挂载卸载功能，用户可以根据任务需求动态调整 GPU 资源，提升资源利用效率。

资源监控：实时监控 GPU、CPU 和内存的使用情况，确保资源状态一目了然。

环境访问：提供多种访问方式，包括 Web 终端、Remote SSH 方式（通过本地 VSCode、PyCharm 插件连接），满足用户多样化的使用需求。容器桌面中新增 VSCode 和插件支持，用户可以直接在容器桌面中使用 VSCode 及其丰富的插件生态。同时，PyCharm 也增加了插件支持，进一步提升开发体验。

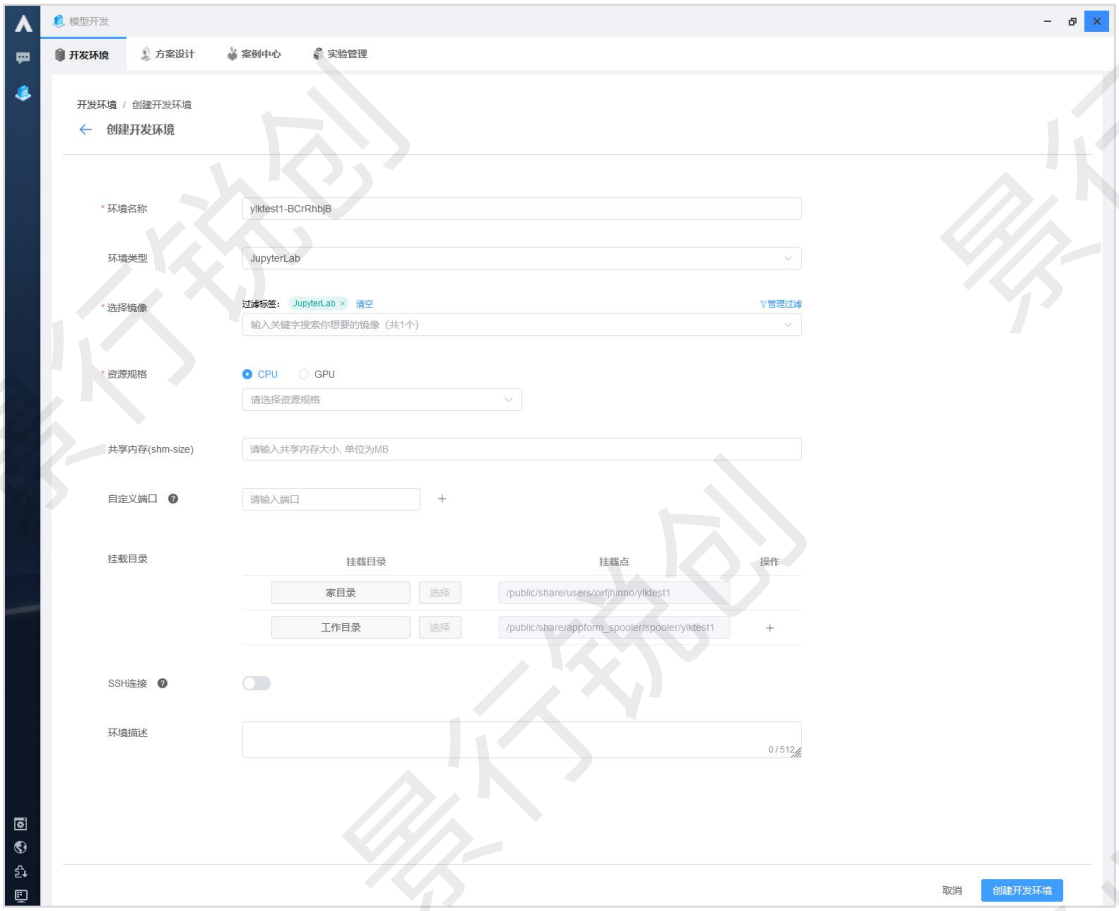
环境变更：用户能够在线调整开发环境，并保存开发环境为新的镜像。

新增 RStudio 类型支持：开发环境新增了对 RStudio 的支持，满足 R 语言开发者的需求。

VSCode 类型支持智能编程插件：VSCode 类型环境现在支持智能编程插件，如代码自动补全、语法检查、代码格式化等，帮助开发者提高编码效率和质量。

2.1.1.1 新建开发环境

以新建“JupyterLab”类型的开发环境为例，操作步骤如下：点击“创建开发环境”按钮，弹出“创建开发环境”窗口，如下图所示：



新建开发环境

图中每个参数的具体含义如下：

环境名称：必填项，默认系统会自动回填一个可用的环境名称。

环境类型：选择创建环境的类型，支持五种类型的环境，如下所示：

- JupyterLab：该环境集成了 Jupyter 开发工具。
- VSCode：该环境集成了 VSCode 开发工具。
- RStudio：该环境集成了 RStudio 开发工具。
- 桌面：提供带 GUI 的操作系统环境。
- Web 终端：仅提供 Web 终端。

选择镜像：选择创建环境实例所用的镜像。不同环境类型的镜像，会根据镜

像标签进行过滤，例如：创建 JupyterLab 类型的环境，默认只能选择包含“JupyterLab”标签的镜像。Web 终端支持所有镜像。如需订制和修改 JupyterLab、VSCode、RStudio 和桌面镜像，请基于 AI 平台提供的基础镜像进行修改。

资源规格：选择创建环境实例所需要的资源规格。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

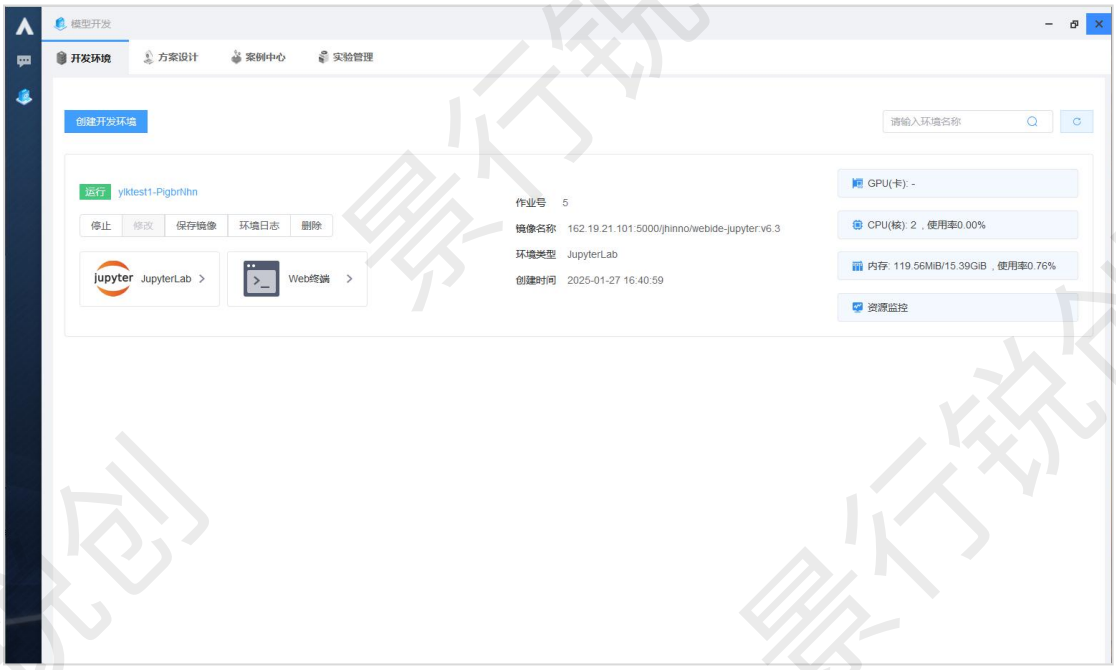
自定义端口：设置创建环境实例所用的端口。

挂载目录：默认会把“家目录”“工作目录”挂载到容器中，也可以额外指定目录，挂载到开发环境的容器中。

SSH 连接：默认关闭。打开时，开发环境启动成功后，可以使用用户名 SSH 连接信息在集群内通过 ssh 访问该容器。

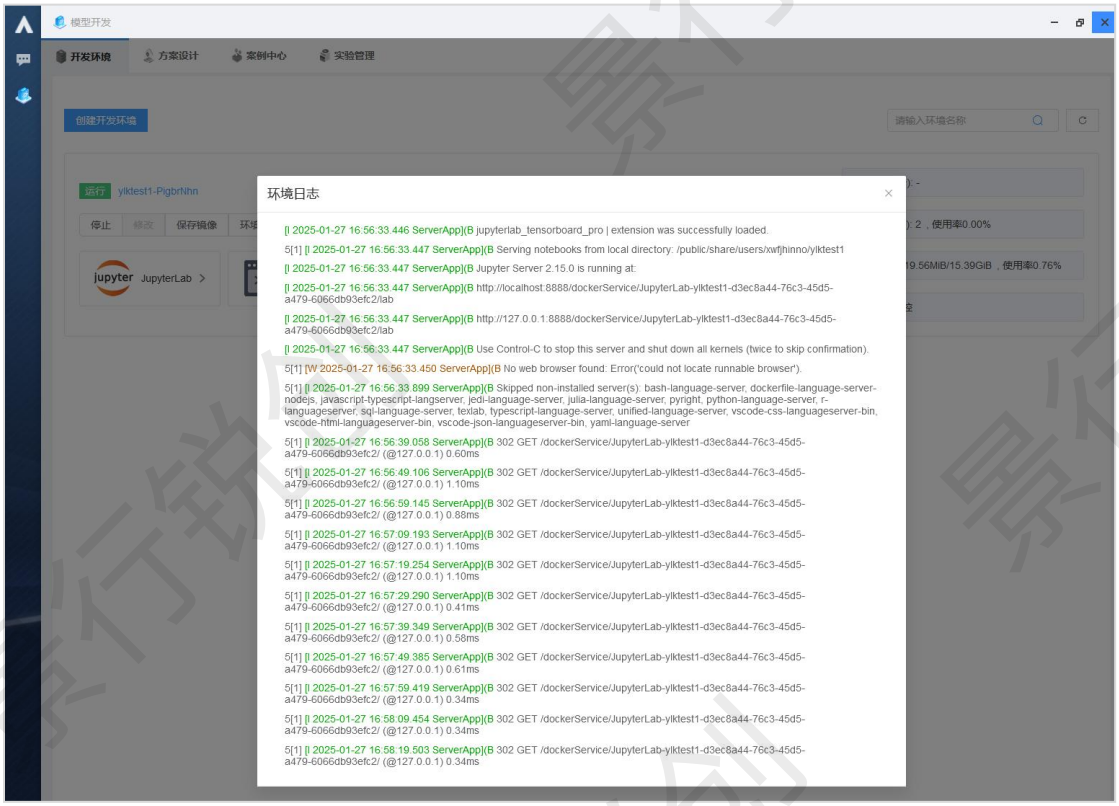
环境描述：输入环境描述信息，可选。

点击“创建开发环境”按钮，即可完成开发环境的创建，如下图所示：



开发环境列表

点击“环境日志”按钮，查看环境实例启动情况，如下图所示：



环境日志

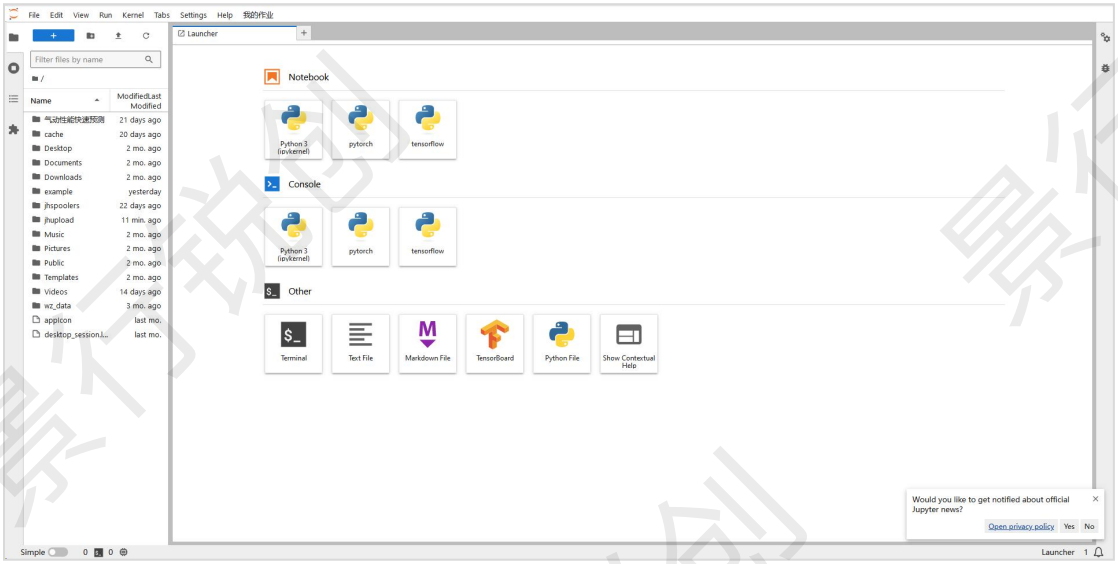
点击右上角的“刷新”按钮，手动更新环境实例的状态，如下图所示：

更新环境实例的状态

2.1.1.2 使用开发环境

2.1.1.2.1 JupyterLab

点击“JupyterLab”类型的开发环境实例上的“Jupyter”卡片按钮，访问 Jupyter 服务，如下图所示：

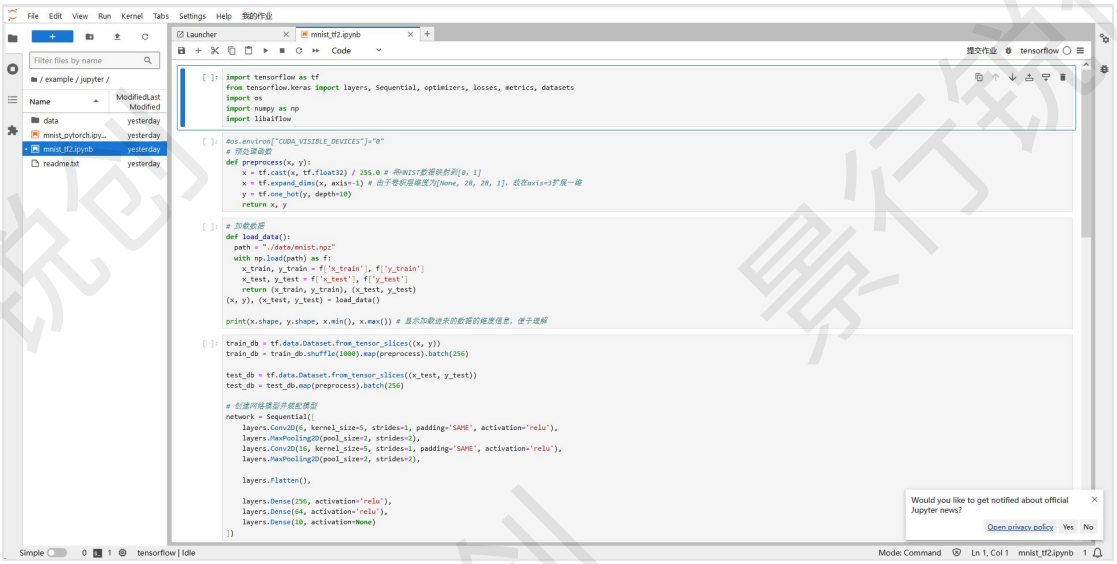


Jupyter 服务

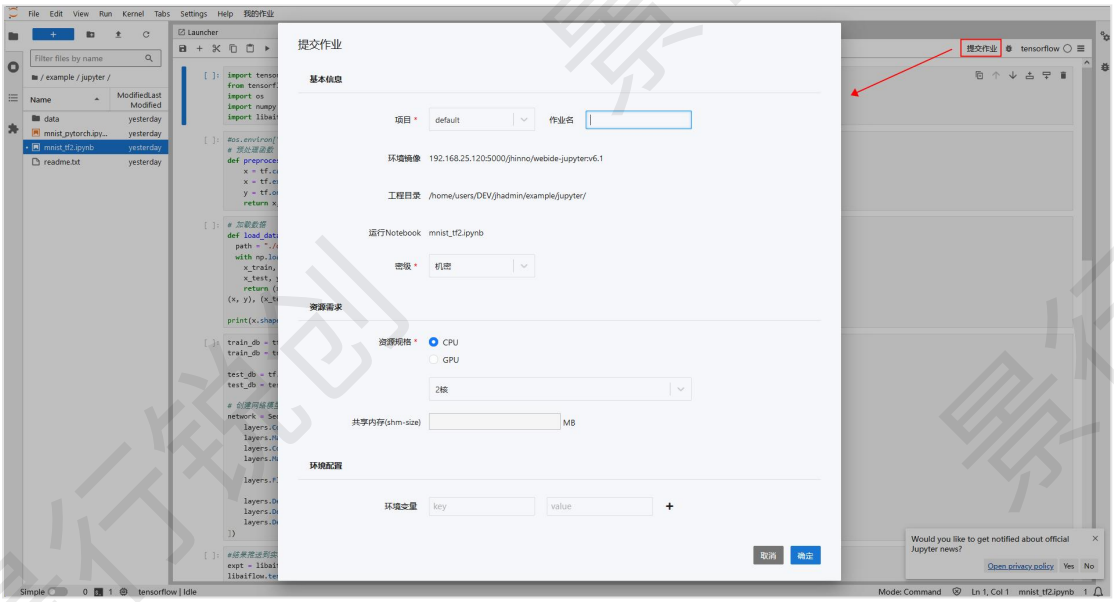
- Jupyter 插件

Jupyter 中内置景行 AI 作业插件，支持从 Jupyter 向集群中提交作业和管理作业。

打开“ipynb”格式的文件，如下图所示：



可以点击“提交作业”按钮，打开作业提交参数设置界面，如下图所示：



作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：jupyter_webide。

环境镜像：默认为创建当前环境实例时选择的镜像，也就是提交作业使用的镜像，此处镜像不可修改；

工程目录：ipynb 文件所在的目录，ipynb 运行所依赖的文件也应该在该目录下，不可修改。

运行 Notebook：选中的 ipynb 文件，如：train.ipynb。

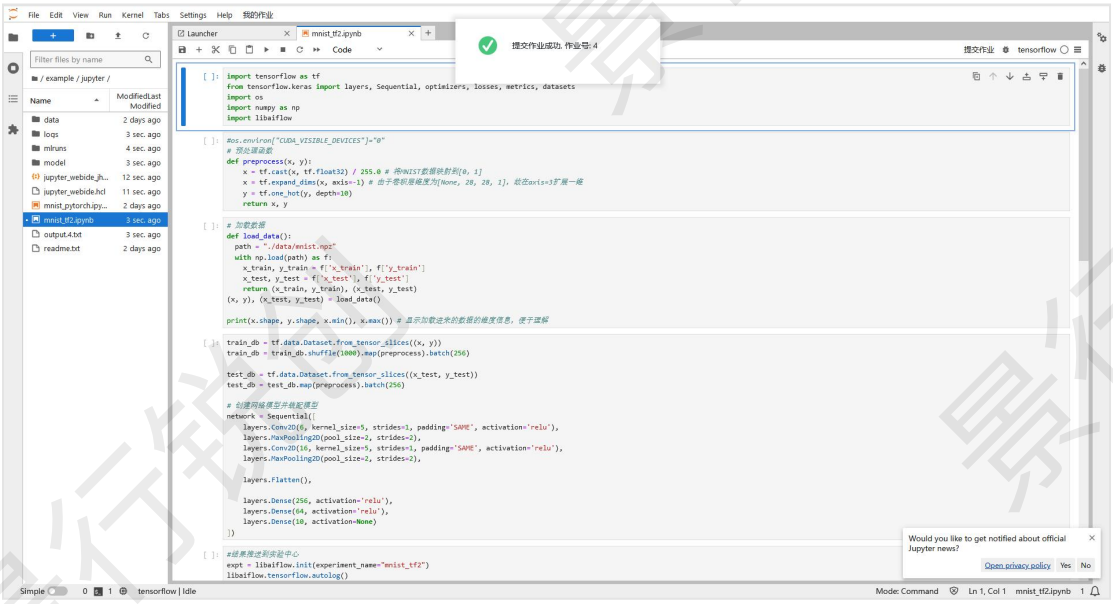
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

资源规格：必填项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

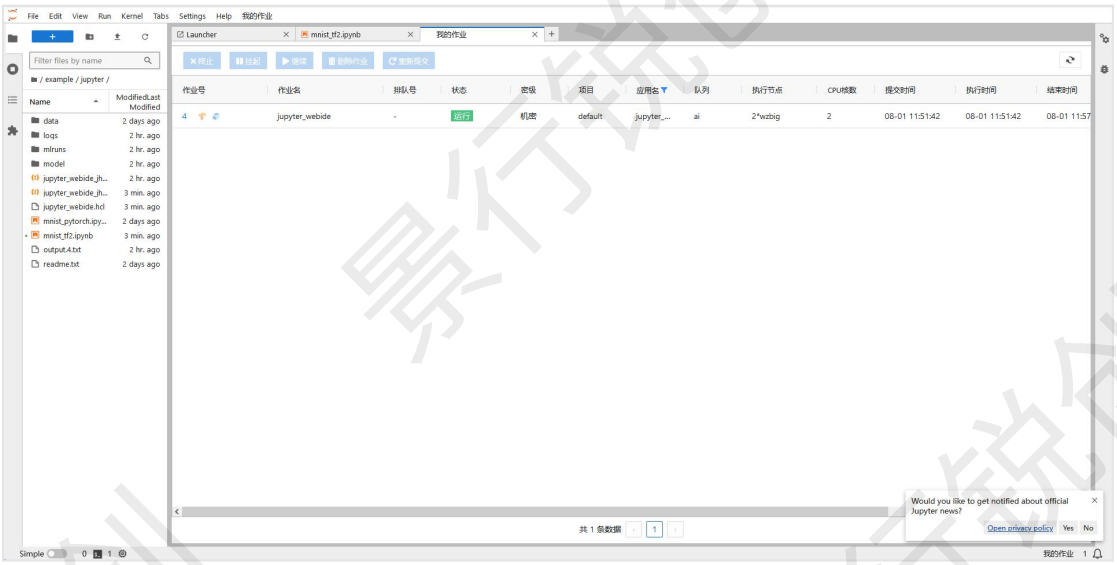
共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

环境变量：指定程序运行所需要的额外环境变量。

点击“确定”按钮，提交作业，如下图所示：

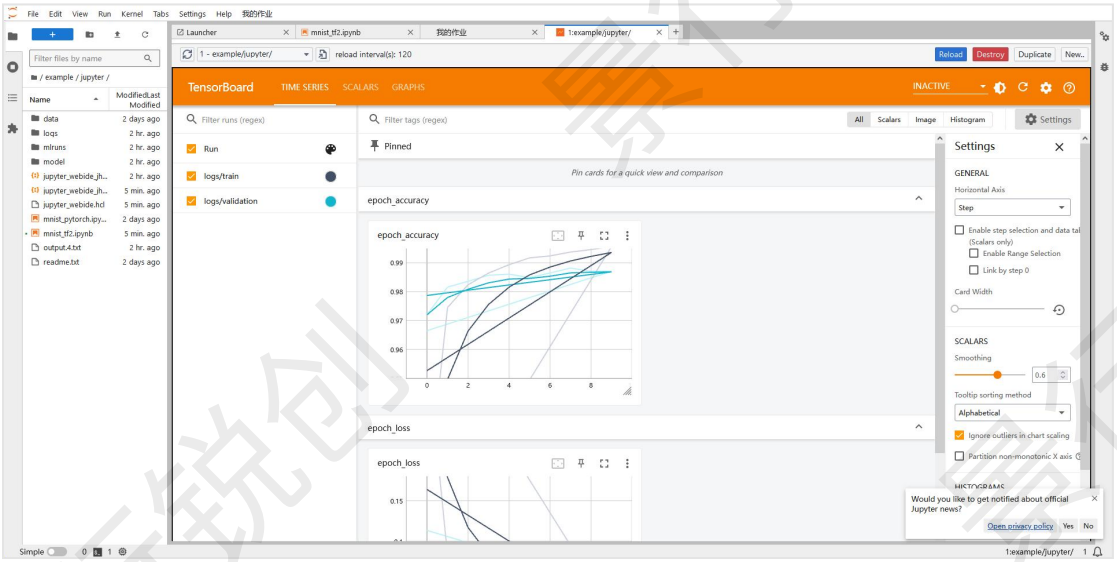


点击顶部工具栏处的“我的作业”按钮，打开我的作业界面，如下图所示：



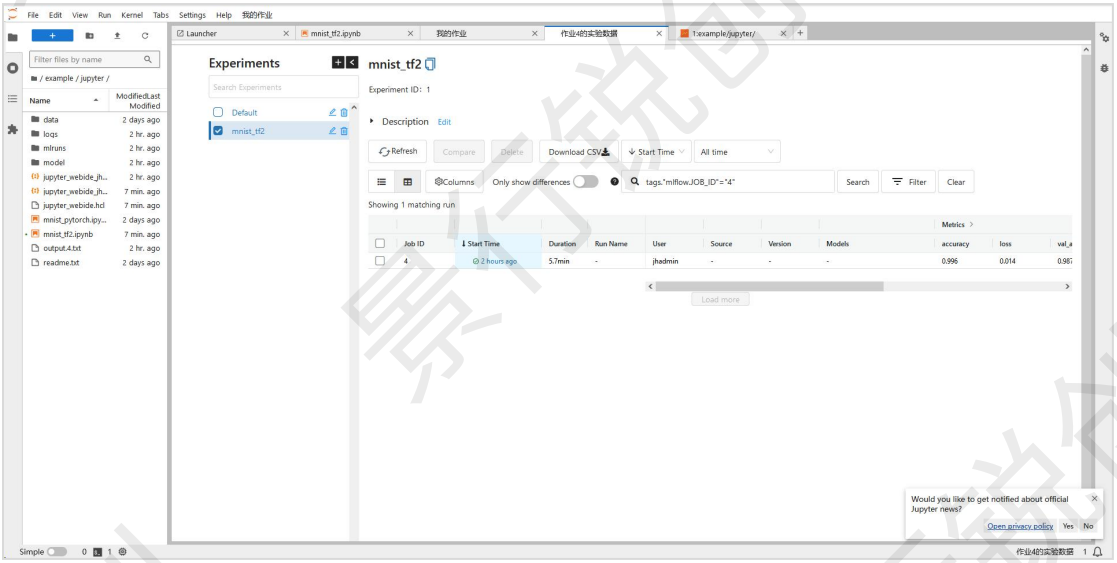
我的作业

点击作业实例上的“Tensorboard”图标按钮，访问 Tensorboard 服务查看模型数据，如下图所示：



访问 Tensorboard 服务

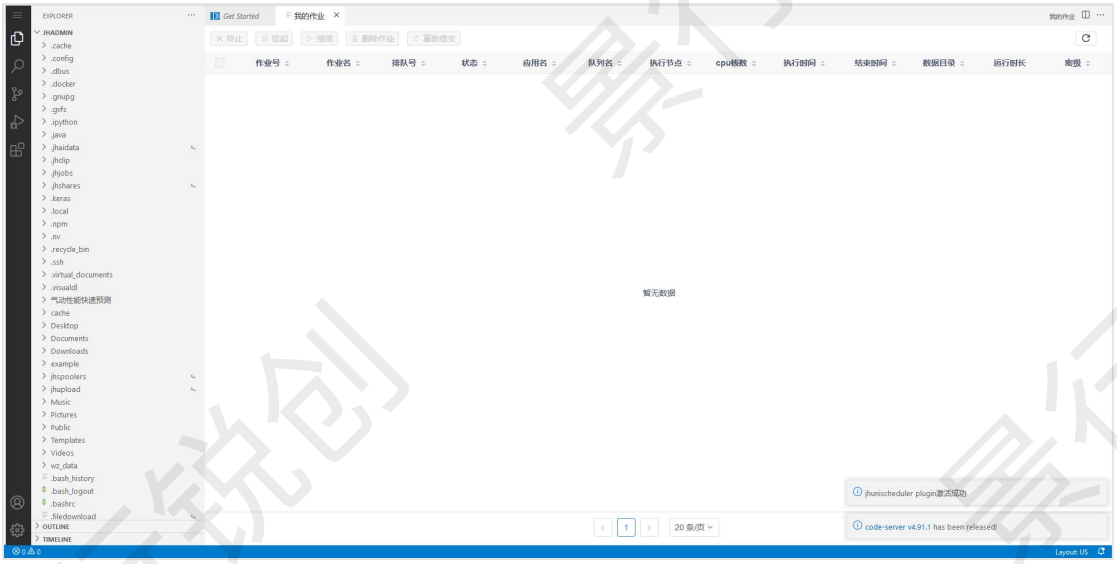
点击作业实例上的“实验管理”图标按钮，查看实验数据，如下图所示：



查看实验数据

2.1.1.2.2 VSCode

点击“VSCode”类型的开发环境实例上的“VSCode”卡片按钮，访问 VSCode 服务，如下图所示：

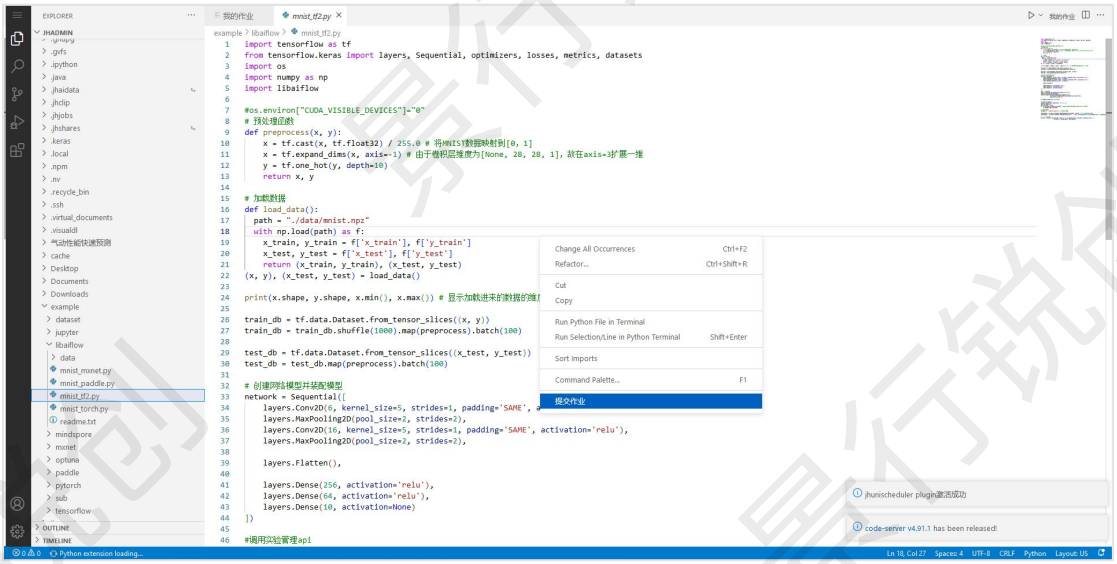


VSCode 初始界面

- VSCode 插件

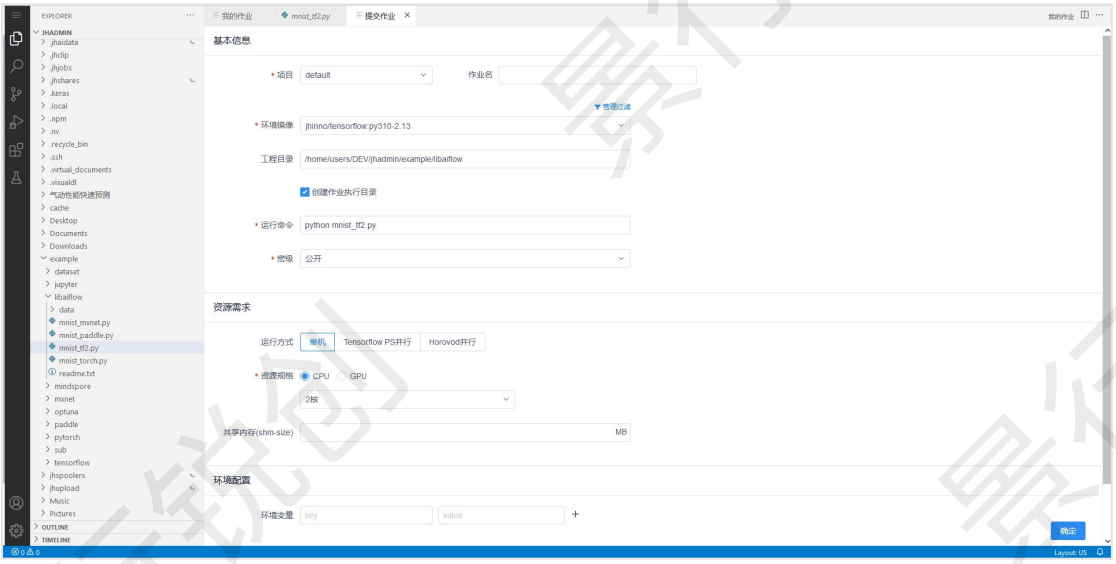
VSCode 中内置智能编程插件和景行 AI 作业插件，支持从 VSCode 向集群中提交作业和管理作业。

选中或者打开文件，在右键菜单中选择“提交作业”，如下图所示：



右键菜单中的提交作业

点击“提交作业”右键菜单，打开作业提交参数设置界面，如下图所示：



作业提交页面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：vscode_webide。

环境镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像。

工程目录：指定代码的工程目录，默认回填运行程序的父目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

密级：管理员开启密级功能后，显示此选项。

运行方式：选择作业运行方式，如下所示：

- 单机：默认选择单机，仅在单节点上运行训练程序。
  - 资源规格：必填项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。
- Tensorflow PS 并行：提交以 Parameter Server 方式实现的 Tensorflow 训练代码，支持跨节点，需填写并行相关参数，如下所

示：

- Server 数：指定参数服务的数量。
- Worker 数：指定训练服务的数量。
- 资源规格：必填项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

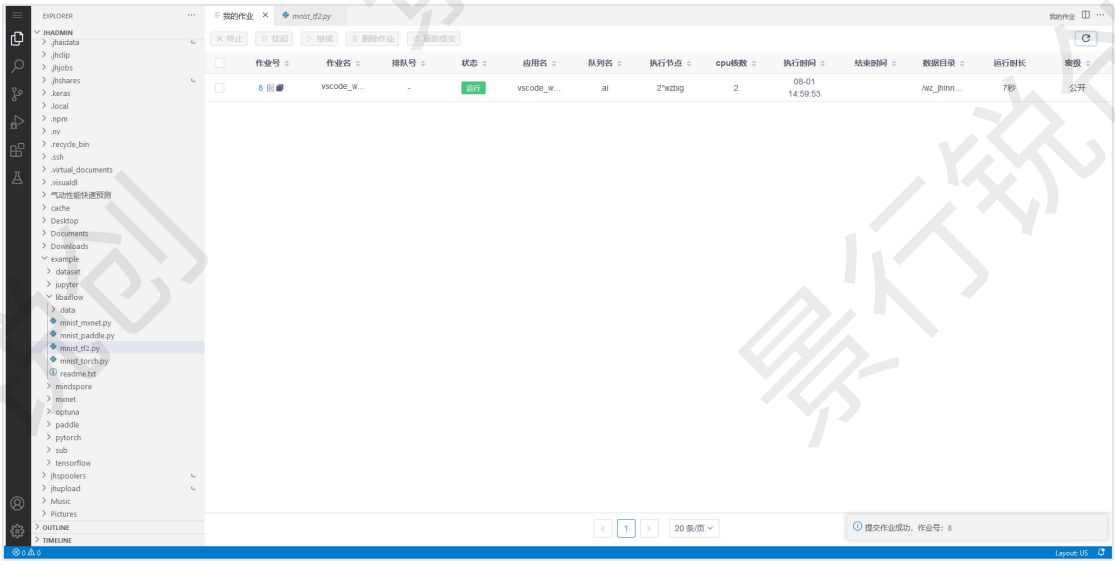
➢ Horovod 并行：提交以 Horovod 实现的并行 AI 训练程序， 需要指定 Horovod 相关参数。Horovod 并行模式需要模型训练代码以 Horovod 库的并行方式进行编码，如下所示：

- 并行实例数：指定并行训练的实例数量。
- 资源规格：必填项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

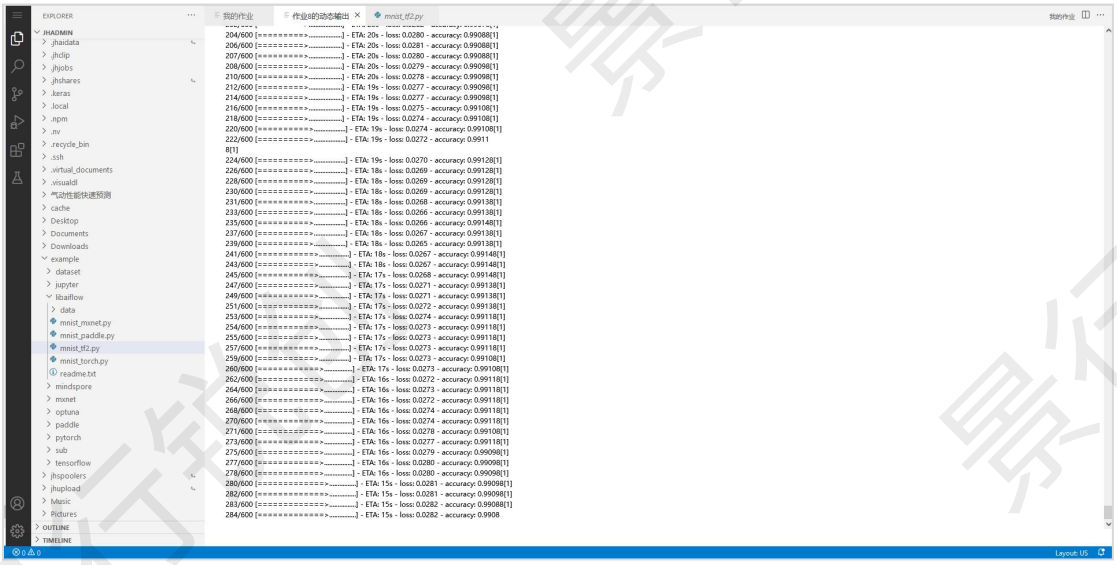
环境变量：指定程序运行所需要的额外环境变量。

在“提交作业”页面中完成基本信息、资源需求和环境配置相关信息的填写和选择后，点击“确定”按钮，提交作业。作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：

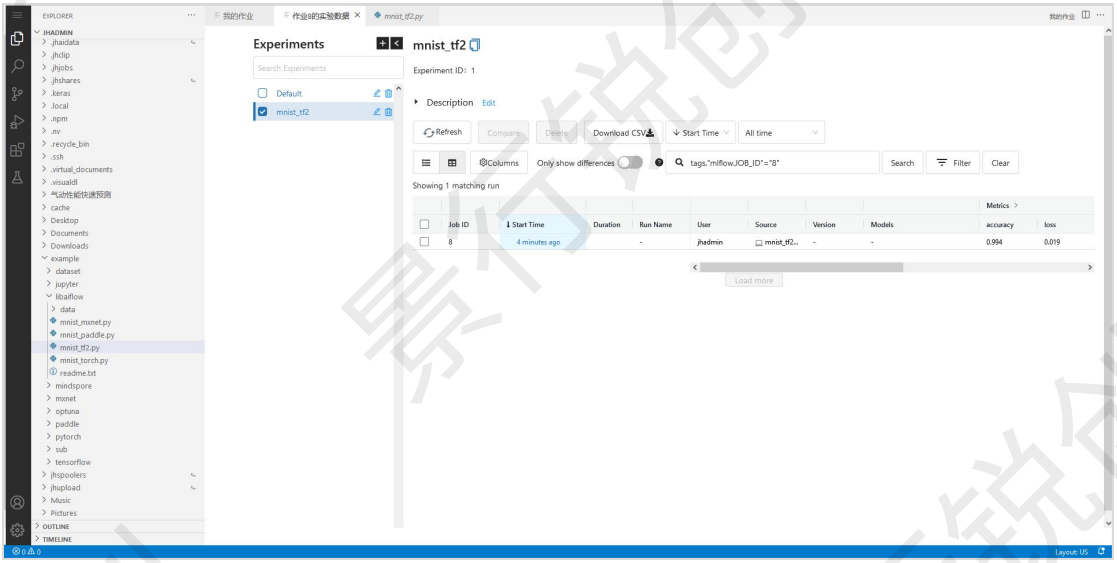


点击作业实例上的“动态输出”图标按钮，查看作业日志的动态输出，如下

图所示：



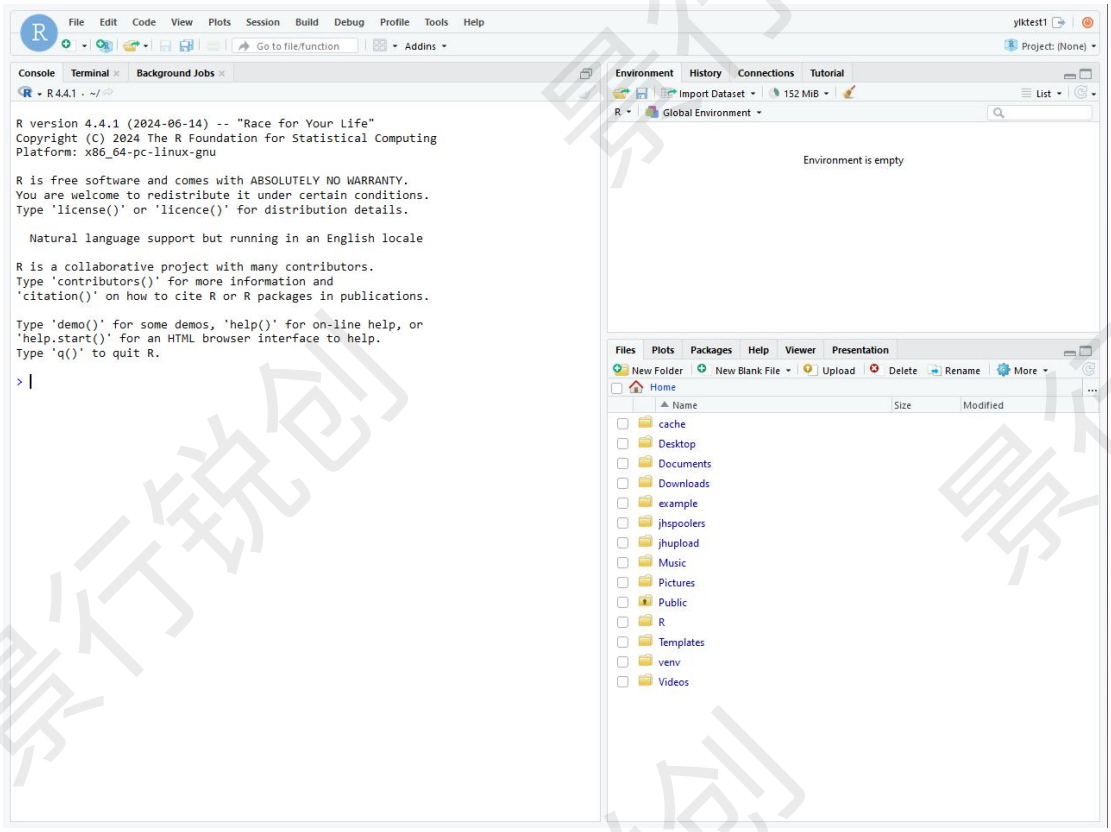
点击作业实例上的“实验管理”图标按钮，查看实验数据，如下图所示：



查看实验数据

2.1.1.2.3 RStudio

点击“RStudio”类型的开发环境实例上的“RStudio”卡片按钮，访问 RStudio 服务，如下图所示：

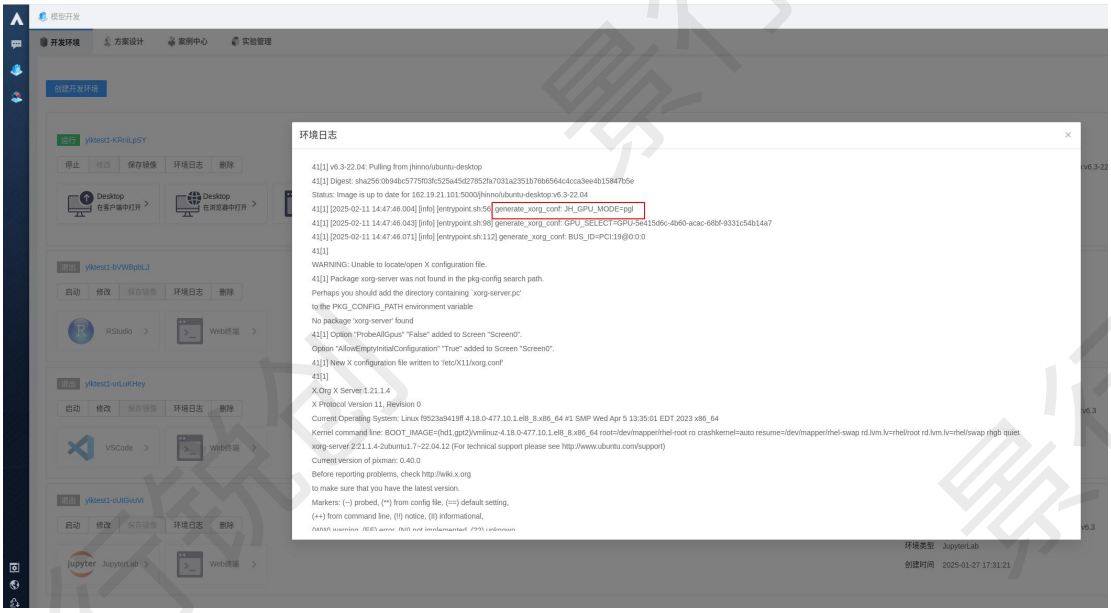


RStudio 服务

2.1.1.2.4 桌面

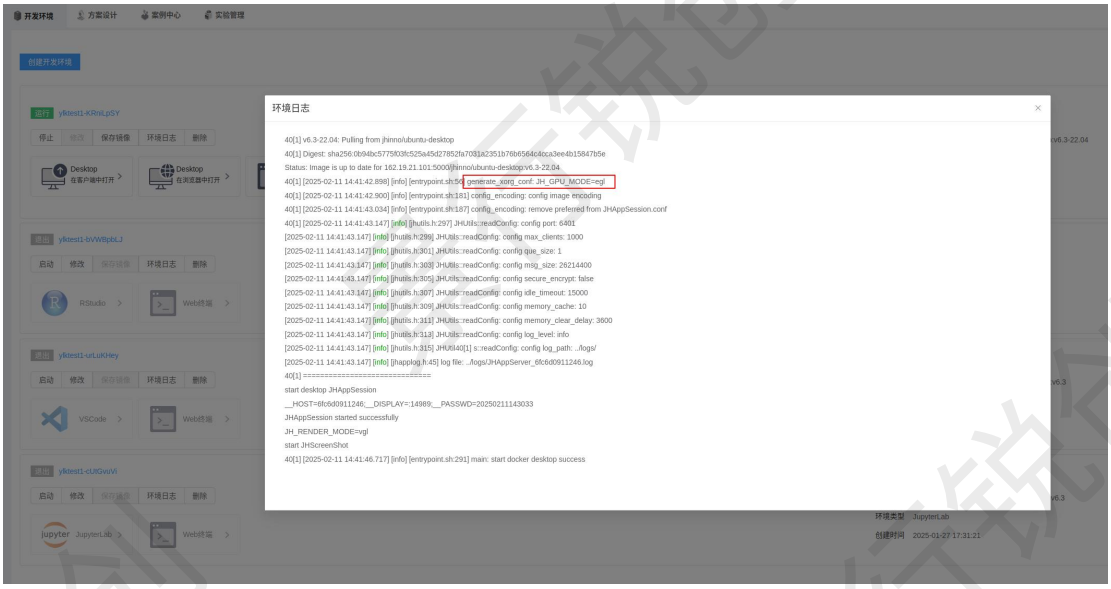
桌面类型的开发环境，提供 CentOS/Ubuntu 系统带 GUI 图像桌面的容器开发环境，并且内置了 Conda，Pycharm IDE 和 VSCode 及插件，用户可以在 GUI 桌面中打开其他图形程序，如果创建环境时选择了 GPU，在环境内支持使用 3D 加速和 CUDA 计算能力。开发环境的容器桌面支持 PGL 容器 GPU 直通模式和 EGL 容器 GPU 共享模式，提高容器内 GPU 3D 渲染性能和兼容性。

创建桌面类型的开发环境时，选择独占的资源规格，增加环境变量 JH_GPU_MODE=pgl 显卡直通模式。如下图所示：



显卡直通模式

选择共享的资源规格，增加环境变量 JH_GPU_MODE=egl，如下图所示：



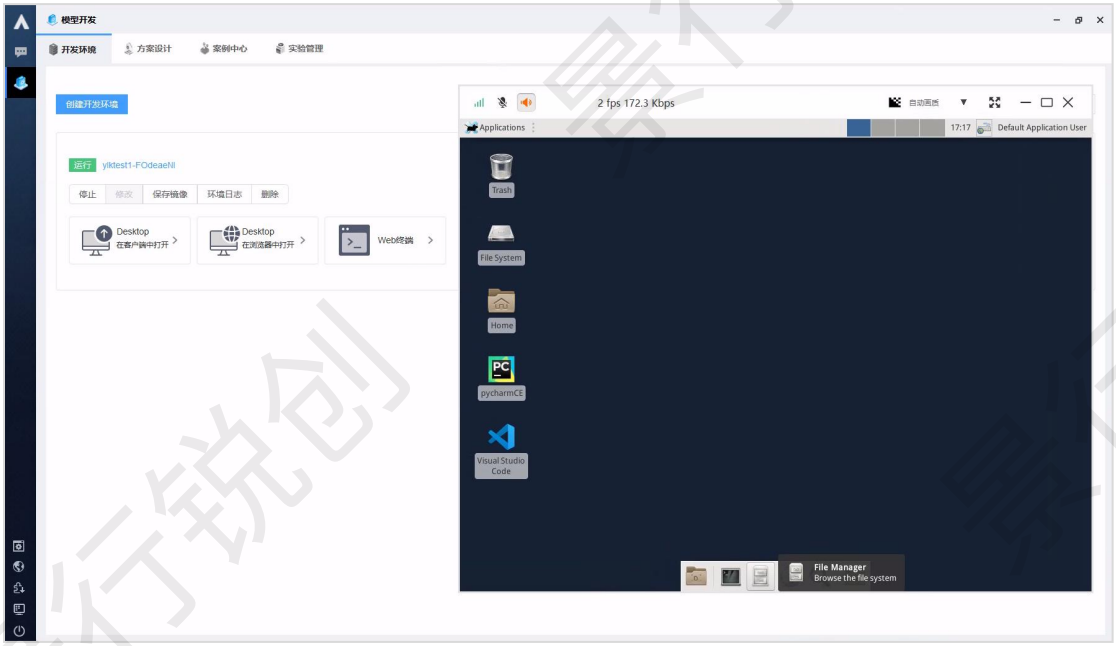
EGL 容器模式

桌面类型的开发环境创建完成后，不仅可以通过点击“桌面客户端打开”卡片按钮，以客户端访问桌面，还可以通过点击“桌面 WEB 打开”卡片按钮，以 WEB 方式访问桌面，如下图所示：

如果以客户端方式，访问桌面，需要先安装景行客户端软件，如下图所示：

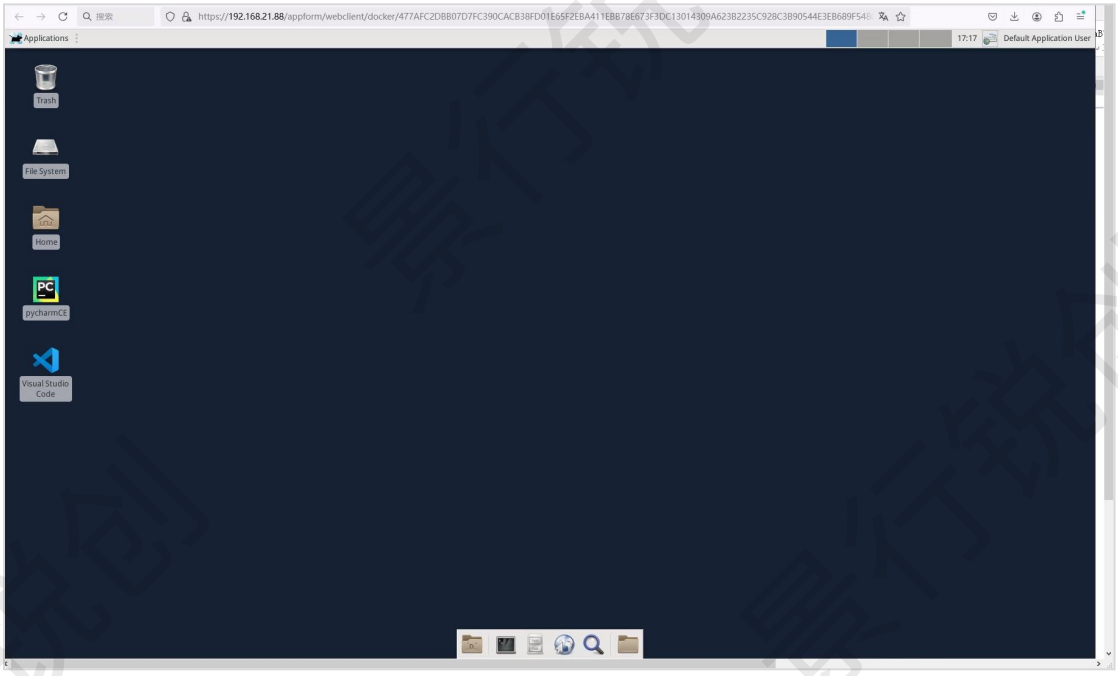
以客户端方式访问桌面

已安装景行客户端软件并重启浏览器后，点击“桌面 客户端打开”卡片按钮，以客户端方式访问桌面，如下图所示：



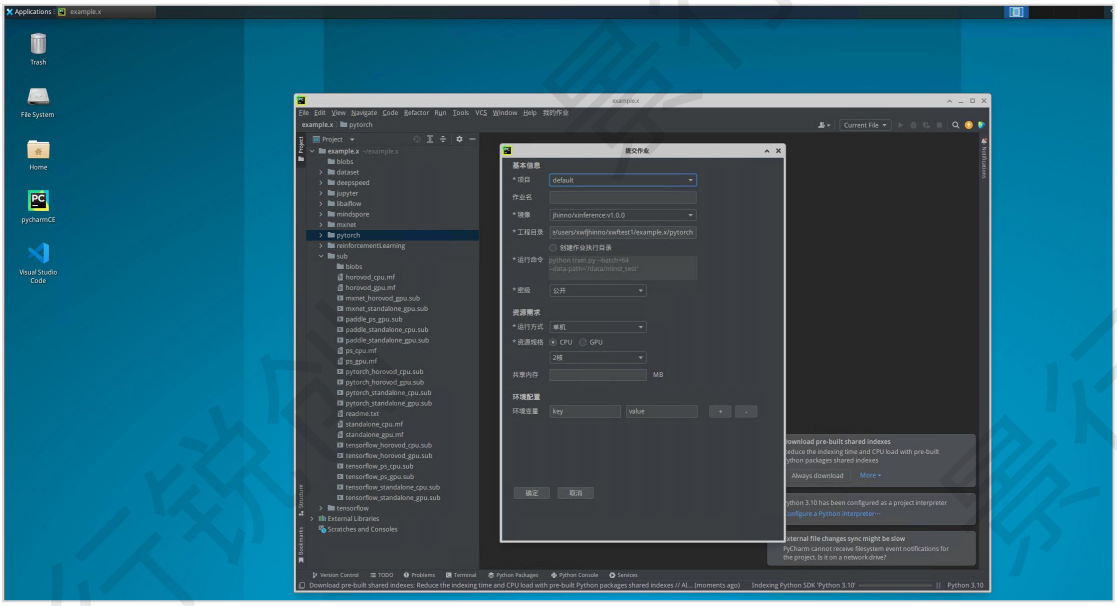
以客户端方式访问桌面

点击“桌面 WEB 打开”卡片按钮，以 WEB 方式访问桌面，如下图所示：

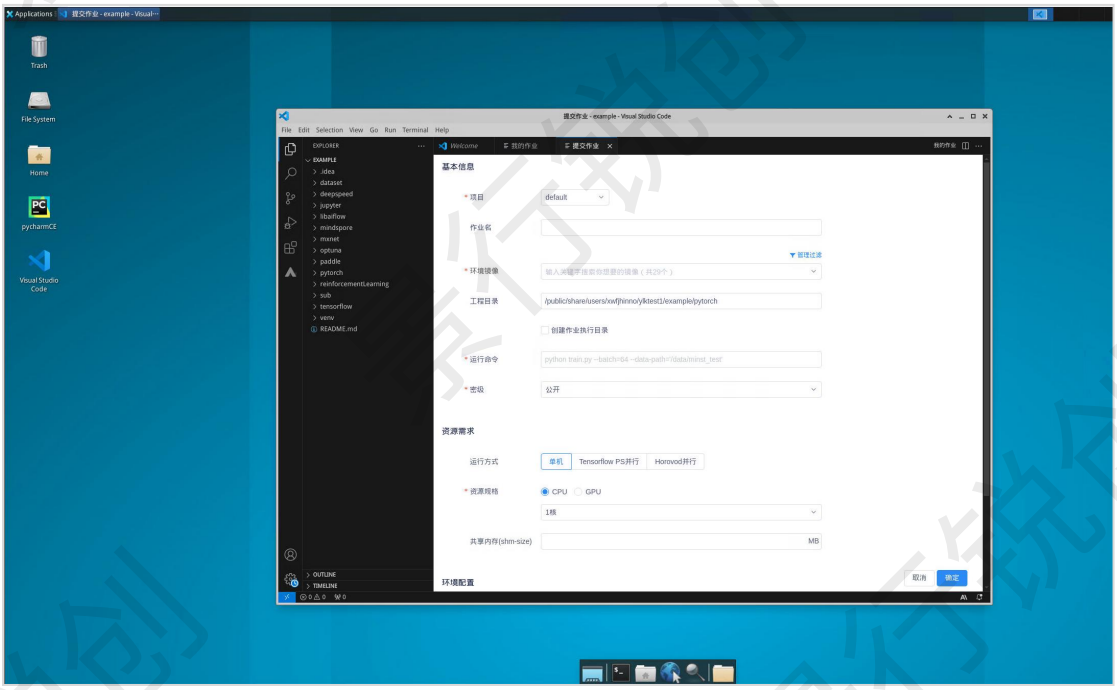


以 WEB 方式访问桌面

桌面内置 PyCharm IDE 和 VSCode 及插件，插件主要功能包括作业提交、我的作业以及对我的作业进行管理。如下图所示：



桌面 PyCharm 及插件



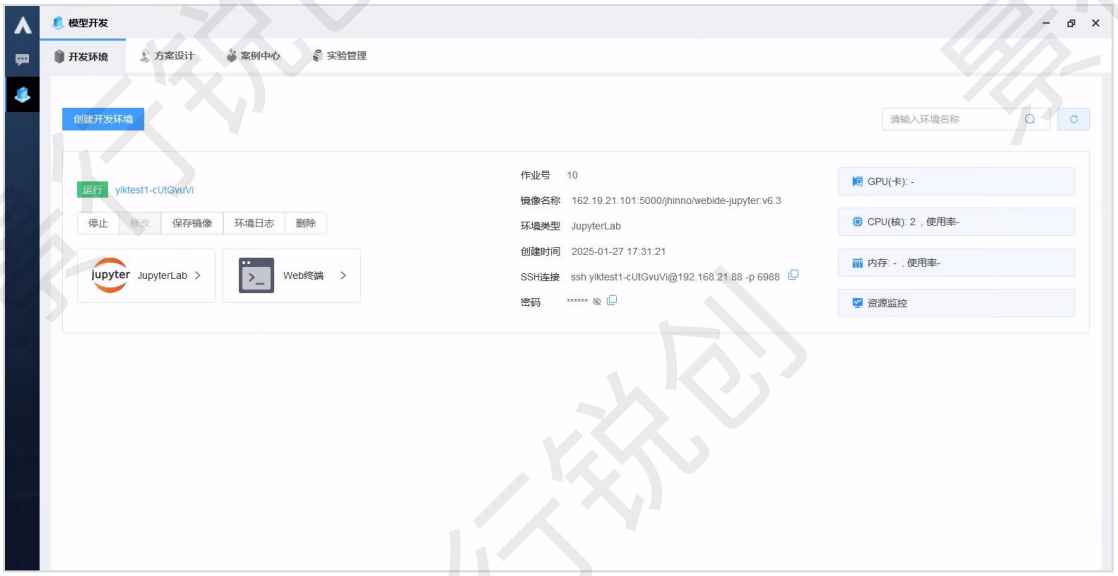
桌面 VSCode 及插件

2.1.1.2.5 使用 SSH 远程连接开发环境

JupyterLab，VSCode，RStudio 和桌面类型的开发环境，默认支持 SSH 连接功能且对应的内置镜像默认集成了 sshd 服务。而 Web 终端类型的开发环境，需要选择不仅支持 sshd 服务（详情见：附录六），还打上了 SSH 标签的镜像，才

能正常使用 SSH 连接功能。

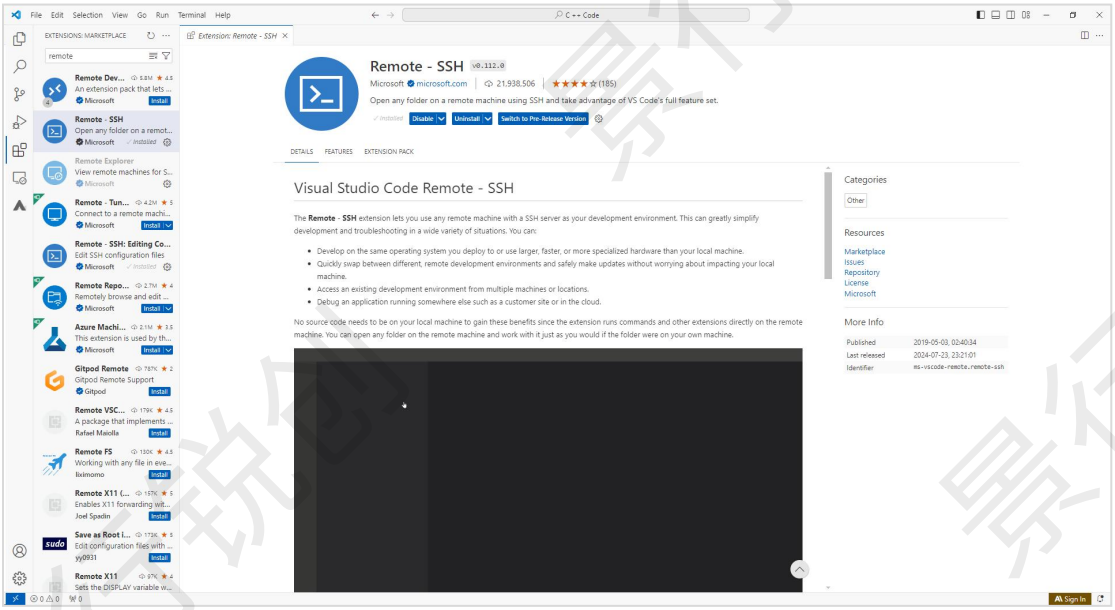
新建开发环境时，如果开启了 SSH 连接功能，在开发环境实例上会显示 SSH 连接信息。使用本地的 SSH 连接工具，复制 SSH 连接和密码，就可以连接到开发环境中。如果已存在且尚未开启 SSH 连接功能的开发环境实例，需要使用 SSH 连接功能，可以通过“修改”开发环境实例，开启 SSH 连接功能。修改后重启开发环境，当开发环境处于“运行中”状态时，即可看到 SSH 连接信息，如下图所示：



开发环境实例的连接信息

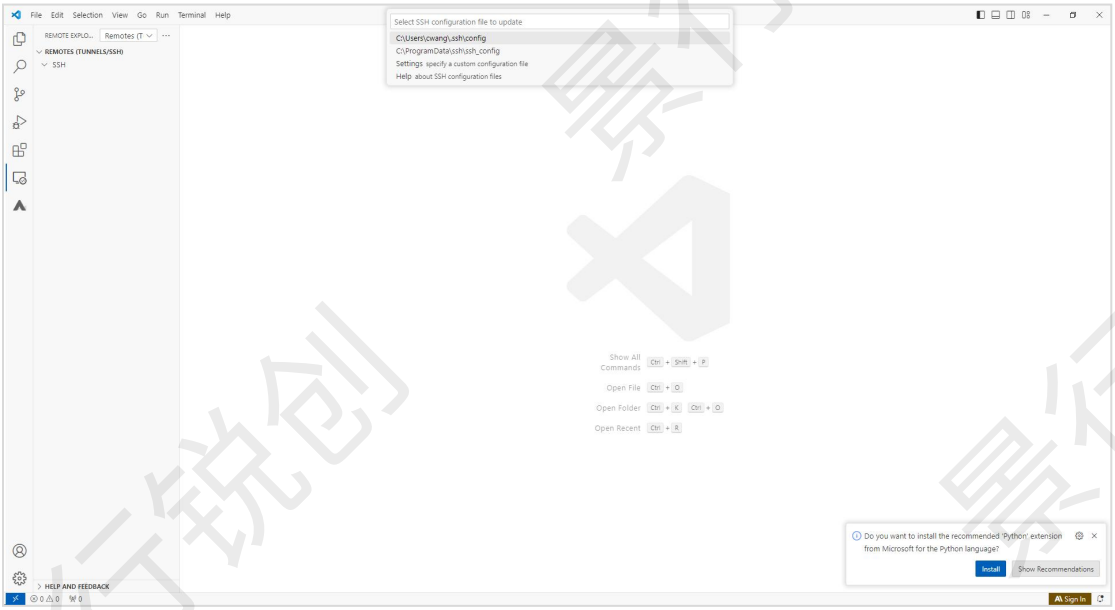
2.1.1.2.5.1 通过 IDE 连接开发环境

通过 IDE 连接开发环境，以“VSCode”IDE 为例，需要安装 Remote 插件连接开发环境，如下图所示：

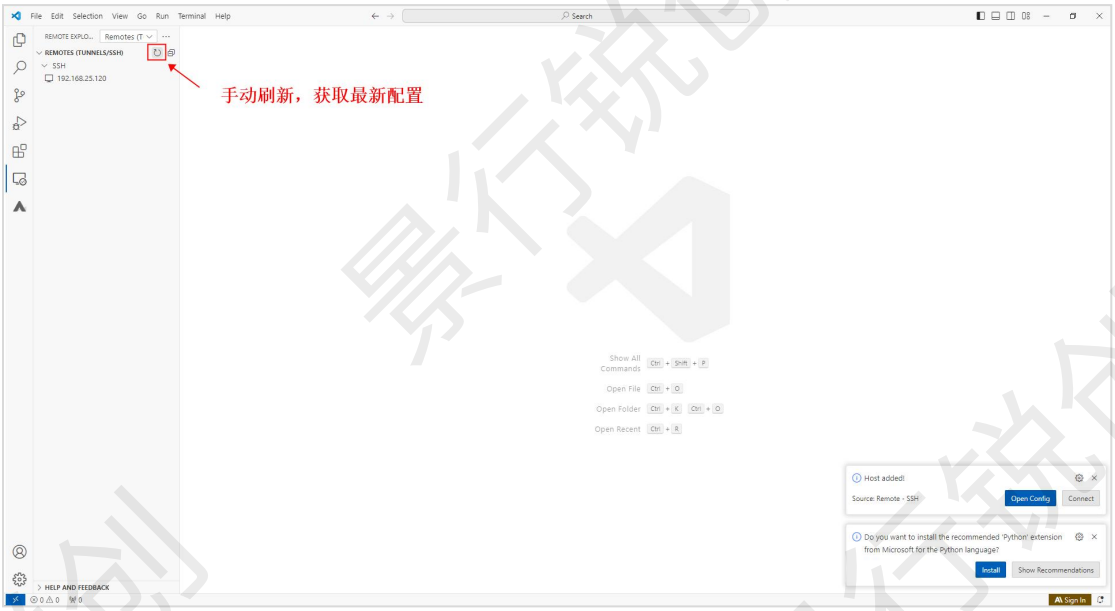


在 VSCode 左侧工具栏中点击“Remote Explorer”图标按钮，打开 Remote Explorer 扩展栏。点击“New Remote”按钮，弹出“Enter SSH Connection Command”弹框，如下图所示：

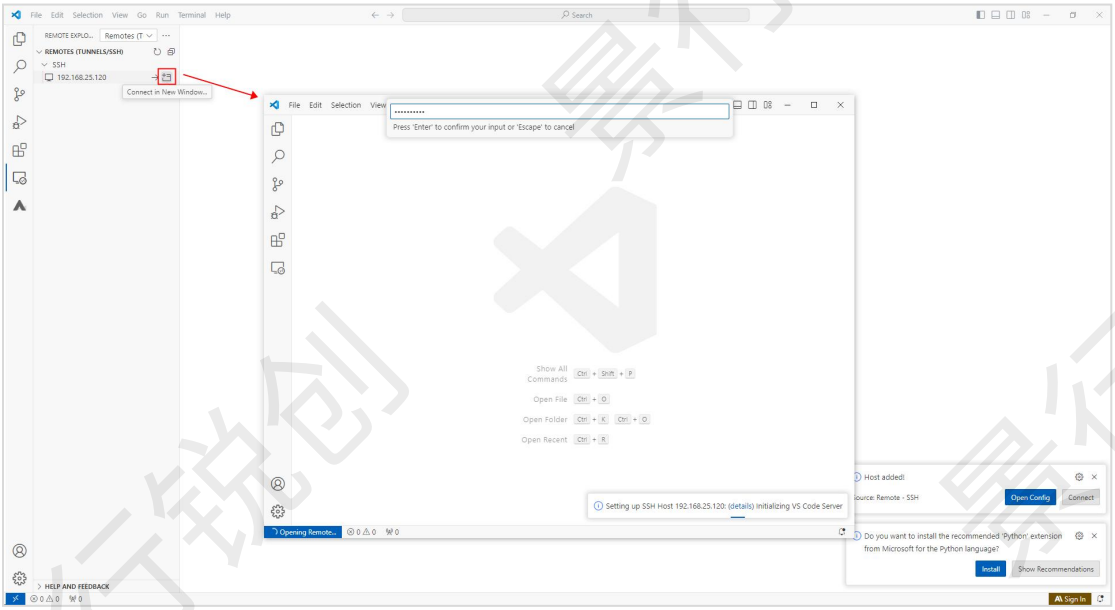
输入 SSH 连接信息后，点击“回车”键，弹出“Select SSH configuration file to update”弹框，如下所示：



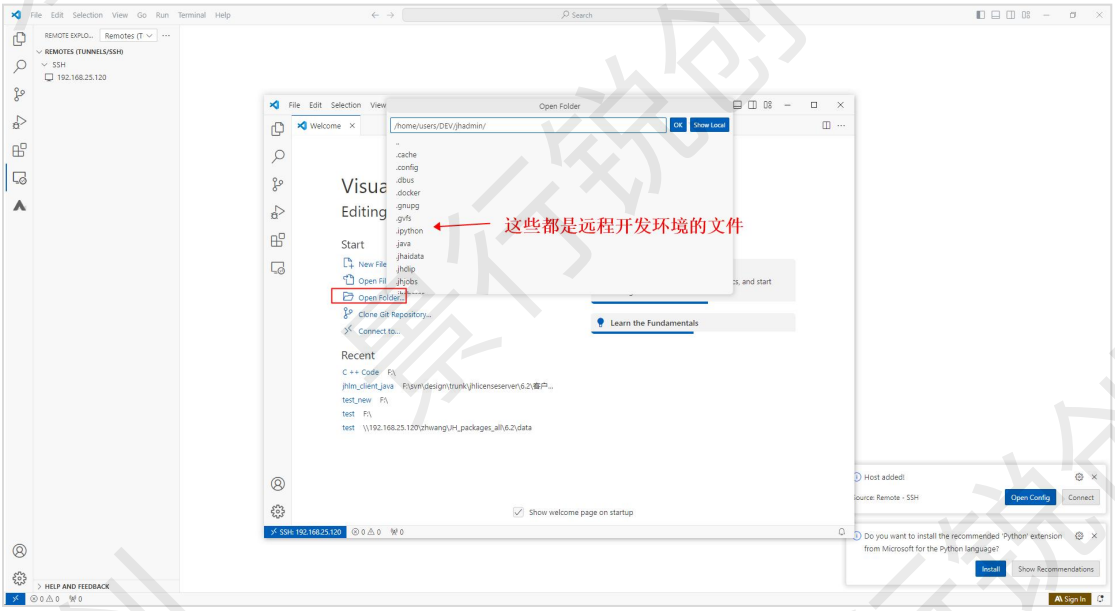
选择一个需要更新的 SSH 配置文件后，需要手动点击“刷新”按钮，获取最新的 SSH 配置，如下图所示：



点击“Connect in New Window”按钮，在新窗口连接开发环境，如下所示



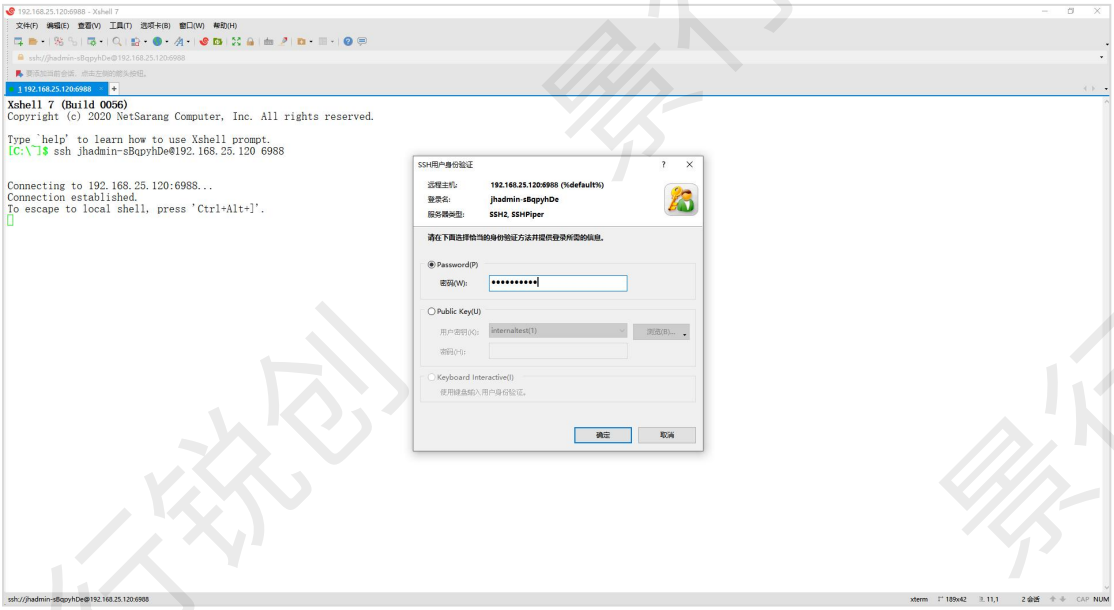
输入密码后，点击“回车”键，连接开发环境，如下图所示：



VSCode IDE 连接开发环境

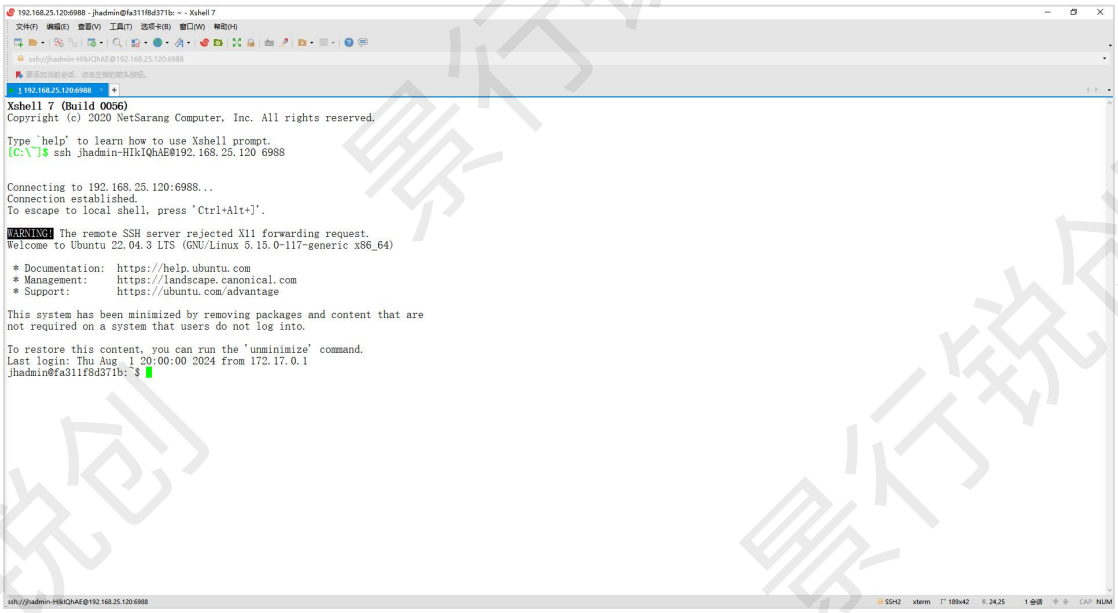
2.1.1.2.5.2 通过 SSH 工具连接开发环境

通过 SSH 工具连接开发环境，以“Xshell”SSH 工具为例，如下图所示：



注：Xshell 不支持-p 参数，所以连接时，不直接应用开发环境的 SSH 连接信息：ssh jhadmin-HIkIQhAE@192.168.25.120 -p 6988，需要手动删除“-p”，进行连接：ssh jhadmin-HIkIQhAE@192.168.25.120 6988。

输入密码后，点击“回车”键，进行连接，如下图所示：

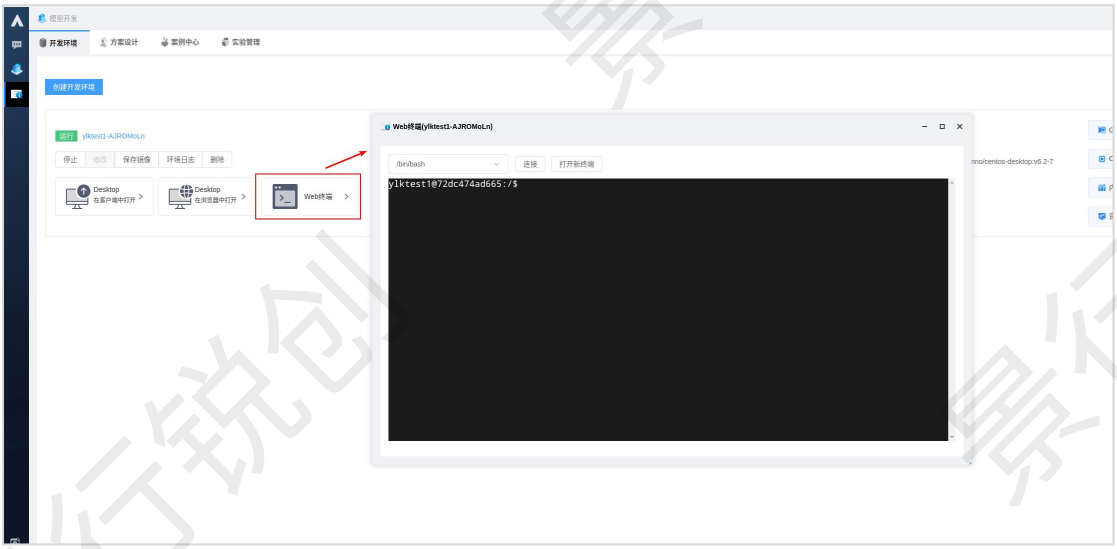


SSH 终端连接

2.1.1.2.6 命令行提交管理作业

开发环境容器中集成了调度命令行接口，可以通过终端命令行方式提交作业

和管理作业，如下图所示：



WEB 终端

以单机 Tensorflow 训练程序为例，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J tensorflow_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/tensorflow:2.13  --cmd='python
mnist_tf2.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
jsub < tensorflow_standalone_cpu.sub
```

管理作业命令详细介绍见《景行资源管理与调度软件用户手册》，以查询作业为例，执行命令如下：

```
jhadmin@c47a7d1c9e79:~$ jjobs
JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME     SUBMIT_TIME    ORDER
88         jhadmin  RUN   ai_excl   jhai110   2*jhai110 *in-BxfsjVtB Feb 05 02:22   -
93         jhadmin  RUN   ai_excl   jhai110   2*jhai110 *ard-jhadmin Feb 06 02:02   -
94         jhadmin  RUN   ai_excl   jhai110   2*jhai110 *in-SwJCdgGU Feb 06 02:04   -
102        jhadmin  RUN   ai_excl   jhai110   jhxihost* *in-KEtjGBOe Feb 06 08:03   -
jhadmin@c47a7d1c9e79:~$
```

注意：在 RStudio 开发环境的“Web 终端”中看到的用户是 root，需要执行 su - userx 切换成普通用户，才能提交景行资源管理与调度作业。

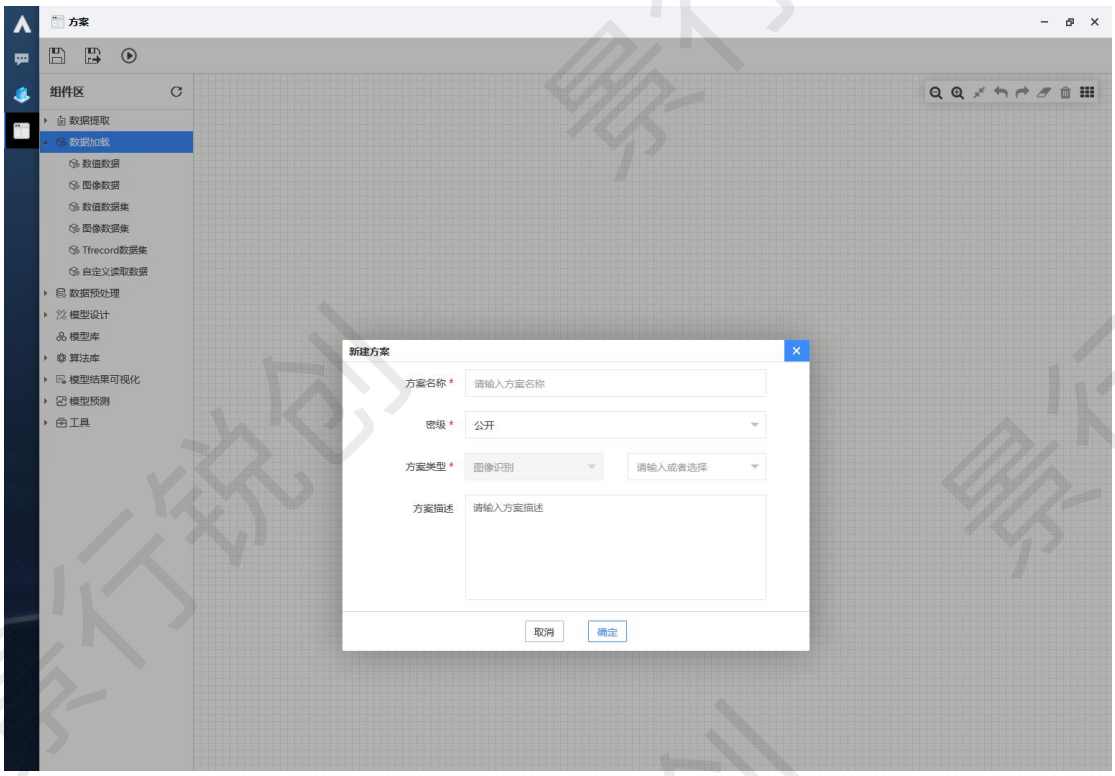
2.1.2 方案设计

方案设计提供人工智能的流程可视化设计，方案提供了数据接入、数据处理、机器学习和深度学习等组件，通过拖拽组件和连线的方式可以实现数据处理和人工智能建模、训练、评估等复杂流程。

2.1.2.1 构建方案设计流程

2.1.2.1.1 新建

在模型开发应用选择方案设计，点击“新建方案”卡片，弹出新建方案的表单页，如下图所示：



新建方案

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有方案名称一致。

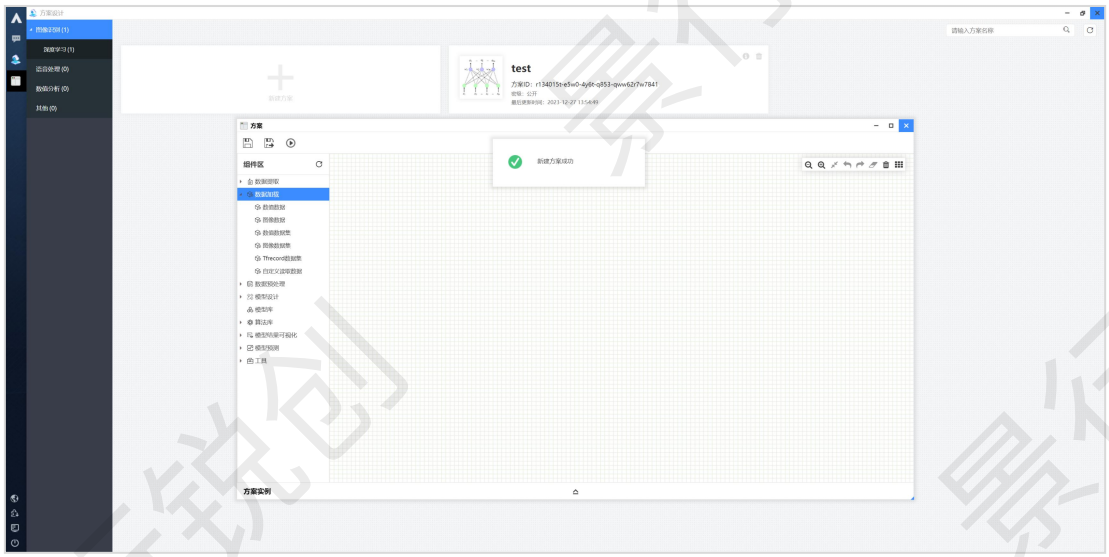
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

方案类型：默认为已选的方案类型，也可以自定义方案类型。

方案描述：用于描述方案设计的相关内容，选填。

2.1.2.1.2 设计

填写完成新建方案的表单后，点击“确定”按钮，进入方案设计页面。

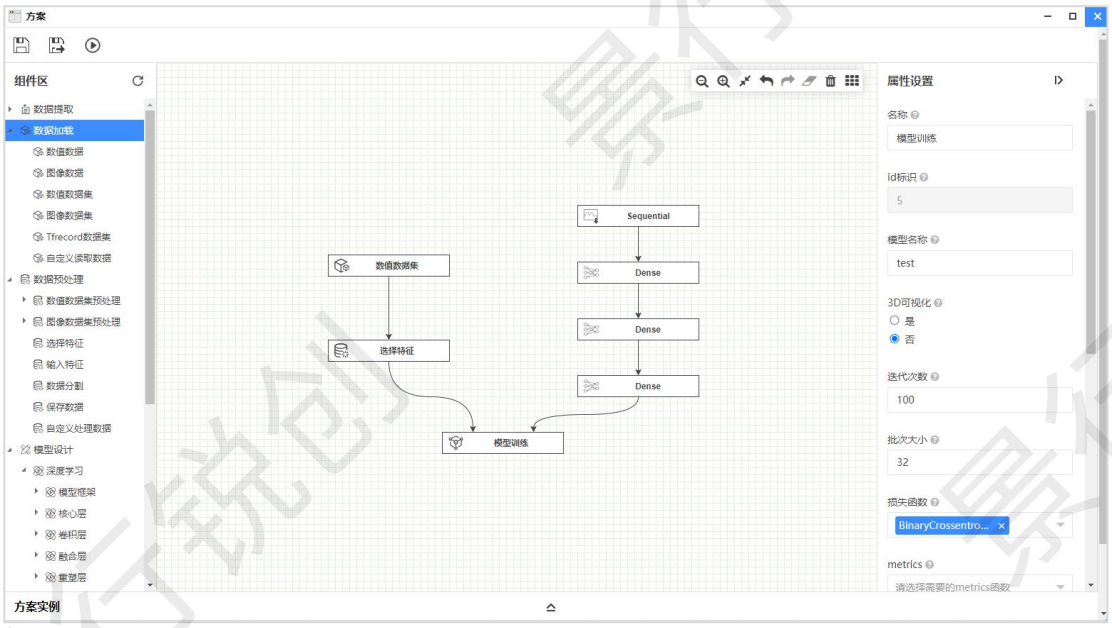


方案设计页面

“方案设计”左侧区域为组件区，组件区又根据组件属性分为“数据提取”，“数据加载”，“数据预处理”，“模型设计”，“模型库”，“算法库”，“模型结果可视化”，“模型预测”和“工具”。每一个组件都是一个模块的封装。

“方案设计”右侧区域为属性区，组件拖拽到画布区域后，属性设置面板就会从画布右侧滑出，用户可自行配置参数。当属性设置有误或未设置属性时，设计区的组件右侧会显示黄色叹号的图标，当鼠标移至此图标上时会显示出错误信息，必须纠正错误后才能够开始运行此方案。

画布右上角的工具栏可对画布区域和画布中的组件进行快捷操作，功能包含缩小、放大、实际比例、后退、前进、删除和清空。其中“删除”按钮仅会删除画布中选中的任意组件，“清空”则会删除整个画布中的内容，长按鼠标右键可拖动整个画布。



画布工具及组件属性配置

自定义函数管理

在自定义函数管理中可以创建和管理“损失函数”和“Metrics 函数”，选择相应的分类并点击新建卡片即可新建自定义函数。

创建成功的自定义函数会以卡片的形式显示在右侧区域中，卡片中的信息包括函数名称、ID 和创建时间，当鼠标停留在卡片右上角的“描述”按钮处时，会显示此函数的描述信息，点击卡片右上角的删除按钮即可删除当前自定义函数，删除操作不可恢复。

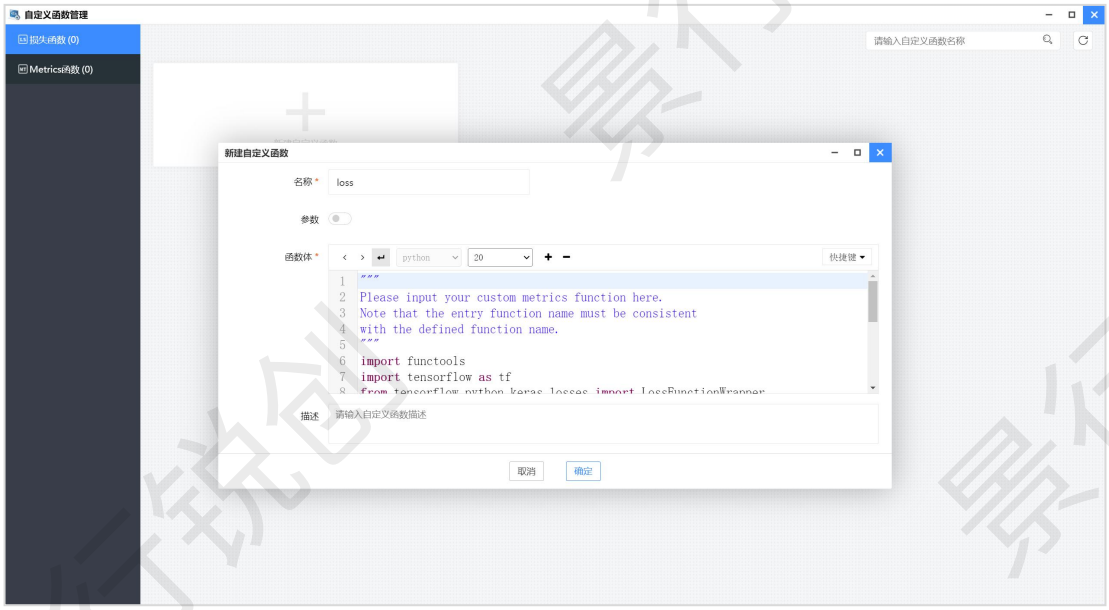
注意：

(1) 在此创建的自定义函数可在方案设计中被引用。

(2) 组件区域添加“模型评估”按钮，在案例详情页面可产生评估结果页面。

➢ 损失函数

进入“自定义函数管理”页面，选择左侧“损失函数”，点击“新建自定义函数”卡片，弹出新建窗口，如下图所示：



自定义损失函数

输入名称，并根据函数是否包含参数选择是否打开“参数”开关，在函数体部分输入自定义函数的内容，在描述部分输入对函数的描述或者备注信息，点击“确定”按钮，即可完成自定义损失函数的创建。

注意：函数“名称”的命名需要和函数入口名称一致。

损失函数内容如下：

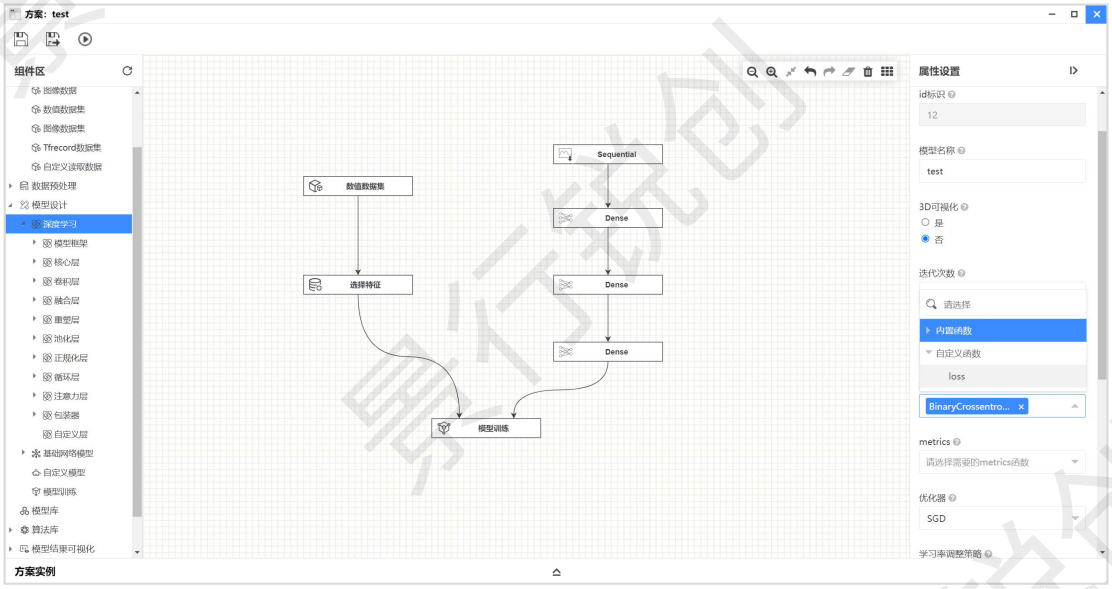
```
"""
Please input your custom metrics function here.
Note that the entry function name must be consistent
with the defined function name.
"""
import functools
import tensorflow as tf
from tensorflow.python.keras.losses import LossFunctionWrapper


def scaled_mean_square_error(y_true, y_pred, mean=0., std=1.):
    y_pred = (tf.convert_to_tensor(y_pred) - mean) / std
    y_true = (tf.cast(y_true, y_pred.dtype) - mean) / std
```

```
    return tf.keras.metrics.mean_squared_error(y_true, y_pred)


class my_loss_1(LossFunctionWrapper):
    def __init__(self, mean=0., std=1.,
name='scaled_mean_square_error'):
        super(my_loss_1,
self).__init__(functools.partial(scaled_mean_square_error, mean=mean,
std=std), name=name)
```

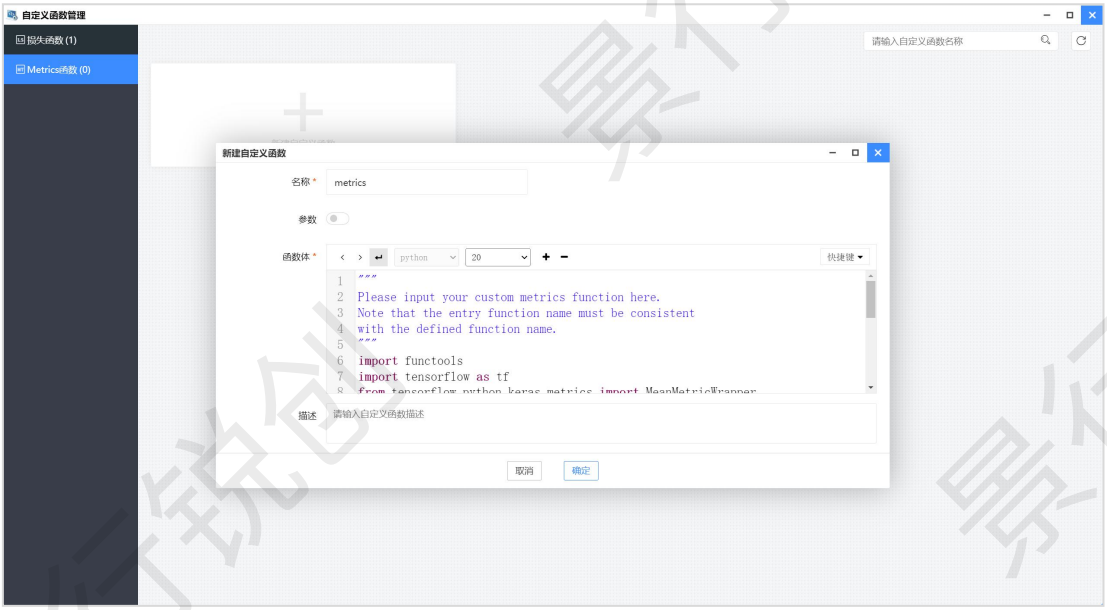
损失函数创建完成后在方案设计中训练深度学习方案结构时可以引用此函数，此函数会在自定义函数中显示。



属性区显示损失函数

➢ Metrics 函数

选择左侧“Metrics 函数”，点击“新建自定义函数”卡片，弹出新建窗口，如下图所示：



自定义 Metrics 函数

输入名称，根据函数是否包含参数选择是否打开“参数”开关，在函数体部分输入自定义函数的内容，在描述部分输入对函数的描述或者备注信息，点击“确定”按钮，即可完成自定义 metrics 函数的创建。

注意：函数“名称”的命名需要和函数入口名称一致。

metrics 函数内容如下：

```
"""
Please input your custom metrics function here.
Note that the entry function name must be consistent
with the defined function name.
"""
import functools
import tensorflow as tf
from tensorflow.python.keras.metrics import MeanMetricWrapper


def mean_scaled_relative_error(y_true, y_pred, epsilon=1.):
    y_pred = tf.convert_to_tensor(y_pred)
    y_true = tf.cast(y_true, y_pred.dtype)
```

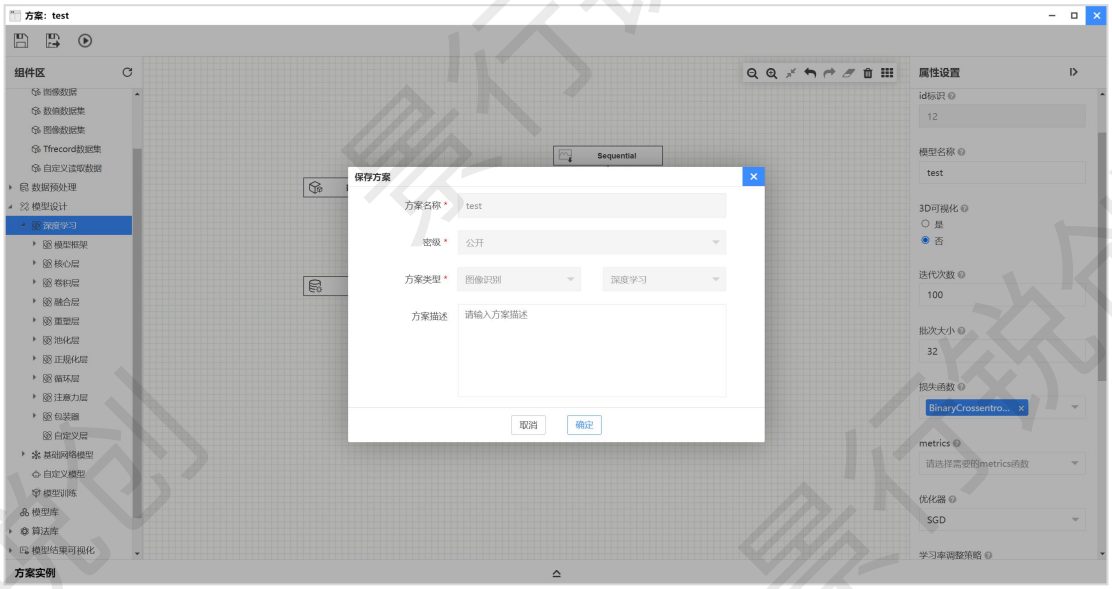
```
    return tf.keras.backend.mean(tf.abs(y_pred - y_true) /
(tf.abs(y_true) + epsilon), axis=-1)


 class my_metrics_1(MeanMetricWrapper):
    def __init__(self, epsilon=0.1,
name='mean_scaled_relative_error', dtype=None):
        super(my_metrics_1,
self).__init__(functools.partial(mean_scaled_relative_error,
epsilon=epsilon), name, dtype=dtype)
```

2.1.2.2 操作

2.1.2.2.1 方案保存

点击工具栏中的“保存”按钮，即可将方案描述和画布上的方案设计保存至数据库，当再次打开该方案时，画布上会显示最近一次保存的方案。“保存方案”页面，如下图所示：



保存方案页面

图中每个参数的具体含义如下：

模型名称：任何状态下均不可修改。

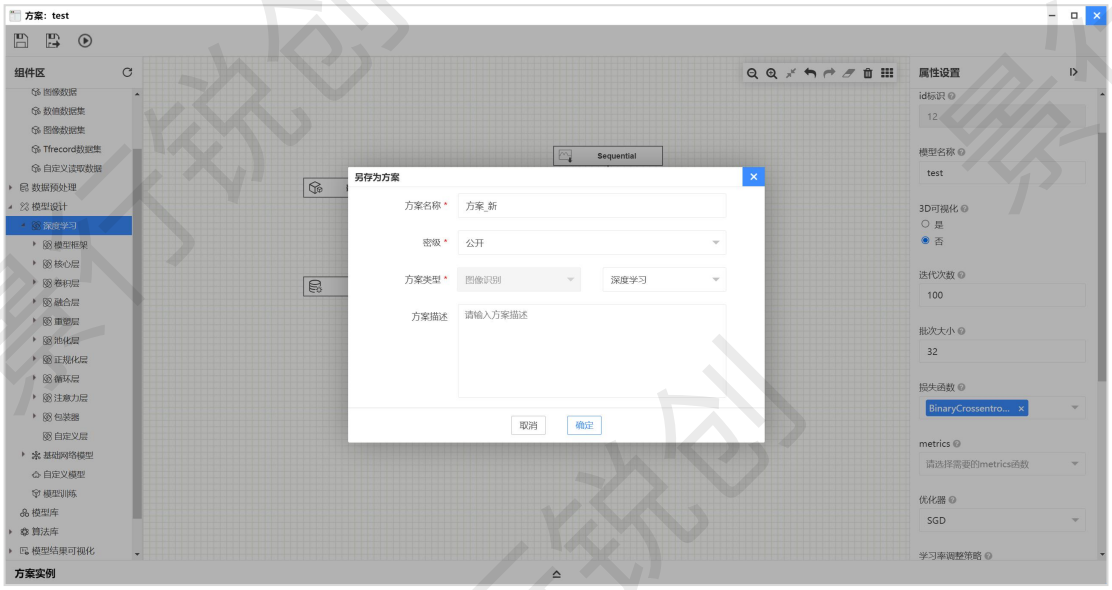
密级：任何状态下均不可修改。

模型类型：任何状态下均不可修改。

模型描述：用于描述方案的相关内容，选填。

2.1.2.2.2 方案另存为

点击工具栏中的“另存为”按钮，即可将方案另存为一个新的方案，方案可以是已保存的和未保存的。“另存为方案”页面。如下图所示：



另存为方案页面

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有的模型名称一致，必填。

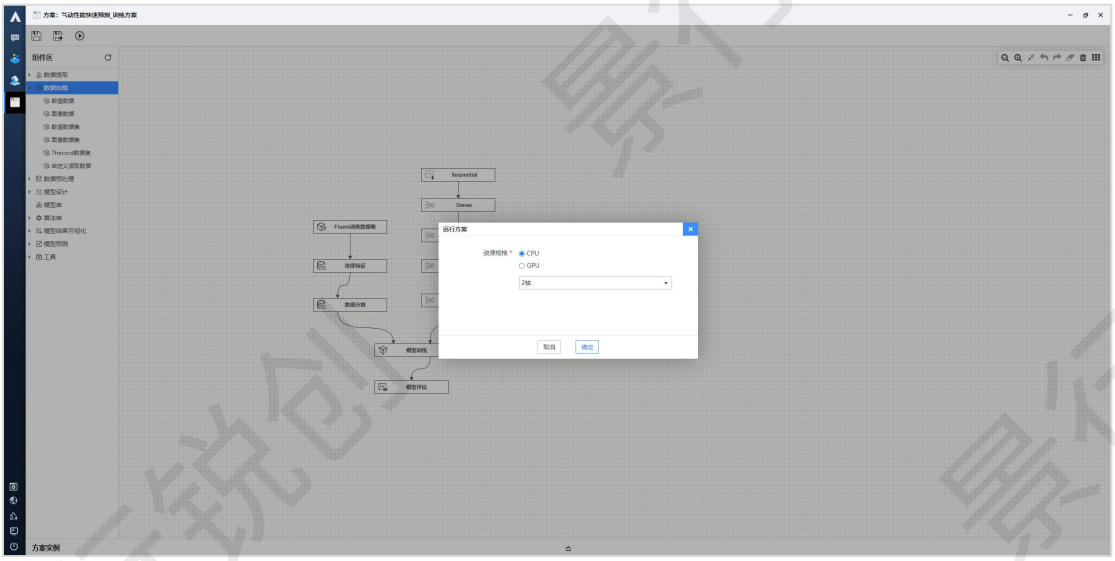
密级：对于另存的方案可根据需要修改密级。

方案类型：默认为当前方案的方案类型，可以选择，也可以自定义方案类型，必填。

方案描述：用于描述方案的相关内容，选填。

2.1.2.2.3 方案运行

方案设计完成后，点击“运行”按钮，弹出“运行方案”窗口，选择训练方案需要使用的资源，资源选择后，点击“确定”按钮，即可开始训练。如下图所示：



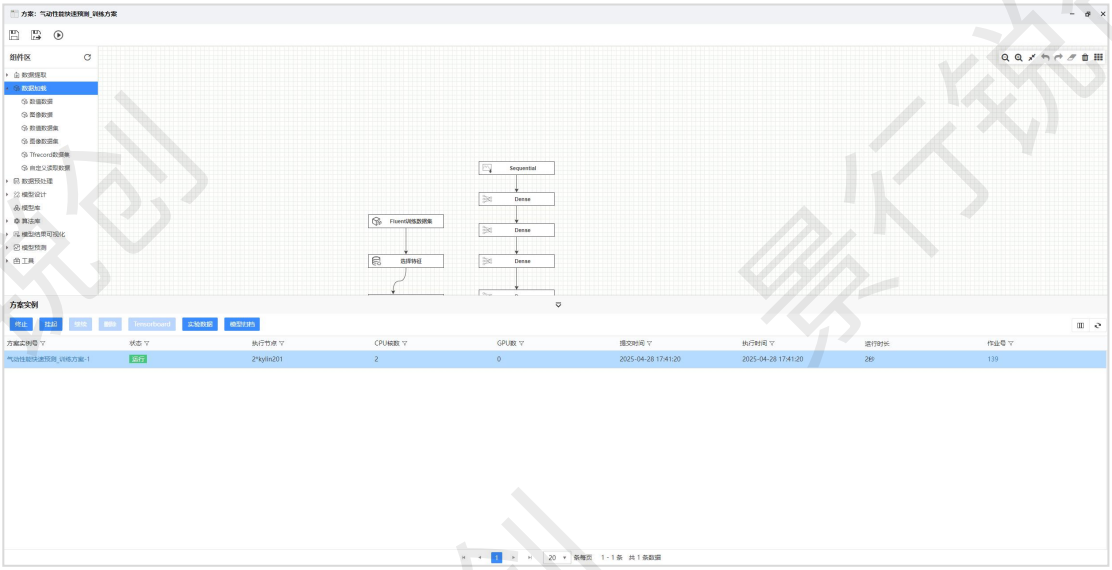
资源配置页面

点击运行按钮，打开选择资源页面，同时进行方案保存。选择资源后，点击确定按钮，即可开始训练方案。

注意：在方案设计页面中，点击“运行”按钮时，错误和问题对应如下：

- 画布中不存在组件时，提示“组件不能为空”。
- 组件的锚点没有连线、必填项属性为空时，提示“请检查异常组件并清除异常”。

开始训练方案后，会自动打开底部方案实例滑块，如下图所示：



方案实例滑块

底部方案实例滑块除了只能显示该方案的所有实例外，功能操作与方案实例一致，具体操作详情请看“方案实例”章节。

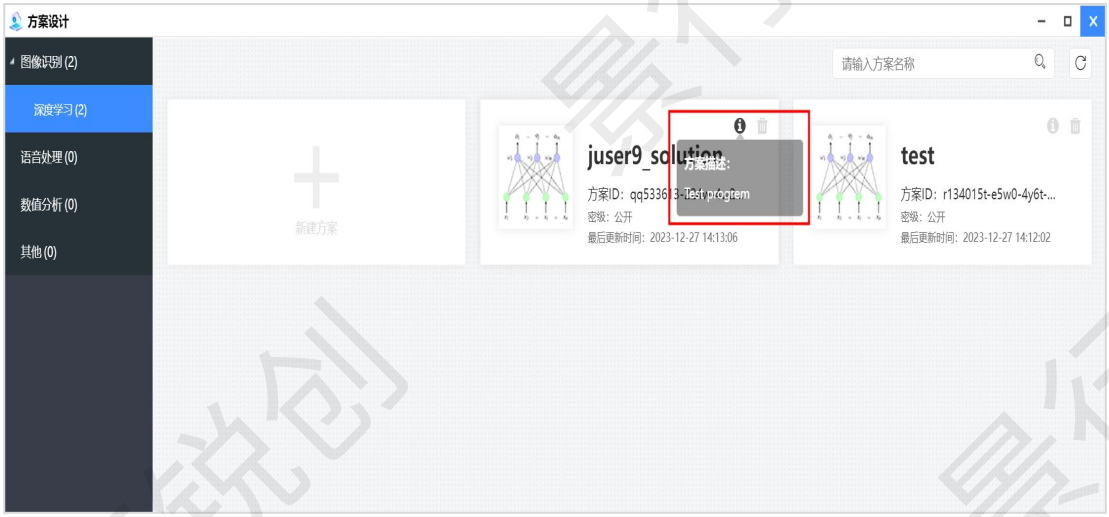
2.1.2.3 管理

方案新建成功后，在方案设计应用中会自动生成了一个方案卡片，卡片内容包含方案名称、方案 ID、最后更新时间，点击方案卡片的右上角功能图标，支持对方案进行“发布”，“查看描述”和“删除”操作。

方案设计管理

2.1.2.3.1 查看描述

鼠标移至方案卡片上的“描述”按钮时，可以查看方案的描述信息，如下图所示：



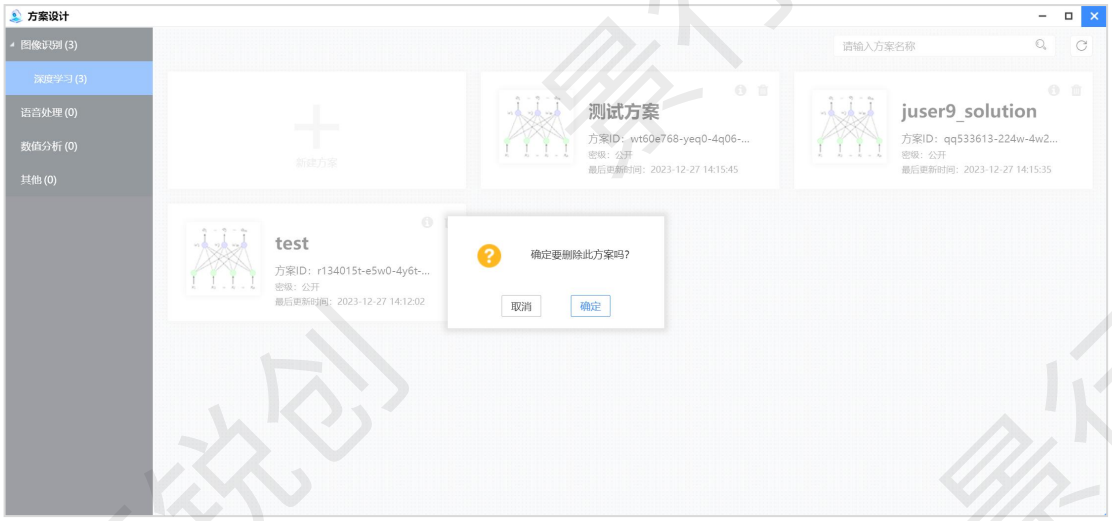
方案描述信息

点击“描述”的图标按钮，可在弹出窗口中修改方案的描述信息，如下图所示：

修改方案描述

2.1.2.3.2 删除

点击“删除”按钮，可以把该方案卡片删除。



删除方案

2.1.2.4 方案实例

方案实例中以列表的形式记录了所有方案的历史运行记录，用户可以通过实例号、名称等筛选功能快速过滤实例。

用户可在方案实例应用中查看自己的所有实例，或在具体的方案设计页面中，通过点击画布下方的方案实例滑块，在展开的面板中查看当前方案实例的信息，亦可对其进行终止、挂起、模型归档、Tensorboard 等操作。

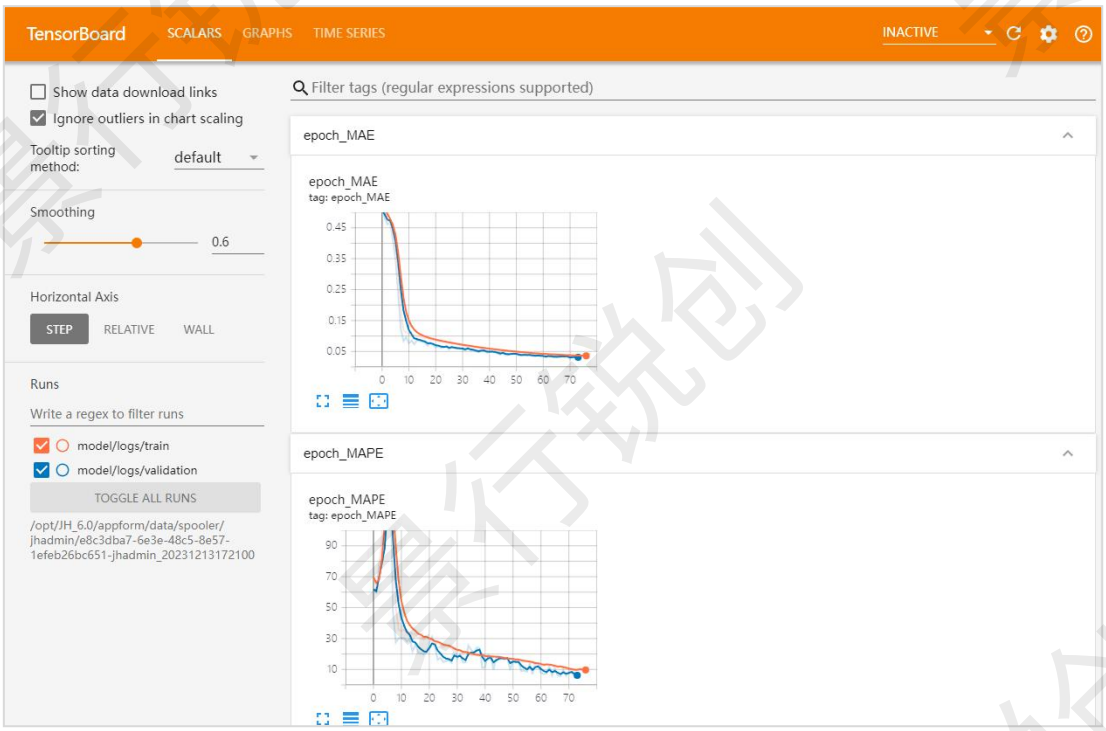
方案实例列表

- 功能按钮：

方案实例支持对方案实例进行“终止”“挂起”“继续”“删除”、Tensorboard、实验数据和模型归档操作。

当深度学习的方案运行完成后，点击“Tensorboard”按钮可以以图表形式查看当前实例运行结果。

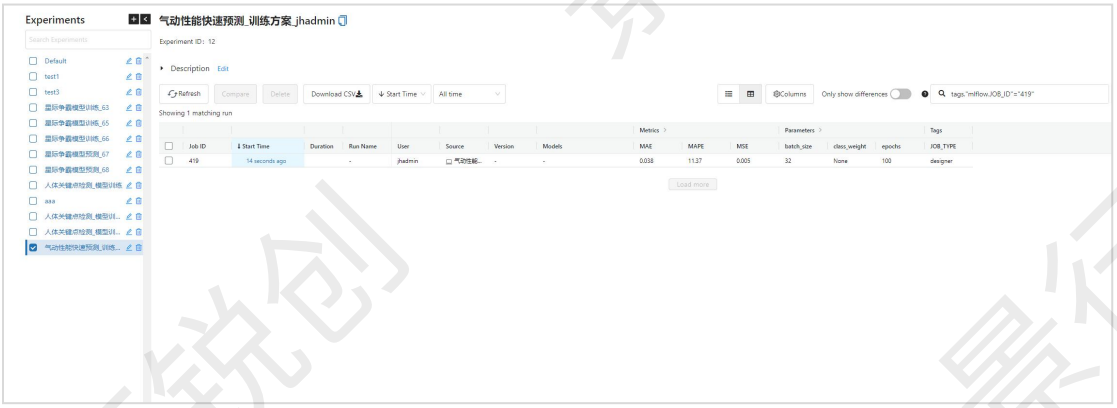
方案实例 Tensorboard 按钮



Tensorboard

模型训练的方案运行过程中，点击“实验数据”跳转到实验管理页面，查看该方案的训练过程参数。

方案实例实验数据按钮

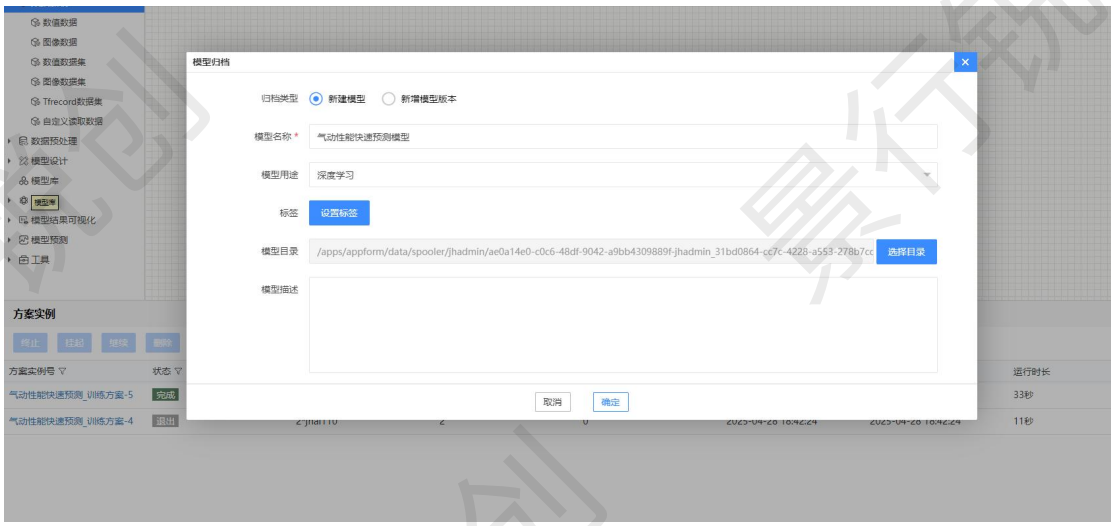


实验数据

模型训练方案运行完成后，点击“模型归档”按钮归档生成的模型到模型库中。

方案实例模型归档按钮

“模型归档”页面如下：



模型归档

归档类型：分为新建模型和新增模型版本。

当选择“新建模型”时，设置参数如下所示：

模型归档类型为新增模型

模型名称：输入归档模型的模型名称。

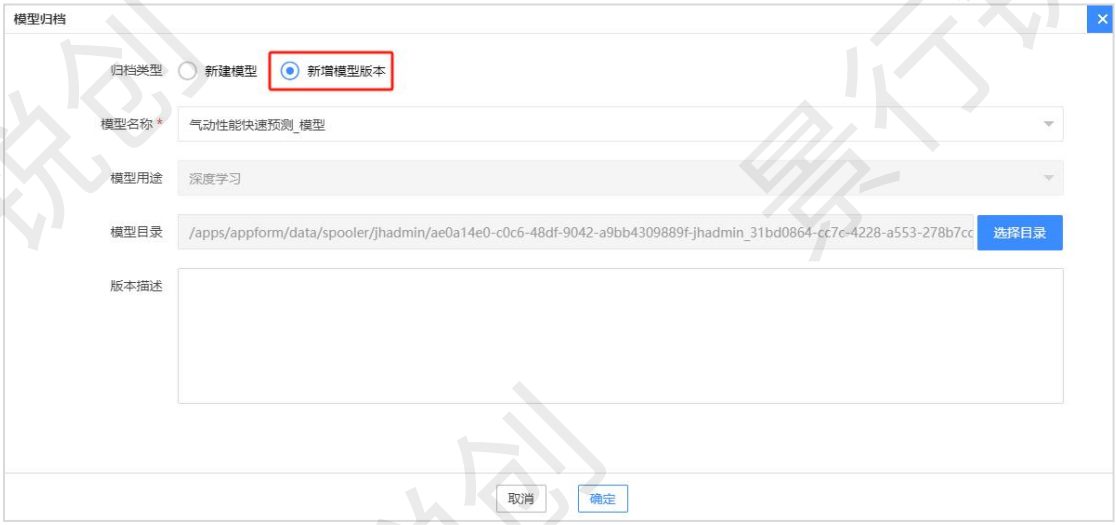
模型用途：选择模型用途，选择机器学习或者深度学习。

标签：设置模型的标签，标签可选内容和模型库标签内容一致。

模型目录：默认回填作业运行 spooler 目录，也可以选择 spooler 目录下的子目录。

模型描述：输入归档模型的描述信息。

当选择归档类型选择“新增模型版本”时，模型归档页面如下：



模型归档类型为新增模型版本

模型名称：选择模型名称，模型名称来自模型库中所有的机器学习模型和深度学习模型。

模型用途：模型用途会根据模型名称自动回填模型名称相应的模型用途。

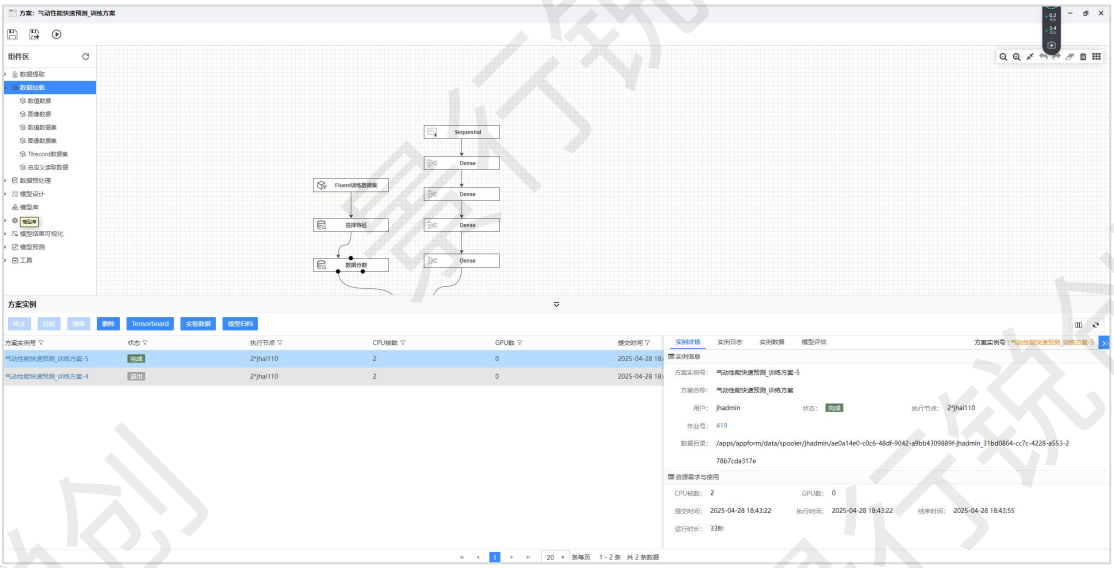
模型目录：默认回填作业运行 spooler 目录，也可以选择 spooler 目录下的某个子目录。

版本描述：添加对新增版本模型的描述信息。

➢ 方案实例滑块：

点击“方案实例号”或者双击方案实例列表中的某条记录，会打开右侧滑块。滑块有三个标签页，分别为实例详情、实例日志和实例数据。

当组件中包含“模型评估”组件，且方案运行成功，滑块会有四个标签页，分别为实例详情、实例日志、示例数据和模型评估。如下图所示：



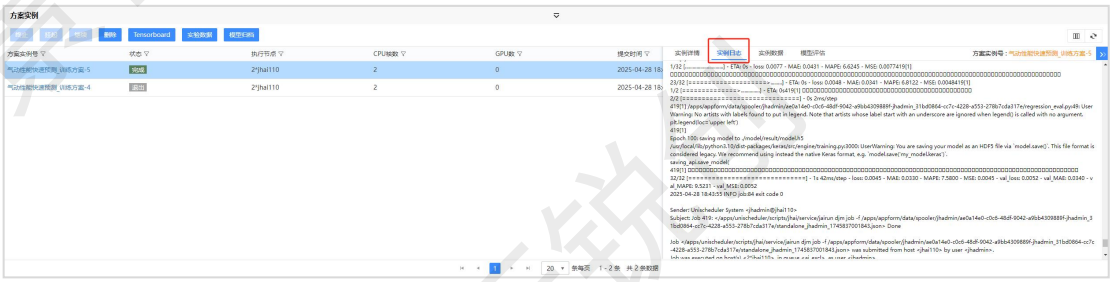
方案设计

实例详情：显示方案实例的基本信息、资源信息。点击作业号，可以查看该实例关联的作业信息。实例详情中包含方案实例关联的数据集信息，包括数据集名称、数据集类型、字符集、分隔符以及数据保存目录。实例详情中包含方案实例关联的模型信息，实例详情中会显示模型信息，包括模型名称、模型类型、超

参数(仅深度学习模型)。

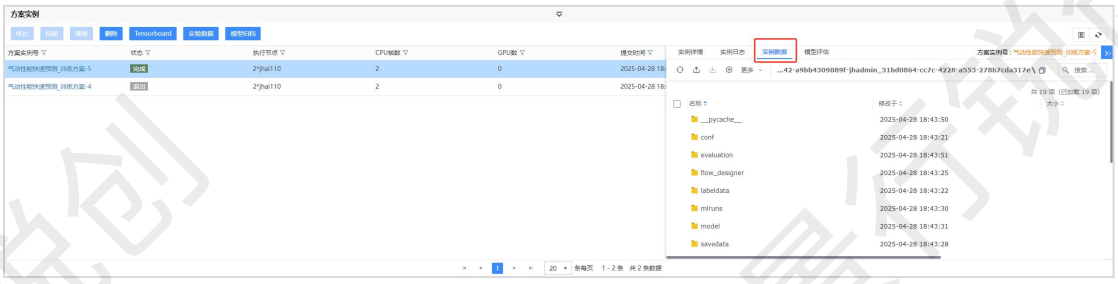
方案实例滑块

实例日志：点击实例日志按钮，打开实例日志界面。方案实例的状态处于正在运行时，该实例日志会实时输出。方案实例的状态处于完成时，会显示该方案实例的所有日志。



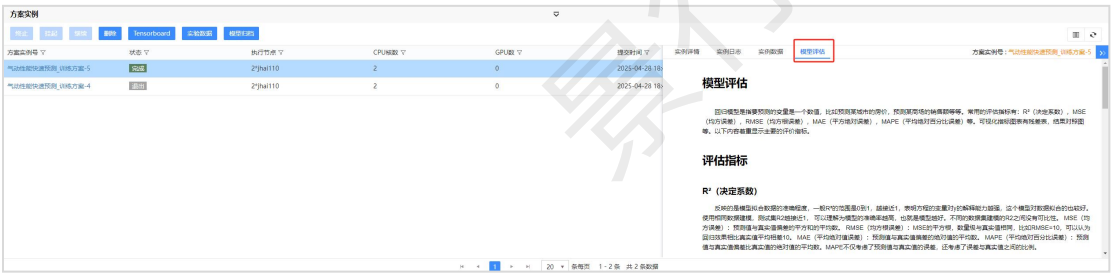
方案实例日志

实例数据：点击实例数据按钮，打开实例数据界面，显示该方案实例包含的所有相关文件。



方案实例数据

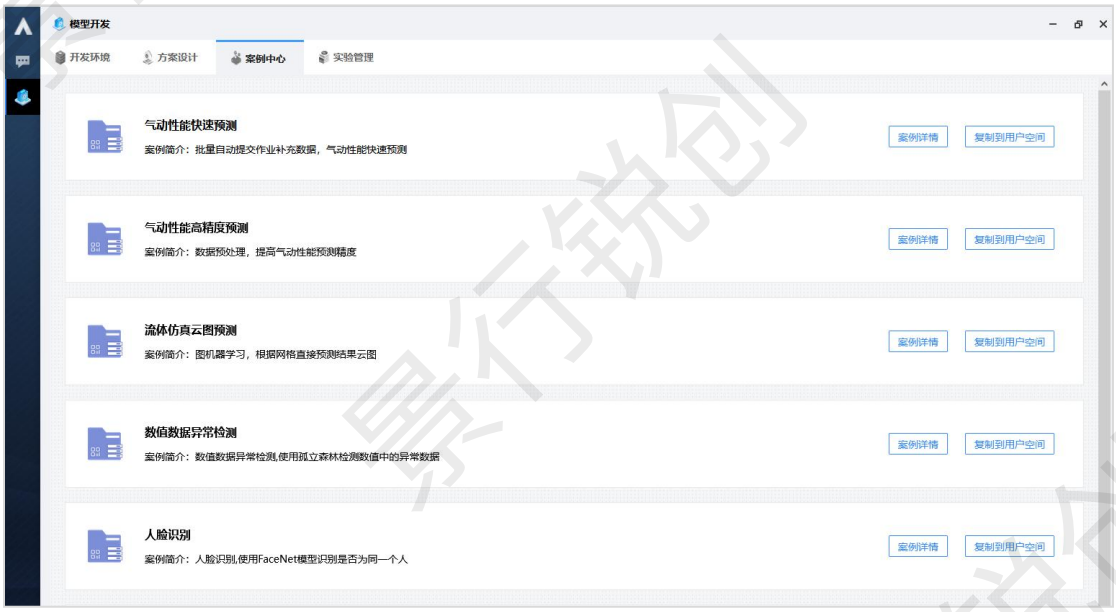
模型评估：点击模型评估按钮，打开模型评估界面，显示该模型的评估结果，如下图所示：



方案模型评估

2.1.3 案例中心

案例中心内置了一些AI案例，方便用户了解AI平台功能的基本使用和流程，用户可以复制案例中心的内置案例到自己的用户空间，查看和运行相关案例。桌面双击“模型开发”应用，选择“案例中心”。如下图所示：



案例中心

2.1.3.1 案例详情

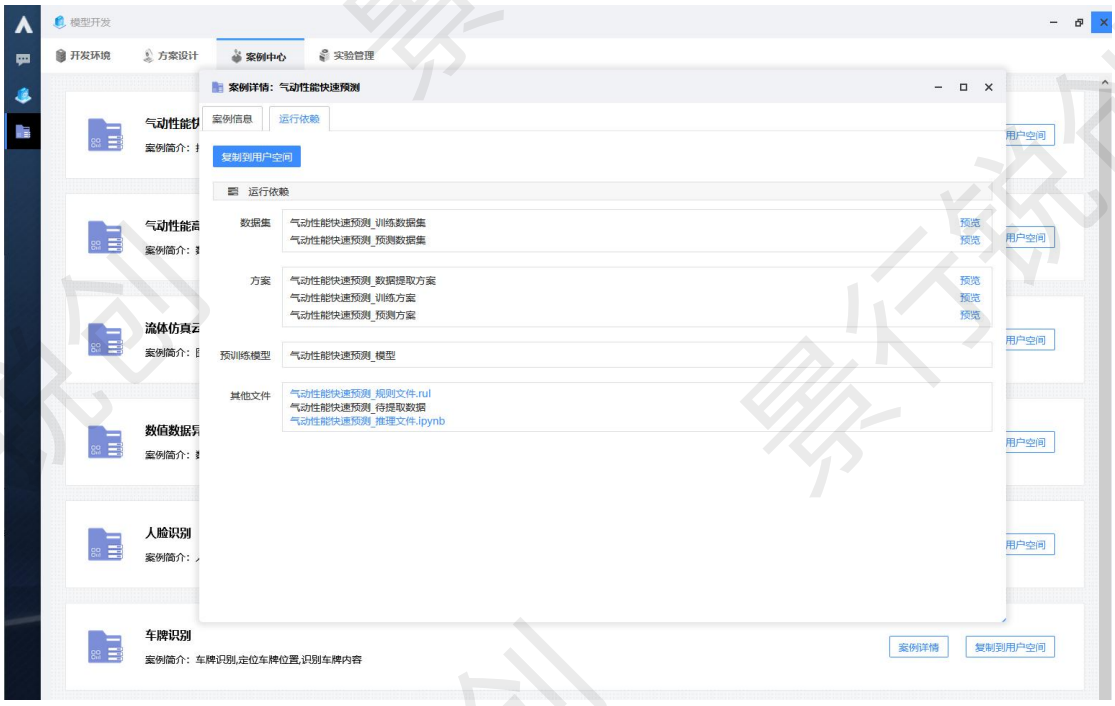
目前案例中心集成了 10 个案例，包括：气动性能快速预测、气动性能高精度预测、流体仿真云图预测、数值数据异常检测、人脸识别、车牌识别、火灾检测、行人轮廓检测、人体关键点检测和强化学习。

案例详情展示了案例的详细介绍和使用方法。以“气动性能快速预测”案例为例，操作步骤如下所示：点击“气动性能快速预测”案例卡片上的“案例详情”按钮，弹出“案例详情”窗口，如下图所示：

案例详情

案例信息界面详细描述了案例的模型背景、数据准备、模型训练、模型效果和硬件要求，用户可以按照该步骤使用案例。

切换至“运行依赖”标签页，用户可以查看案例所需要的数据、方案、模型等依赖。如下图所示：

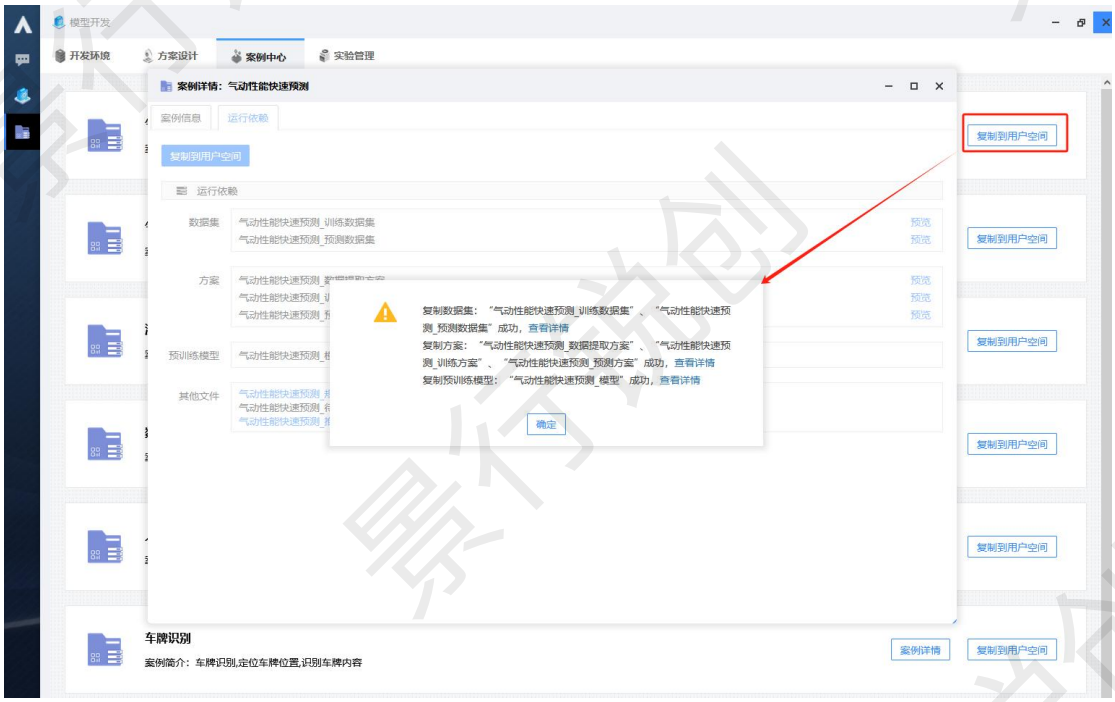


运行依赖页面

其中，“预览”按钮，可以查看数据集的数据或者方案的方案结构；“其他文件”框中提供了推理文件，支持下载。

2.1.3.2 案例使用

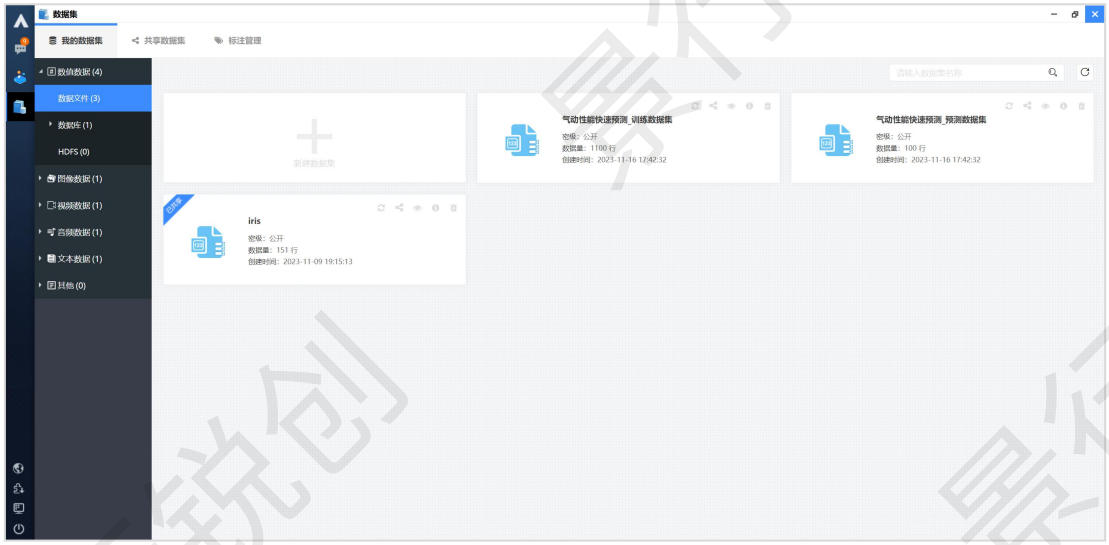
点击“复制到用户空间”按钮，可以将案例依赖的所有数据文件复制到用户自己的环境中。以“气动性能快速预测”案例为例，操作步骤如下所示：点击“气动性能快速预测”案例卡片右边的“复制到用户空间”按钮，将案例复制到自己的空间，如下图所示：



复制到用户空间

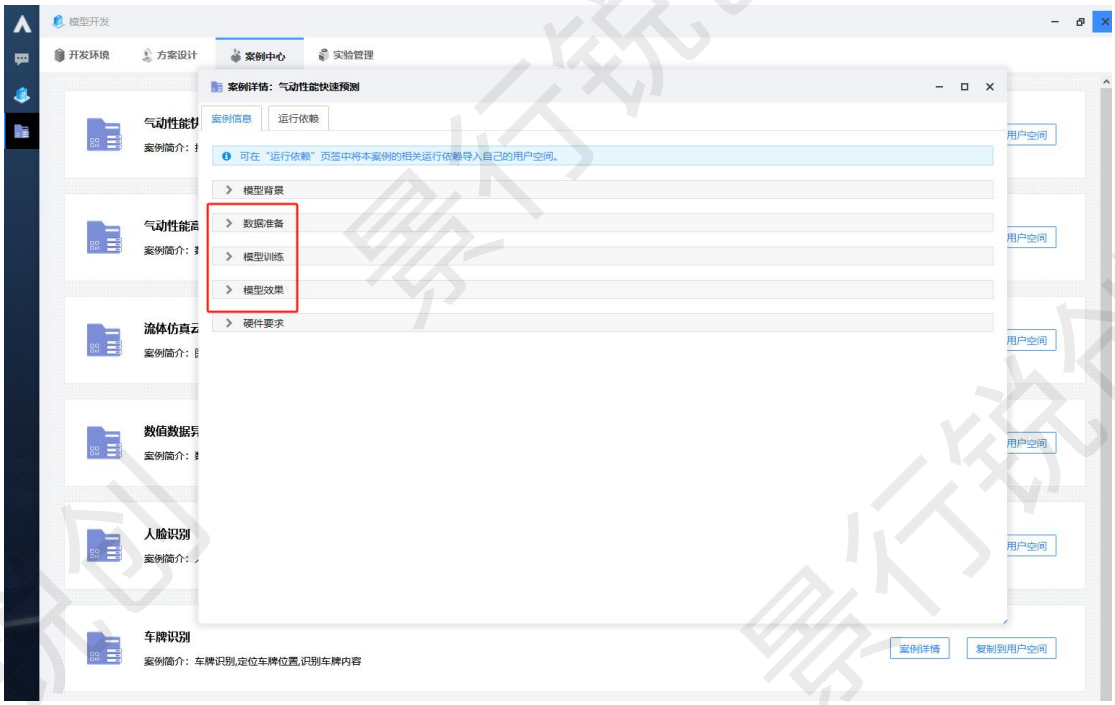
注意：案例的每次复制都会覆盖用户本地的同名数据集、方案和模型。因此用户可以将修改过的方案进行另存为，防止被覆盖导致的丢失。

可以点击提示中的“查看详情”跳转链接，查看复制过来的数据集、方案或者模型。以查看复制过来的数据集为例，操作步骤如下所示：点击复制数据集项的“查看详情”跳转链接，可以在“数据集管理”-“我的数据集”中，查看数据集信息，如下图所示：

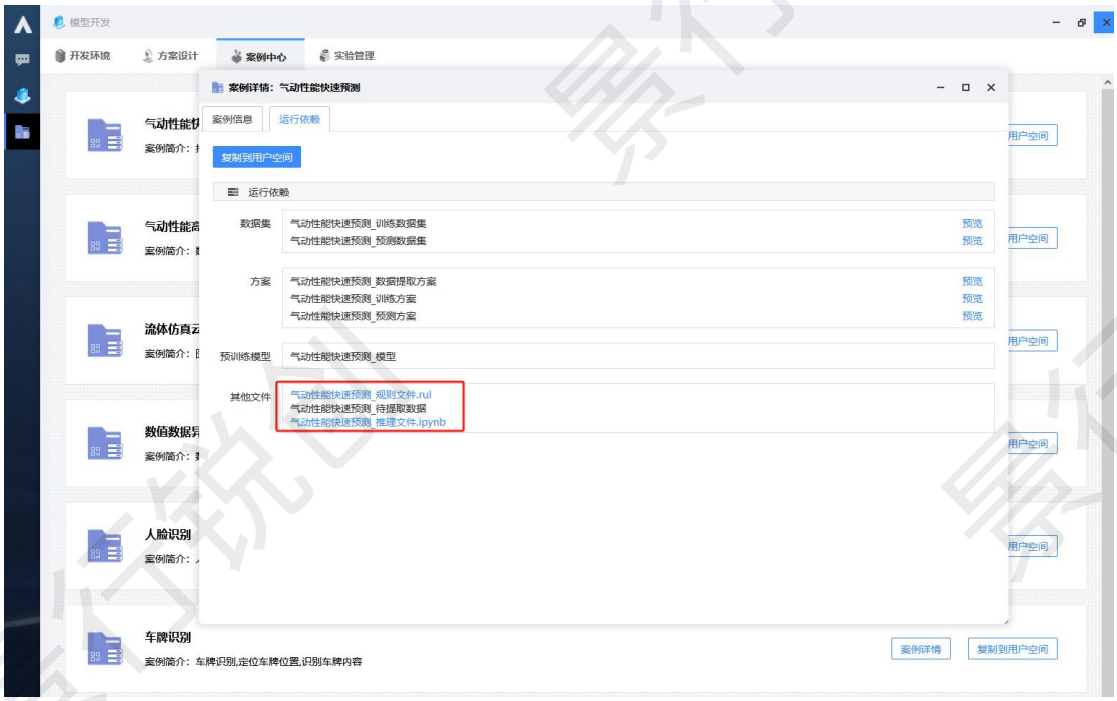


查看数据集信息

根据“案例详情”中的步骤：“数据准备”-“模型训练”-“模型效果”，依次准备数据、训练数据、部署服务、推理数据即可，推理文件在“运行依赖”中下载，如下图所示：



查看案例信息



推理文件下载

2.1.4 实验管理

实验管理为景行人工智能平台提供跟踪、比较和记录模型训练指标的工具，训练指标包括训练的参数，超参数，评估结果，模型文件等，用户可以通过实现管理界面查看和对比多次实验的数据。

涉及的一些关键字：

实验（experiment）：相当于项目名，所有的调参都是针对这个实验。

运行（Run）：实验的一次运行记录。

参数(parameter)：实验运行的参数、超参数。

指标(metric)：实验运行产生的指标结果。

文件(artifact)：记录**实验运行相关的**文件，可以包括代码、日志或保存的模型。

2.1.4.1 示例程序

如果需要使用实验管理的功能，需要在代码中集成 libaiflow API，目前仅支持 Python。

自动记录：Tensorflow 和 Pytorch 框架，支持通过 libaiflow.tensorflow.autolog 和 libaiflow.pytorch.autolog 提供的 Callback 机制实现自动收集 tensorflow 和 pytorch 实验的参数，指标等多种实验训练数据。下面展示 autolog 自动收集数据的关键代码，（完整样例代码参考附录四样例一）。

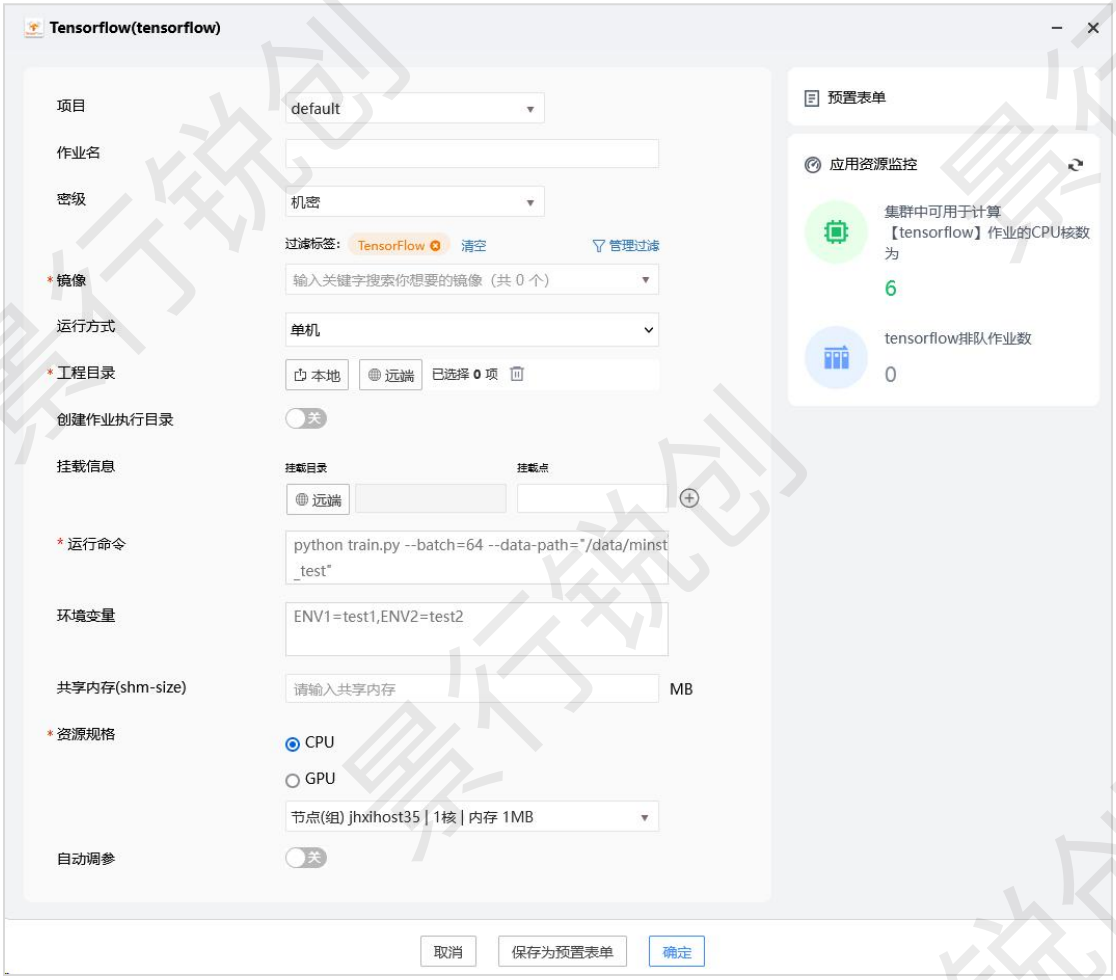
```
import libaiflow
expt = libaiflow.init(experiment_name="mnist_tf2_cpu")
libaiflow.tensorflow.autolog()
```

手动记录：通过调用指标和超参数相关的 api 记录和收集数据（完整样例代码参考附录四样例二）。

```
import libaiflow
expt = libaiflow.init(experiment_name="pytorch-sample")
……
with libaiflow.start_run(experiment_id=expt.experiment_id):
    lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
    lr.fit(train_x, train_y)
    predicted_qualities = lr.predict(test_x)
    (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
    print("Elasticnet model (alpha=%f, l1_ratio=%f):" % (alpha,
l1_ratio))
    print("  RMSE: %s" % rmse)
    print("  MAE: %s" % mae)
    print("  R2: %s" % r2)
    libaiflow.log_param("alpha", alpha)
    libaiflow.log_param("l1_ratio", l1_ratio)
    libaiflow.log_metric("rmse", rmse)
    libaiflow.log_metric("r2", r2)
    libaiflow.log_metric("mae", mae)
```

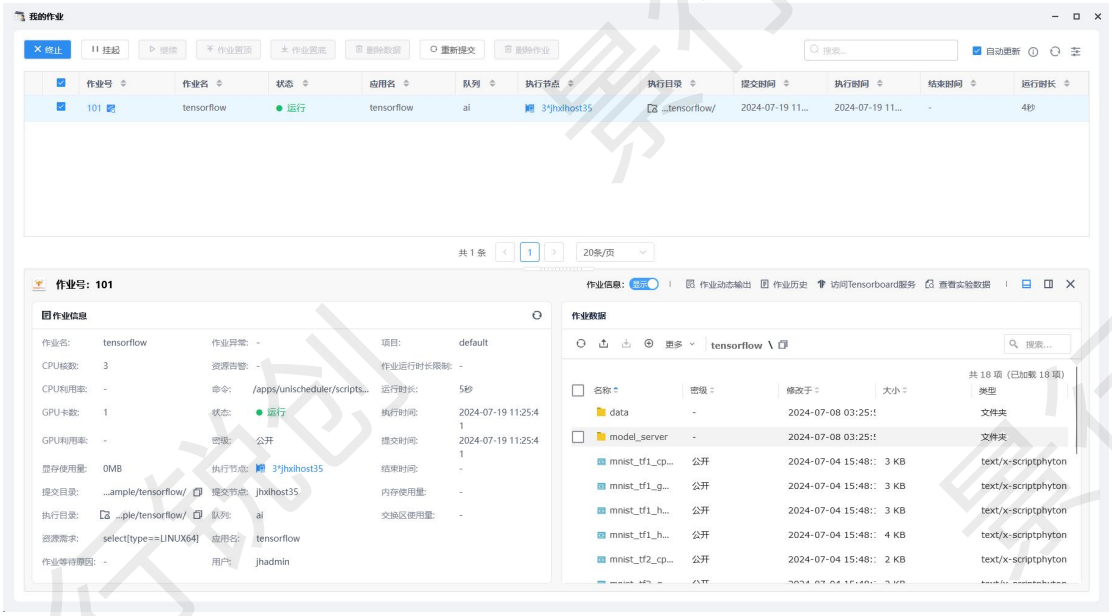
2.1.4.1.1 提交 AI 作业

景行人工智能的内置训练框架镜像中已经默认安装 Libaiflow SDK，如果用户的模型训练代码中，已经集成了 libaiflowSDK，可以通过 Web 页面提交 AI 作业进行模型训练。以提交 Tensorflow 作业为例，如下图所示：



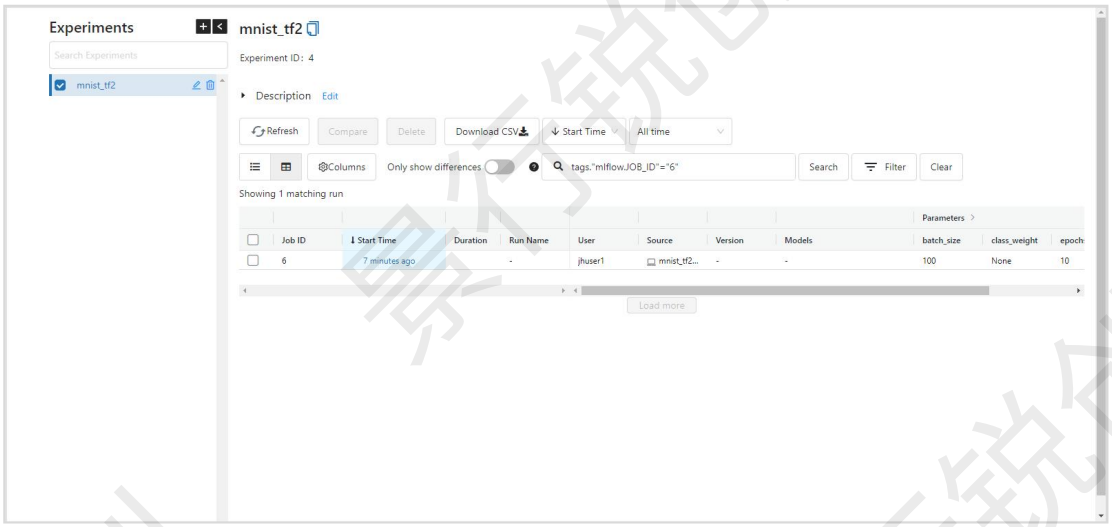
提交 AI 作业

Tensorflow 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



注意：仅当模型训练代码中调用了 libaiflow SDK，才能查看实验数据。

点击“查看实验数据”按钮，可以查看实验数据，如下图所示：

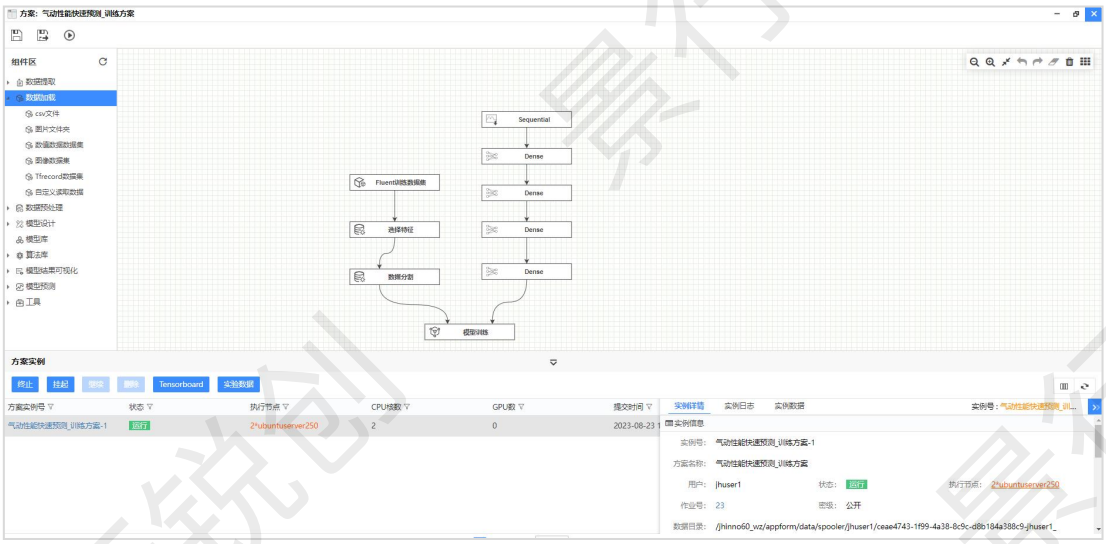


查看实验数据

2.1.4.1.2 可视化建模

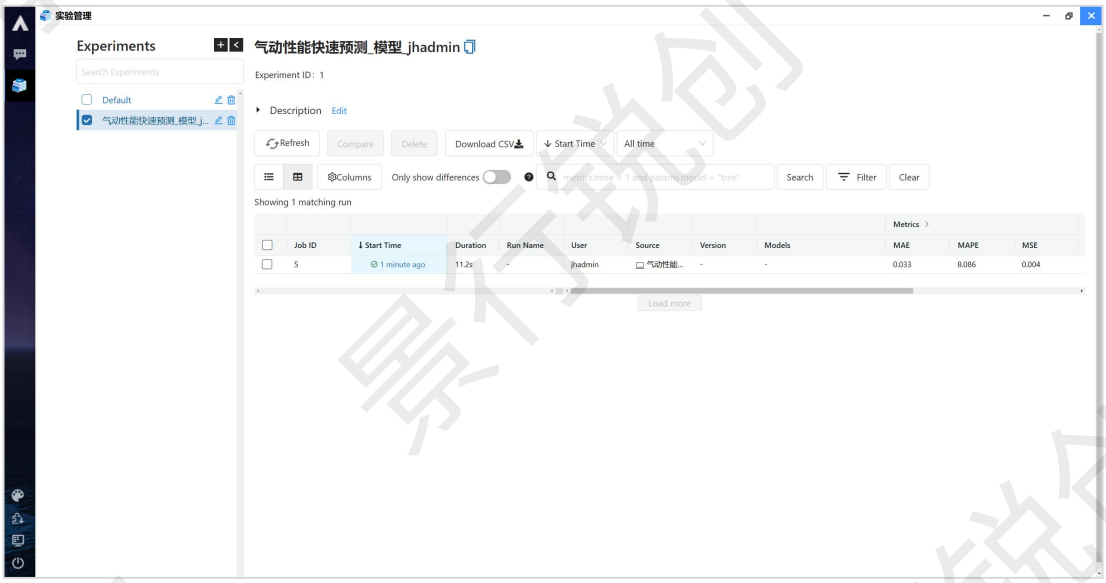
使用可视化建模方式构建的方案，已默认集成了 Libaiflow SDK，会自动记录相关的参数和指标数据

打开方案设计管理，进入方案设计器界面，构建模型结构流程、训练模型，如下图所示：



运行方案

点击“实验数据”，可以打开实验管理页面，如下图所示：

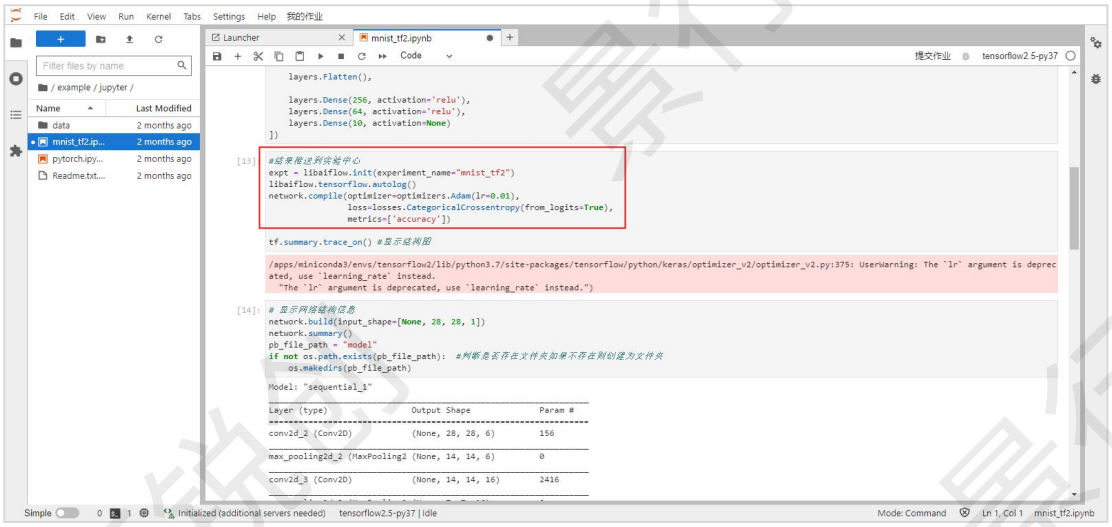


查看实验数据

2.1.4.1.3 WebIDE

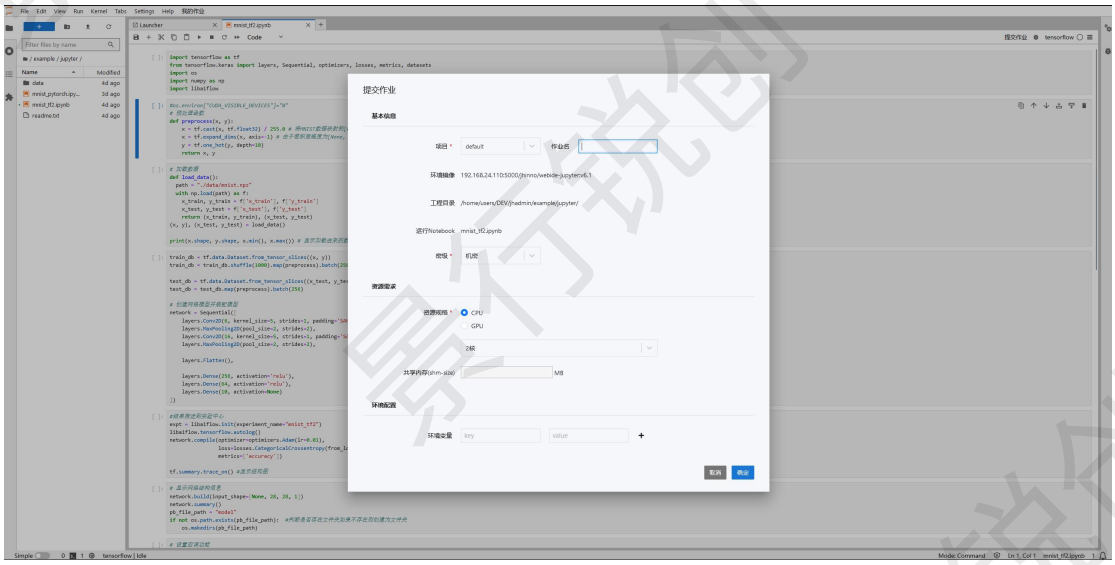
WebIDE 类型的开发环境已经默认安装 libaiflow SDK，在 WebIDE 中运行集成了 libaiflow SDK 的代码时，会记录相关的实验数据，

访问 Jupyter 服务，运行集成了 libaiflow SDK 的代码，会推送参数和指标到实验管理。如下图所示：



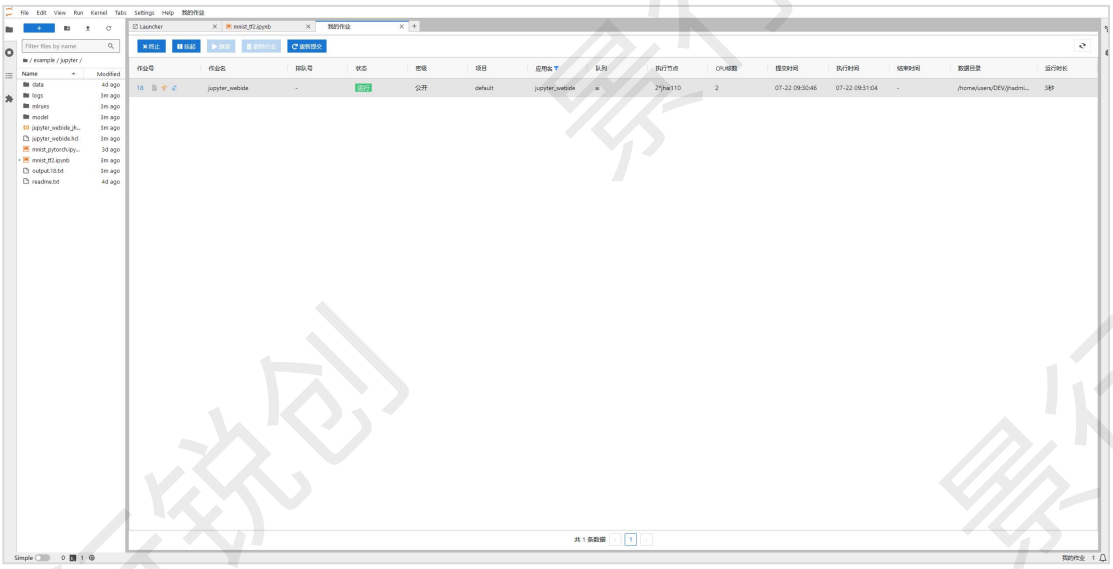
“Jupyter”服务

点击“提交作业”按钮，提交作业，如下图所示：



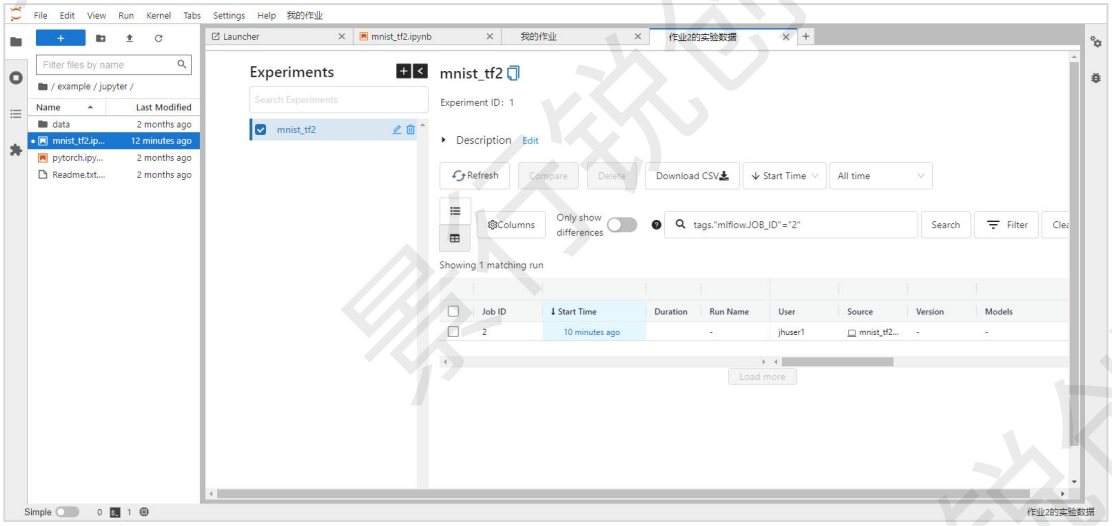
提交作业

点击“我的作业”按钮，打开我的作业 tab 页，如下图所示：



我的作业

点击作业实例上的“实验管理”图标按钮，查看实验数据，如下图所示：



查看实验数据

2.2 AI 模型训练

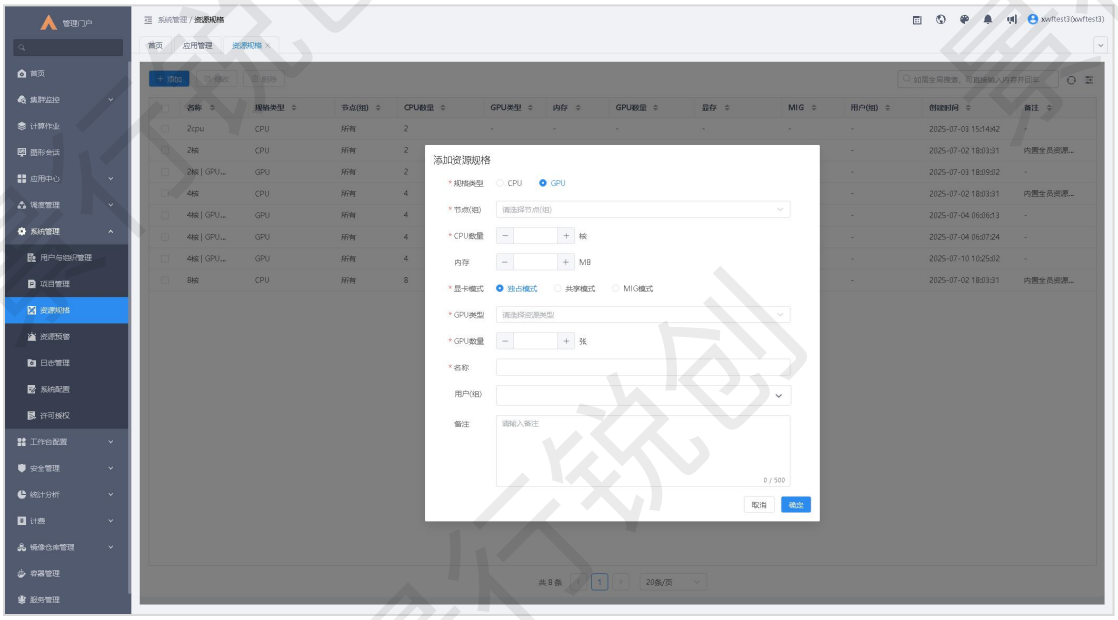
景行人工智能平台集成了主流深度学习框架，包括 Tensorflow、Pytorch、Mxnet、PaddlePaddle、MindSpore 和 DeepSpeed，并为用户提供了直观的任务提交 Web 页面和命令行方式提交。

通过 Web 门户，用户能够方便地提交训练作业，灵活指定训练所需的资源规格，包括 CPU、内存、GPU、GPU 类型。用户提交完成后可以在作业管理中，实时

查看训练进度，训练日志，以及访问 Tensorboard、VisualDL、MindInsight 等可视化服务。并且平台支持多种框架的并行训练，同时为用户提供了对不同资源规格的选择，包括单机单 GPU、单机多 GPU，以及多机多 CPU。

在创建资源规格时，支持指定独占、共享和 MIG 三种模式，允许指定资源的提交节点（组），更加细粒度地控制资源分配情况，并且对于不同的资源可以指定部分用户（组）可见。

在管理门户中，选择“系统管理”下的“资源规格”来创建，如下图所示：



添加 AI 资源规格

资源规格参数：

规格类型：选 CPU 或 GPU 主导资源分配，区分计算任务类型。

节点（组）：任务运行的节点分组，决定调度目标。可在调度管理下的节点与节点组中创建。

CPU 数量：需要申请的 CPU 核心数。

内存：以 MB 为单位的内存容量，满足程序运行需求。

显卡模式：独占（独享 GPU）、共享（多任务共用）、MIG（切分 GPU 为独立实例），适配不同 GPU 复用需求。

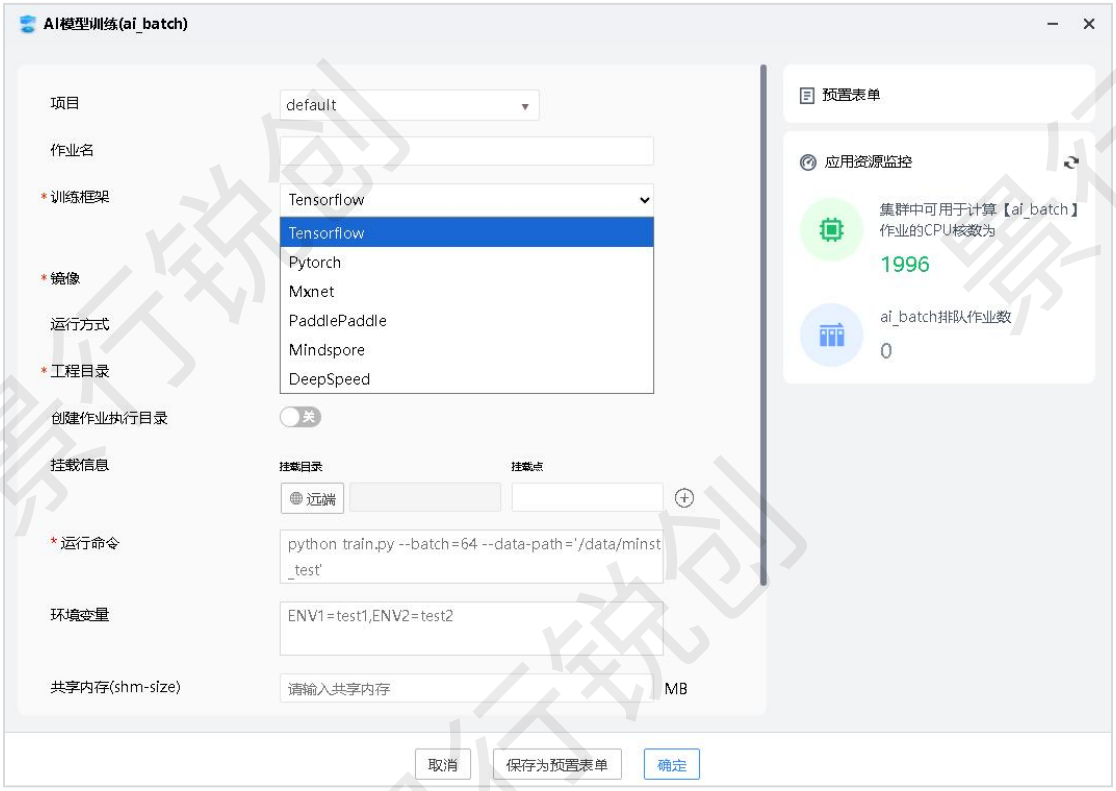
GPU 类型：具体 GPU 型号，匹配任务对算力、特性的要求。

GPU 数量：申请的 GPU 卡数，支撑多卡并行任务。

名称：自定义标识，方便区分、管理资源申请。

用户（组）：资源归属的用户/组，做权限与配额管控。

点击“AI 模型训练（ai_batch）”桌面图标，AI 模型训练参数设置界面，从训练框下拉列表中选择训练框架，切换到不同类型训练框架，如下图所示：



AI 模型训练

命令行提交作业，用户在 sub 脚本中设置队列、资源、镜像等参数，如下所示：

```
#!/bin/sh

#BSUB -J tensorflow_standalone_cpu（作业名称）
#BSUB -q ai（队列名称）
#BSUB -app jhai_command （应用名称）
#BSUB -mf standalone_cpu.mf （machine 文件，用于设置 cpu 数，slot 数和 gpu 数量）
#BSUB -o output.%J.txt （输出日志名称）
```

```
 ${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/tensorflow:2.13 --cmd='python mnist_tf2.py'
--workdir=${HOME}/example/tensorflow （运行命令）
```

machine 文件示例如下：

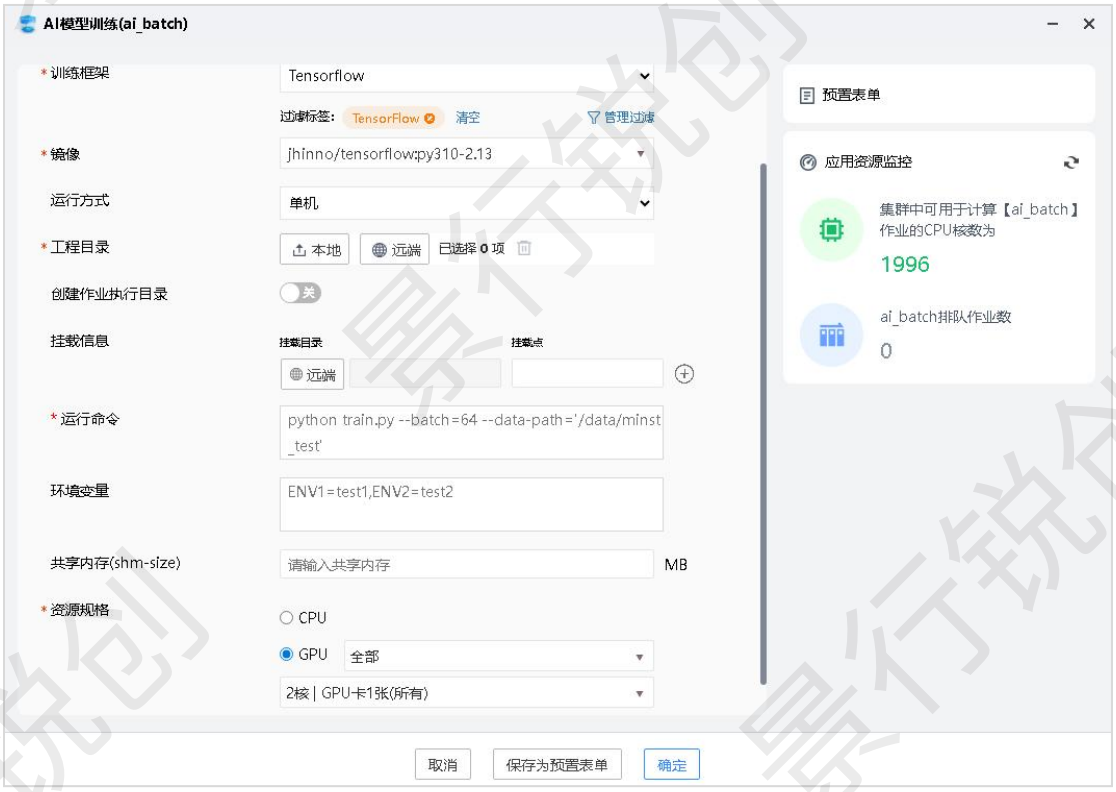
```
[host:all slots:1 gpu:1] （host 运行节点名称，slots 数，gpu 数）
```

2.2.1 Tensorflow

2.2.1.1 可视化方式提交

2.2.1.1.1 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“Tensorflow”，如下图所示：



Tensorflow 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：tensorflow。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Tensorflow”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“Tensorflow”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) 挂载目录：指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) 挂载点：挂载目录在容器中对应的目录，未填写时挂载点和挂载目录的路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见本章节的“自动调参”模块。

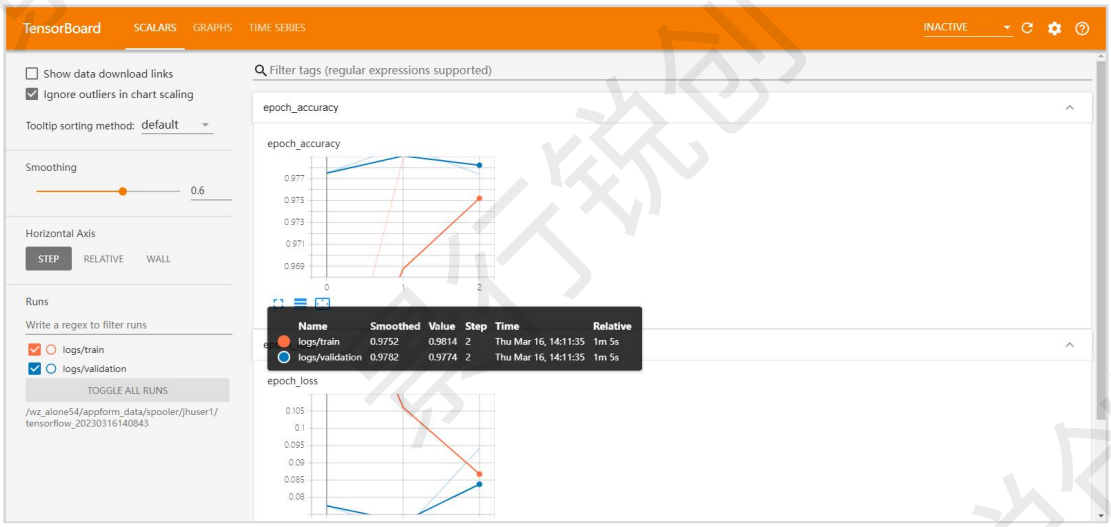
预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

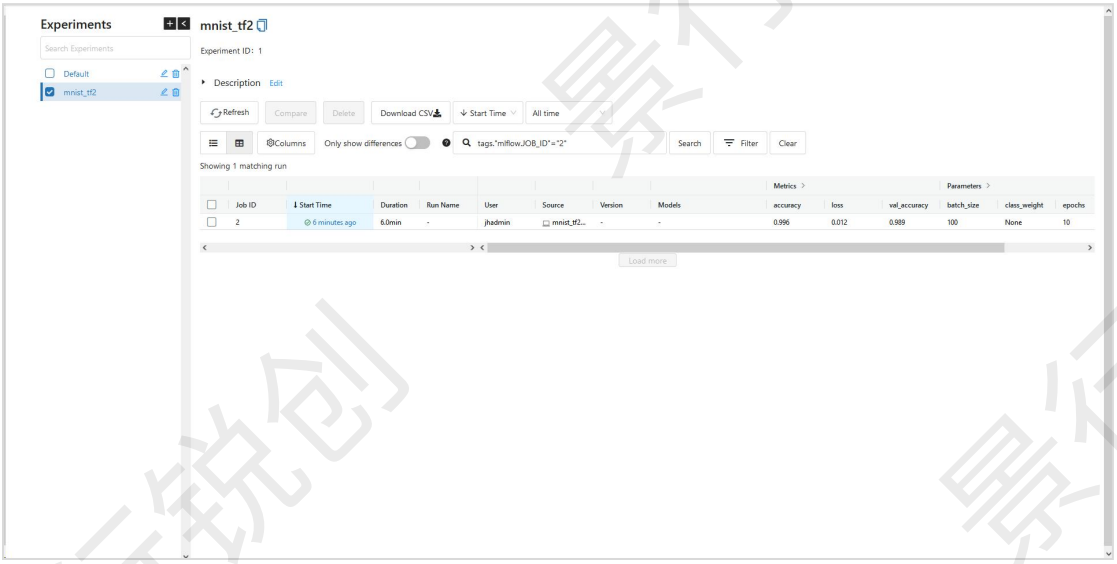
Tensorflow 作业提交后，会自动打开“我的作业”页面，方便用户在此查

看提交作业的信息，如下图所示：

点击作业详情右上角“访问 Tensorboard 服务”按钮，可以查看训练情况，如下图所示：



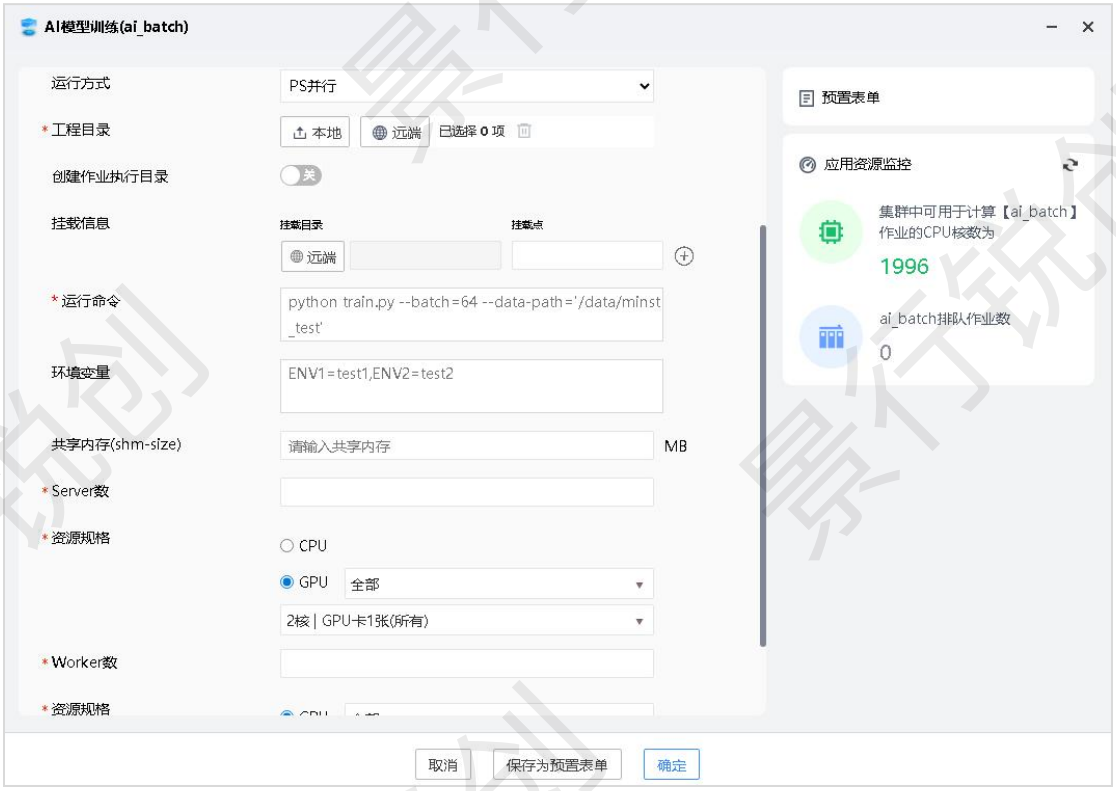
如果运行程序中集成了实验管理 api，点击作业详情右上角“查看实验数据”按钮，可以查看实验数据，如下图所示：



2.2.1.1.2 PS 并行模式

提交以 Parameter Server 方式实现的训练代码，支持跨节点，需填写并行相关参数，如下所示：

- **Server 数：**指定参数服务的数量。
- **Worker 数：**指定训练服务的数量。
- **资源规格：**分别指定每个 Server 和 Worker 使用的资源。

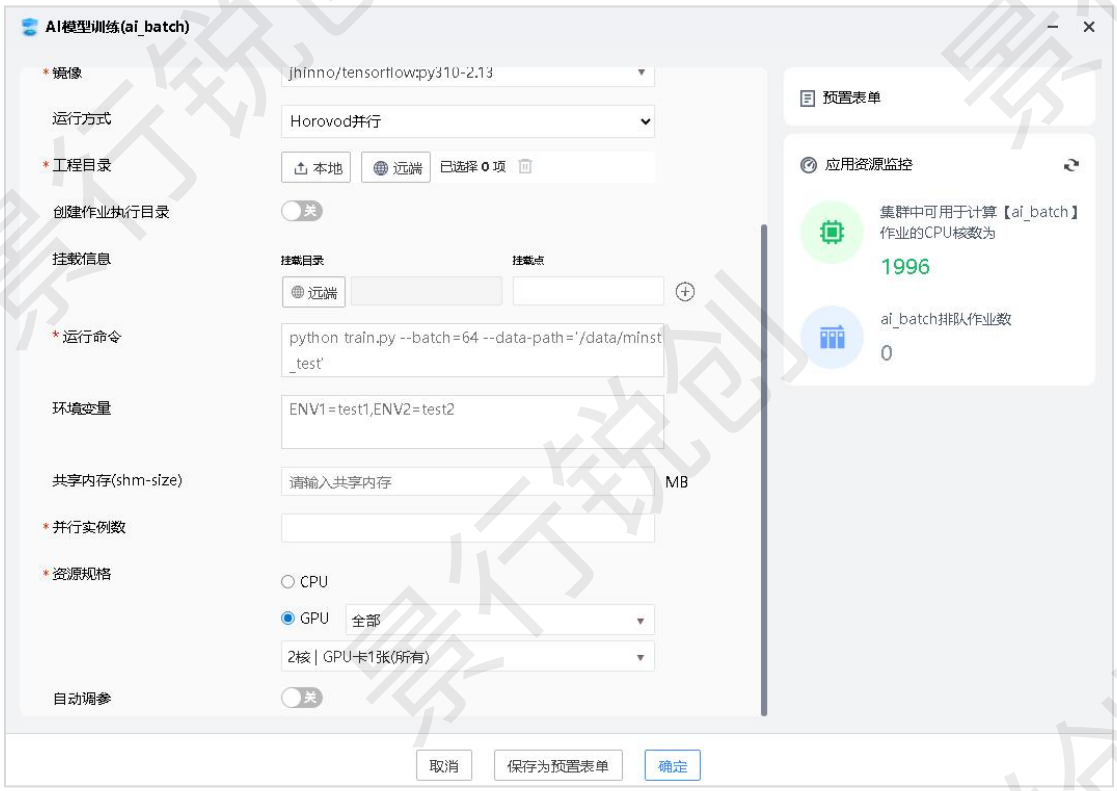


Tensorflow PS 并行模式作业提交界面

2.2.1.1.3 Horovod 并行模式

提交以 Horovod 实现的并行 AI 训练程序，并指定并行实例数和每个实例的资源规格。如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的计算资源。



Tensorflow Horovod 并行模式作业提交页面

2.2.1.2 命令行方式提交

2.2.1.2.1 单机模式

通过终端命令行方式提交单机 Tensorflow 训练程序，提交脚本如下所示：

```
#!/bin/sh


#BSUB -J tensorflow_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
```

```
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/tensorflow:2.13  --cmd='python
mnist_tf2.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < tensorflow_standalone_cpu.sub
```

2.2.1.2.2 PS 并行模式

通过终端命令行方式提交以 Parameter Server 方式实现的 Tensorflow 训练代码，支持跨节点，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J tensorflow_ps_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf ps_cpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2
```

```
 ${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type ps --image 192.168.0.153:5000/jhinno/tensorflow:2.13
--cmd='python tensorflow2_ps.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_ps_cpu.sub

```
1*[host:all slots:2 tag:"server"]
1*[host:all slots:2 tag:"worker"]
```

ps_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < tensorflow_ps.sub
```

2.2.1.2.3 Horovod 并行模式

通过终端命令行方式提交以 Horovod 实现的并行 Tensorflow 训练程序，提交脚本如下所示：

```
#!/bin/sh


#BSUB -J tensorflow_horovod_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf horovod_cpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type horovod --image 192.168.0.153:5000/jhinno/tensorflow:2.13
--cmd='python mnist_tf2_horovod.py'
--workdir=${HOME}/example/tensorflow
```

tensorflow_horovod_cpu.sub

```
2*[host:all slots:2]
```

horovod_cpu.mf

准备好脚本后通过以下命令提交：

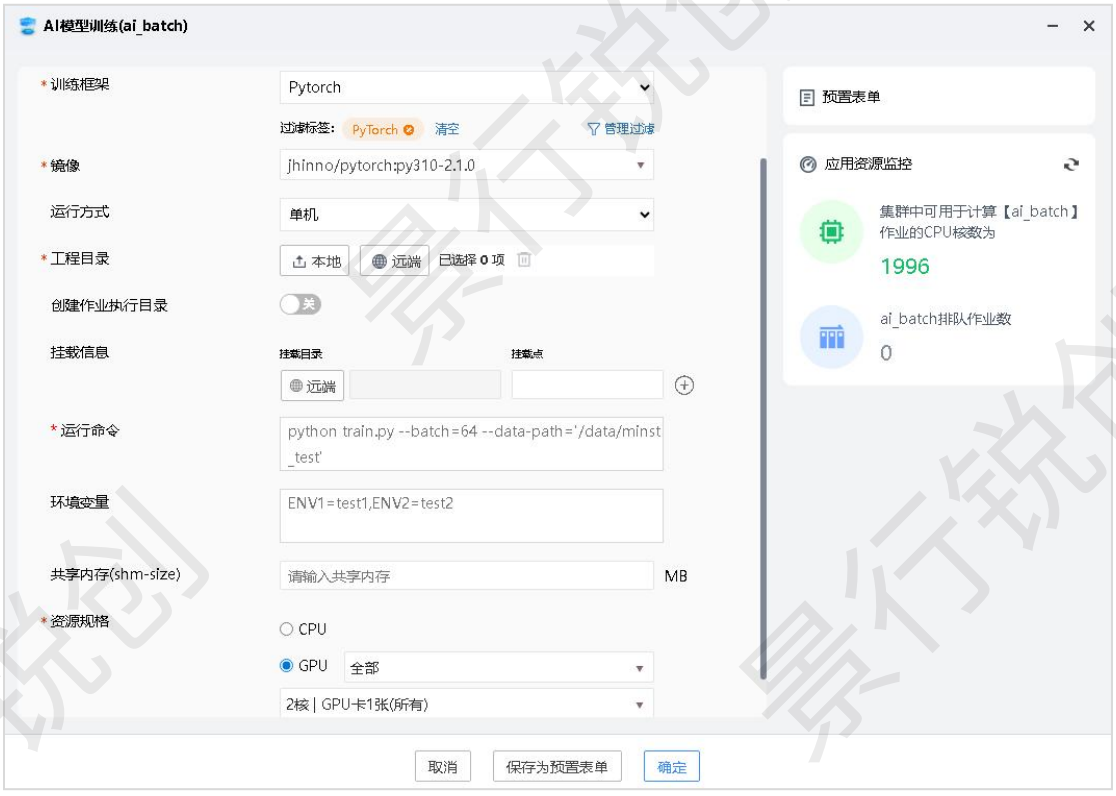
```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < tensorflow_horovod_cpu.sub
```

2.2.2 Pytorch

2.2.2.1 可视化方式提交

2.2.2.1.1 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“Pytorch”，如下图所示：



Pytorch 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：pytorch。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Pytorch”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“pytorch”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

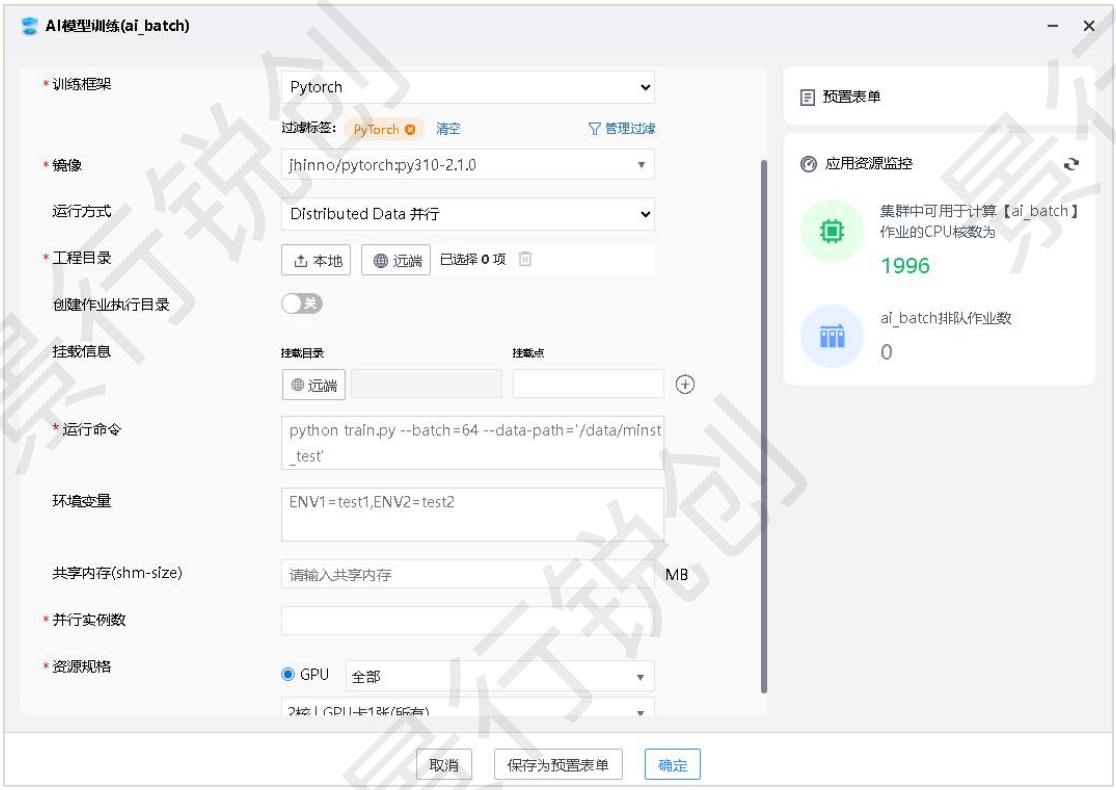
预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

2.2.2.1.2 Distributed Data 并行(DDP)模式

提交以 Distributed Data Parallel(DDP)实现的分布式数据并行 AI 训练程序，并指定并行实例数和每个实例的资源规格，如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的计算资源。

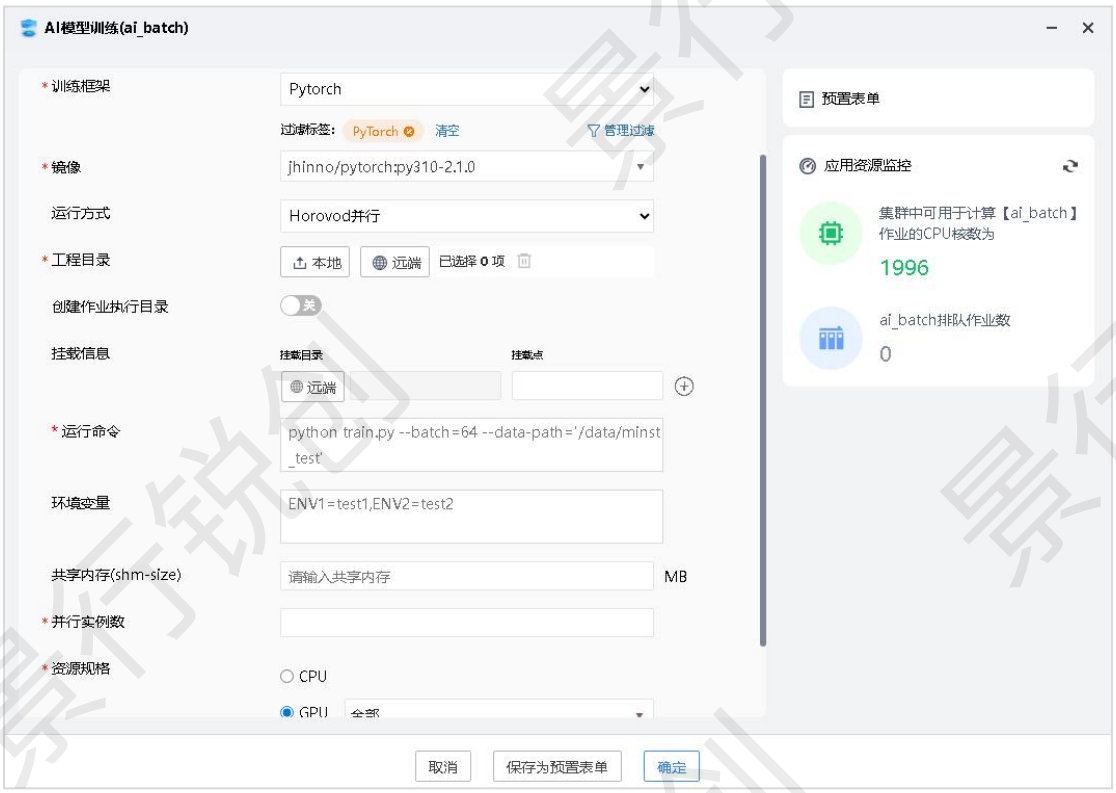


Pytorch DDP 模式作业提交界面

2.2.2.1.3 Horovod 并行模式

提交以 Horovod 实现的并行 AI 训练程序，并指定并行实例数和每个实例的资源规格，如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的计算资源。



Pytorch Horovod 并行模式作业提交界面

2.2.2.2 命令行方式提交

2.2.2.2.1 单机模式

通过终端命令行方式提交 Pytorch 单机训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J pytorch_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image 192.168.0.153:5000/jhinno/pytorch:2.1.0
--cmd='python mnist_torch.py' --workdir=${HOME}/example/pytorch
```

pytorch_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < pytorch_standalone_cpu.sub
```

2.2.2.2.2 Horovod 并行模式

通过终端命令行方式提交以 Horovod 实现的 Pytorch 并行训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J pytorch_horovod_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf horovod_cpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type horovod --image 192.168.0.153:5000/jhinno/pytorch:2.1.0
--cmd='python mnist_torch_horovod.py'
--workdir=${HOME}/example/pytorch
```

pytorch_horovod_cpu.sub

```
2*[host:all slots:2]
```

horovod_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
```

```
jsub < pytorch_horovod_cpu.sub
```

2.2.3 Mxnet

2.2.3.1 可视化方式提交

2.2.3.1.1 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“Mxnet”，如下图所示：

Mxnet 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：mxnet。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Mxnet”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“Mxnet”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

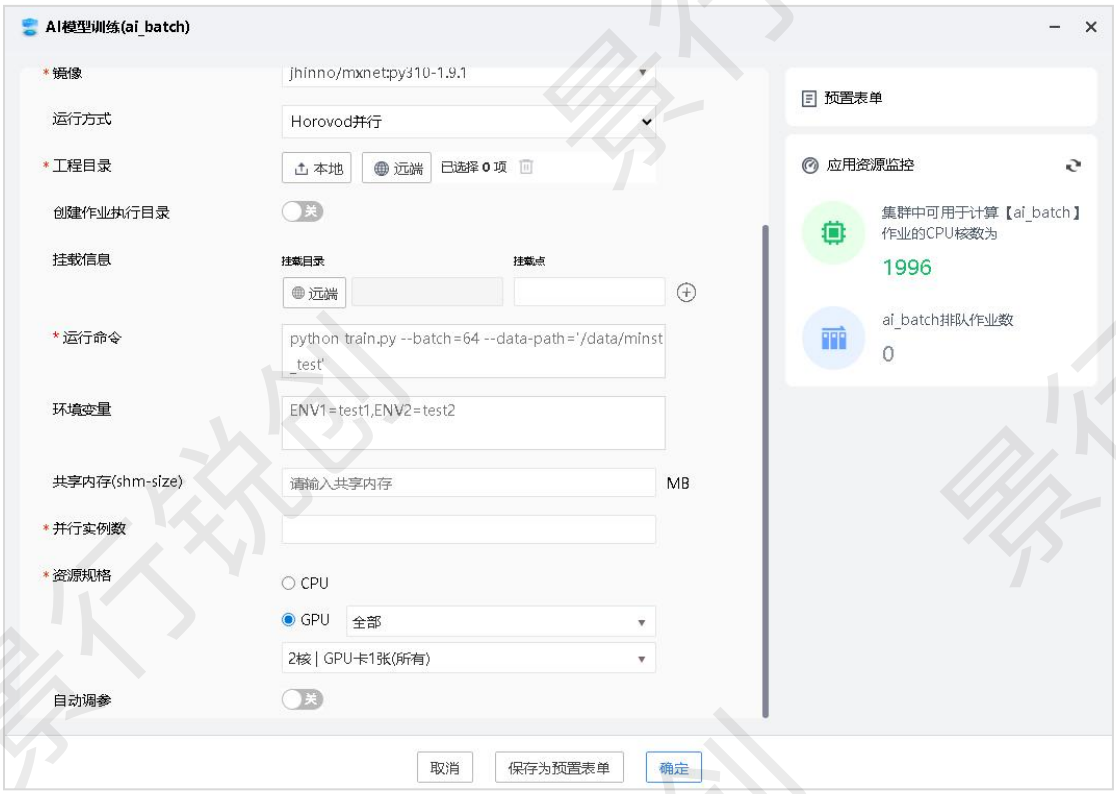
预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

2.2.3.1.2 Horovod 并行模式

提交以 Horovod 实现的并行 AI 训练程序，并指定并行实例数和每个实例的资源规格，如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的资源。



Mxnet Horovod 并行模式作业提交页面

2.2.3.2 命令行方式提交

2.2.3.2.1 单机模式

通过终端命令行方式提交单机 Mxnet 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J mxnet_standalone_gpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_gpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image 192.168.0.153:5000/jhinno/mxnet:1.9.1
--cmd='python mnist_mxnet.py' --workdir=${HOME}/example/mxnet
```

mxnet_standalone_gpu.sub

```
[host:all slots:1 gpu:1]
```

standalone_gpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < mxnet_standalone_gpu.sub
```

2.2.3.2.2 Horovod 并行模式

通过终端命令行方式提交以 Horovod 实现的并行 Mxnet 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J mxnent_horovod_gpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf horovod_gpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type horovod --image 192.168.0.153:5000/jhinno/mxnet:1.9.1
--cmd='python mnist_mxnet_horovod.py' --workdir=${HOME}/example/mxnet
```

mxnet_horovod_gpu.sub

```
2*[host:all slots:1 gpu:1]
```

horovod_gpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
```

```
jsub < mxnet_horovod_gpu.sub
```

2.2.4 PaddlePaddle

PaddlePaddle（百度飞桨）是由百度推出的开源深度学习平台，旨在为开发者提供灵活、高效的工具和库，以构建各种深度学习模型。以下是 PaddlePaddle 的一些主要特点和优势：

全面的深度学习支持：PaddlePaddle 支持多种深度学习任务，包括图像识别、自然语言处理、语音识别等。它提供了丰富的预训练模型和预处理工具，使开发者能够更轻松地构建自己的模型。

灵活的动态图和静态图：PaddlePaddle 支持动态图和静态图的混合编程。这使得开发者可以选择适合其任务和习惯的编程方式，从而更好地平衡灵活性和性能。

分布式训练：PaddlePaddle 提供了强大的分布式训练能力，支持在多个设备和多个节点上进行模型训练，以应对大规模数据和复杂模型的挑战。

多端部署：PaddlePaddle 支持模型在不同平台上的部署，包括移动端、嵌入式设备和服务器。这使得开发者能够将训练好的模型应用到各种应用场景中。

易用性：PaddlePaddle 注重开发者体验，提供了简洁、清晰的 API 和文档。同时，它还集成了可视化工具，帮助开发者更好地理解和调试他们的模型。

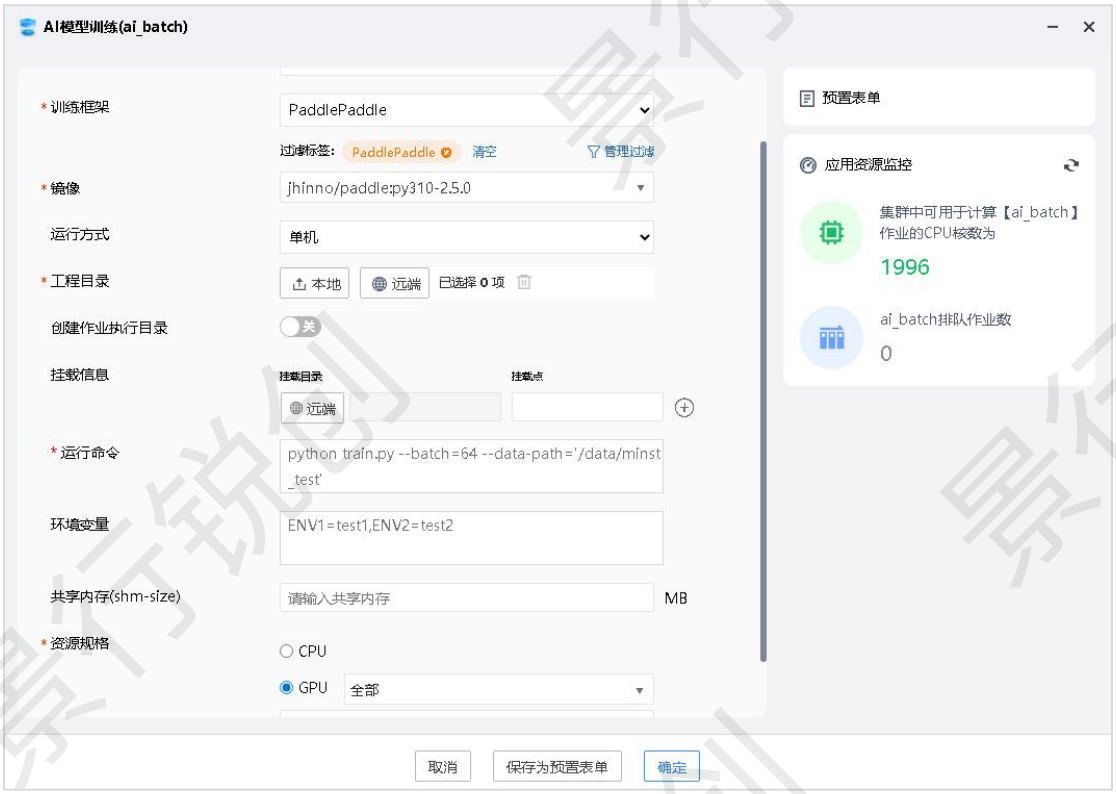
总体而言，PaddlePaddle 是一款强大而全面的深度学习平台，适用于各种规模和领域的深度学习项目。

2.2.4.1 可视化方式提交

2.2.4.1.1 单机模式

2.2.4.1.1.1 单机单卡

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“PaddlePaddle”，如下图所示：



PaddlePaddle 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：PaddlePaddle。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“PaddlePaddle”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“PaddlePaddle”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂

载数据集。

2） **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train_demo.py。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

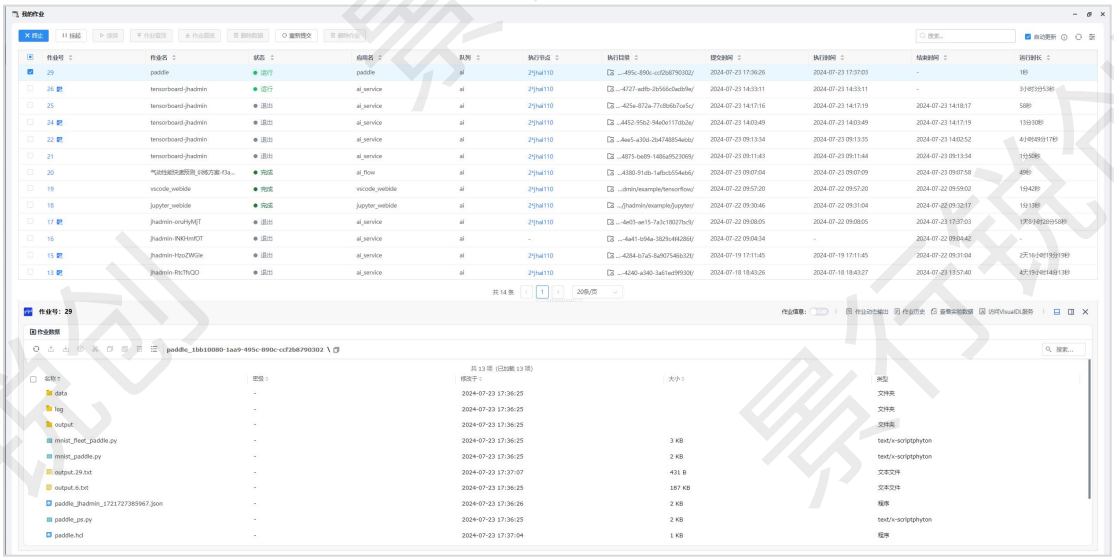
资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

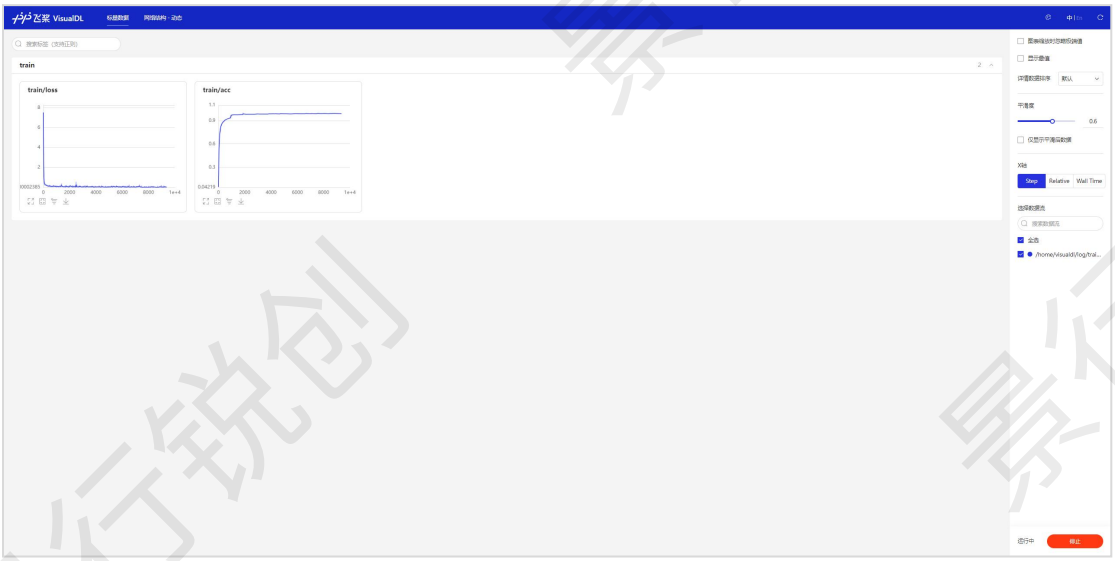
PaddlePaddle 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



作业信息查看

点击作业滑块右上角的“访问 VisualDL 服务”按钮，可以查看训练情况，

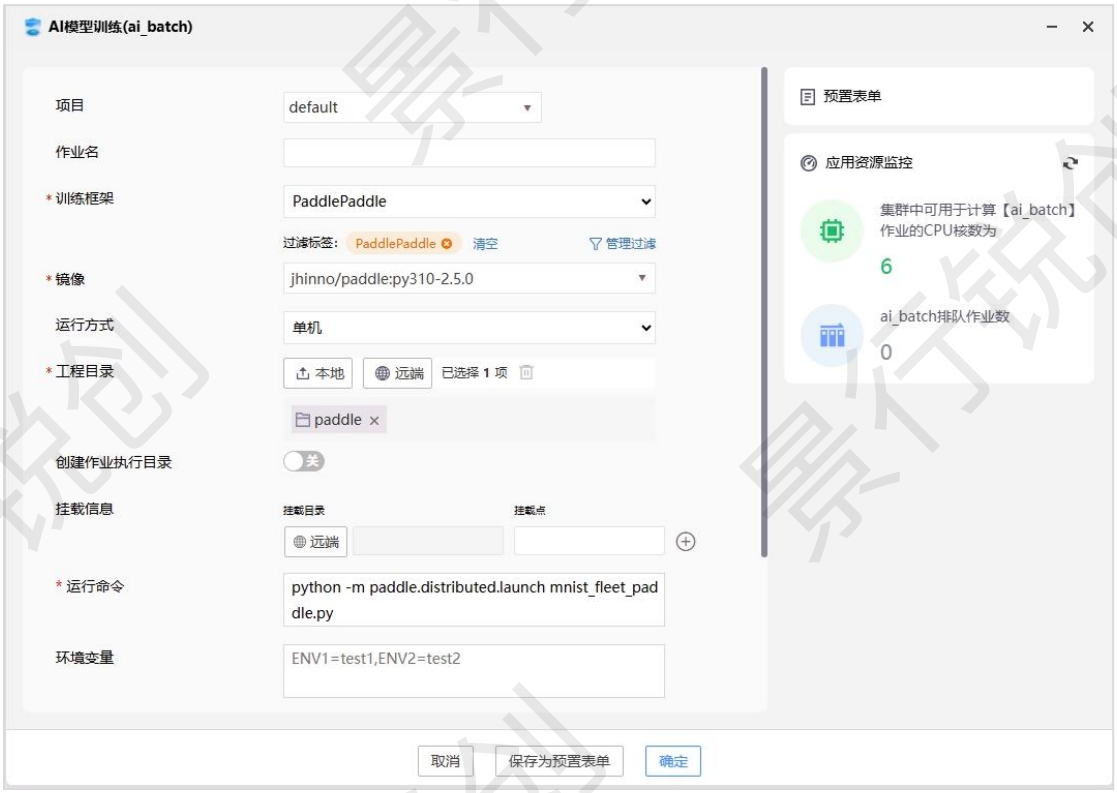
如下图所示：

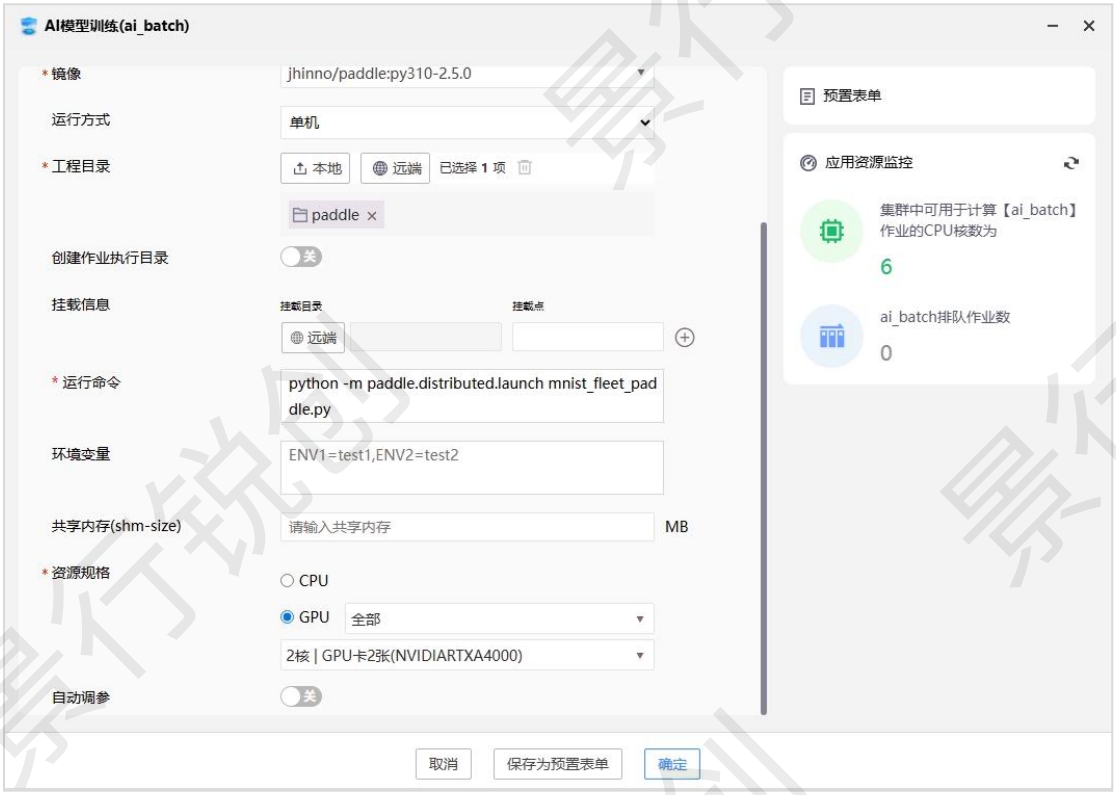


访问 VisudlDL 服务

2.2.4.1.1.2 单机多卡

提交单机多卡 PaddlePaddle 作业，需要在运行命令处额外配置“-m paddle.distributed.launch”参数，例如：python -m paddle.distributed.launch train.py，如下图所示：



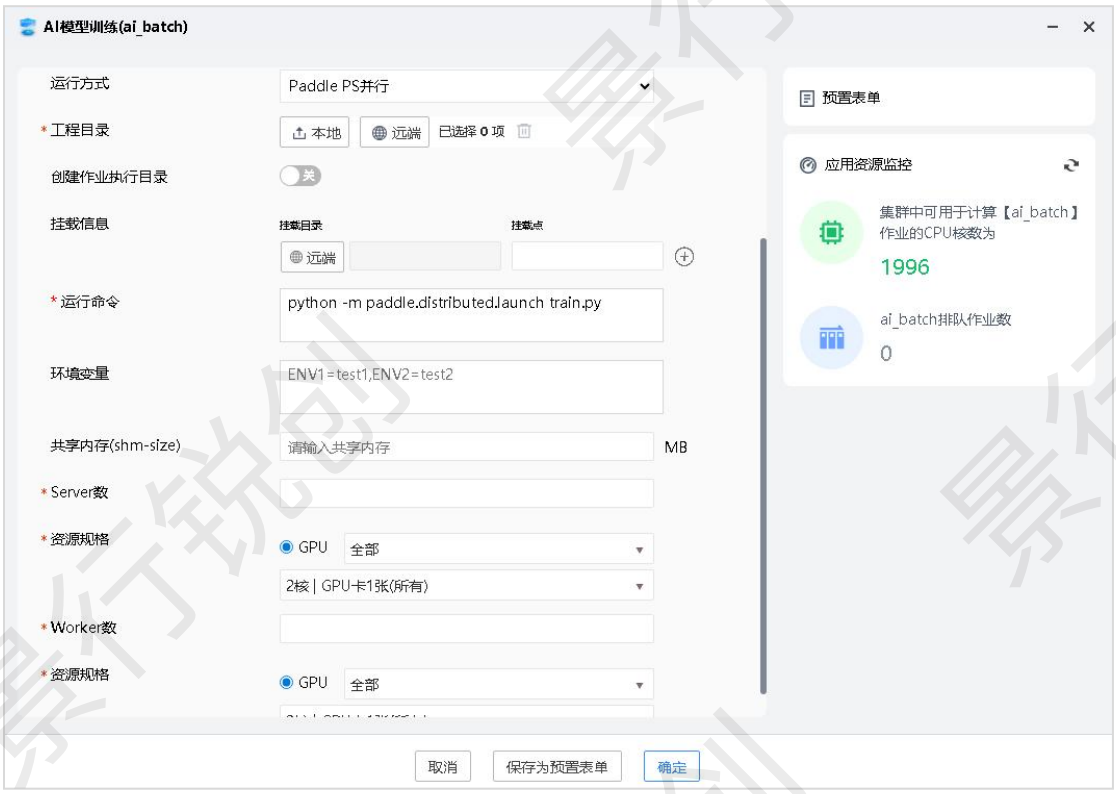


PaddlePaddle 单机多卡模式作业提交界面

2.2.4.1.2 Paddle PS 并行模式

提交以 Paddle 框架 Parameter Server 方式实现的训练代码，支持跨节点，需填写并行相关参数，如下所示：

- **Server 数：**指定参数服务的数量。
- **Worker 数：**指定训练服务数量。
- **资源规格：**分别指定每个 Server 和 Worker 使用的资源。



Paddle PS 并行模式作业提交页面

2.2.4.2 命令行方式提交

2.2.4.2.1 单机模式

通过终端命令行方式提交单机 PaddlePaddle 训练程序，提交脚本如下所示：

```
#!/bin/sh
```

```
#BSUB -J paddle_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image 192.168.0.153:5000/jhinno/paddle:2.5.0
--cmd='python mnist_paddle.py' --workdir=${HOME}/example/paddle
```

paddle_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < paddle_standalone_cpu.sub
```

2.2.4.2.2 Paddle PS 并行模式

通过终端命令行方式提交 Paddle PS 模式 PaddlePaddle 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J paddle_ps_gpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf ps_gpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type ps --image 192.168.0.153:5000/jhinno/paddle:2.5.0
--cmd='python paddle_ps.py' --workdir=${HOME}/example/paddle
```

paddle_ps_gpu.sub

```
1*[host:all slots:2 tag:"server"]
1*[host:all slots:2 gpu:1 tag:"worker"]
```

ps_gpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < paddle_ps_gpu.sub
```

2.2.5 MindSpore

MindSpore 是华为推出的开源深度学习框架，旨在提供灵活、高性能、易用的深度学习开发平台。它支持端到云的全场景 AI 应用开发，并具有以下特点：

自动并行：MindSpore 能够自动分析网络结构并生成相应的并行计算图，提高训练性能。

全场景：适用于各种场景，包括设备、边缘和云端，支持多种硬件架构。

算子融合：通过算子融合技术，降低内存占用，提高计算效率。

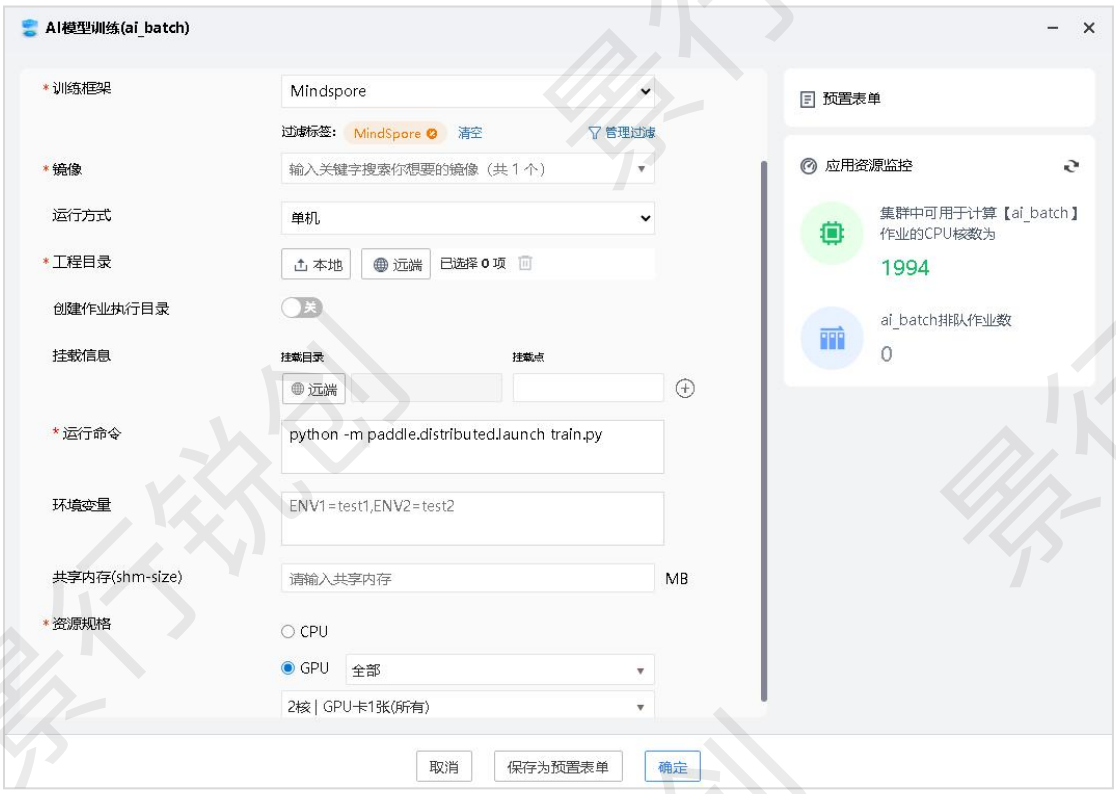
模型保护：提供了模型保护技术，有助于保护模型的安全性和隐私性。

易用性：设计简单直观，提供了友好的 API 和丰富的文档，降低开发者的学习成本。

2.2.5.1 可视化方式提交

2.2.5.1.1 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“MindSpore”，如下图所示：



MindSpore 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：mindspore。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Mindspore”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“MindSpore”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂

载数据集。

2） **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --data-path=/data --batch=54。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

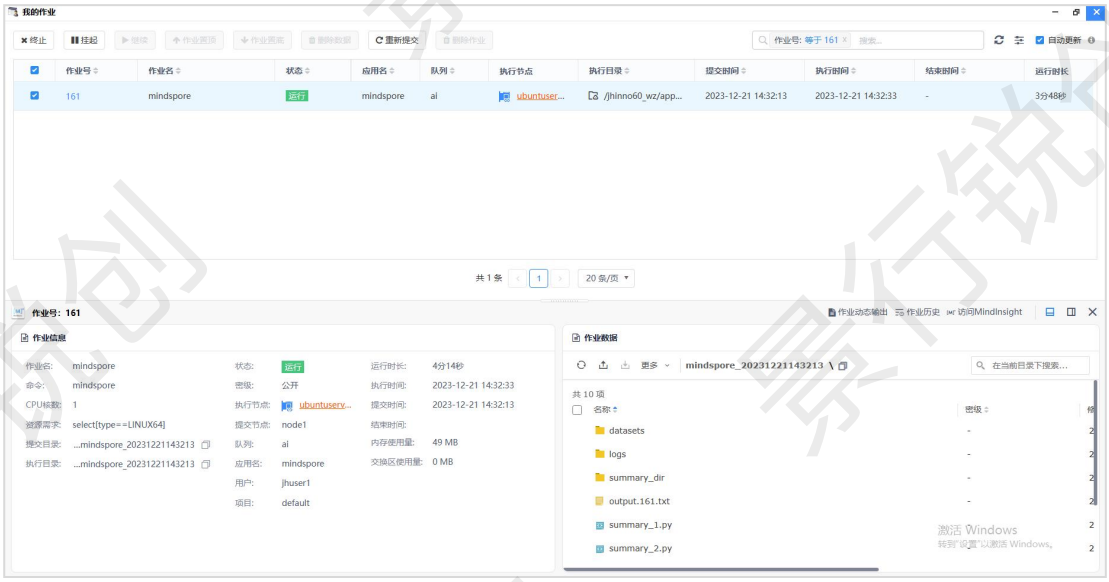
资源规格：必选项，选择运行程序所需要的资源配置。只能选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

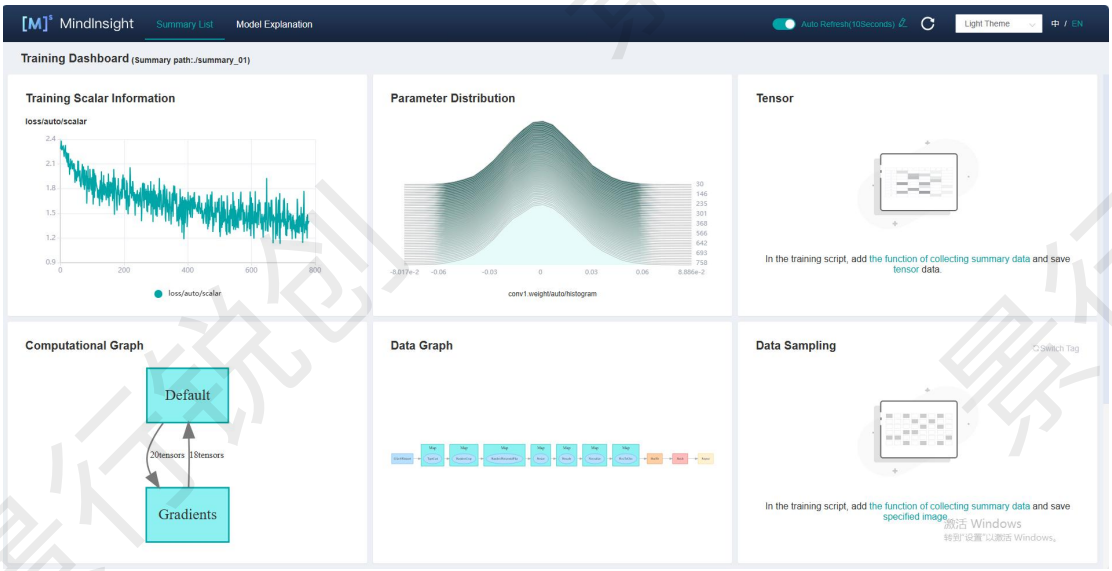
应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

MindSpore 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



作业信息查看

点击作业滑块右上角的“访问 MindInsight”按钮，可以查看训练情况，如下图所示：

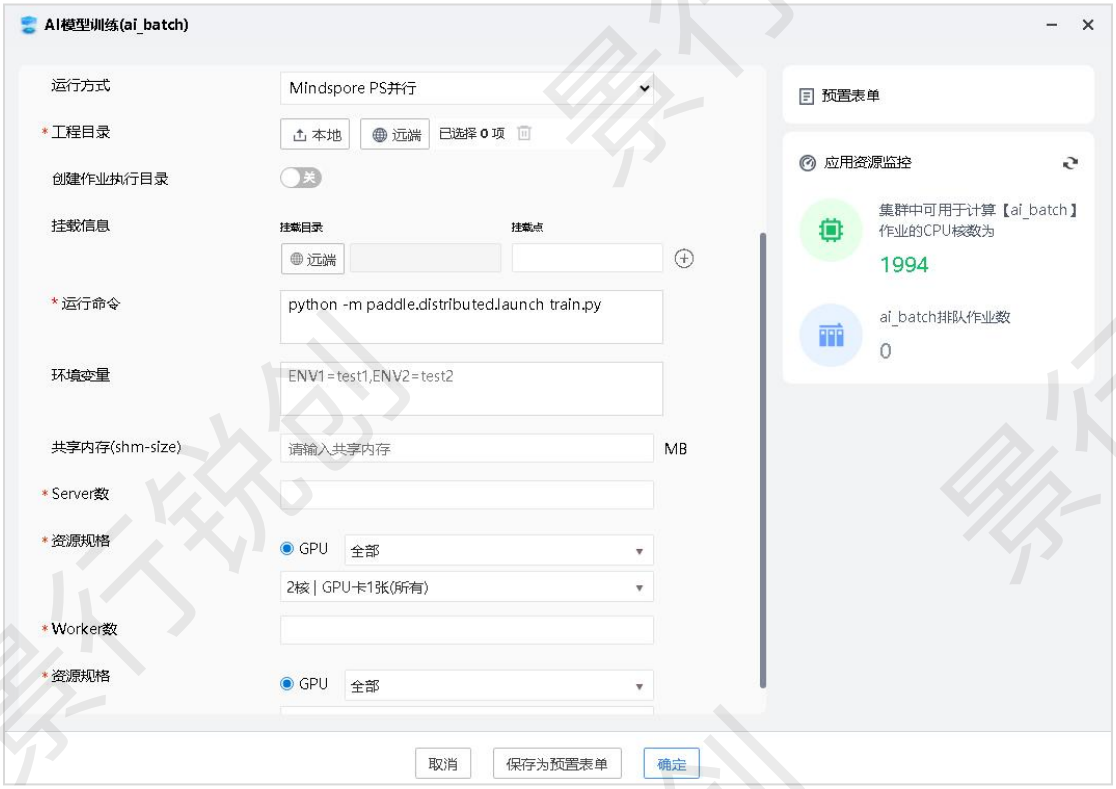


访问 MindInsight

2.2.5.1.2 MindSpore PS 并行模式

提交以 MindSpore 框架 Parameter Server 方式实现的训练代码，支持跨节点，需填写并行相关参数，如下所示：

- **Server 数：**指定参数服务的数量。
- **Worker 数：**指定训练服务的数量。
- **资源规格：**分别指定每个 Server 和 Worker 使用的资源。



MindSpore PS 并行模式作业提交页面

2.2.5.1.3 Open MPI 并行模式

提交以 Open MPI 实现的并行 AI 训练程序， 并指定并行实例数和每个实例的资源规格。如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例使用的计算资源。

MindSpore Open MPI 并行模式作业提交页面

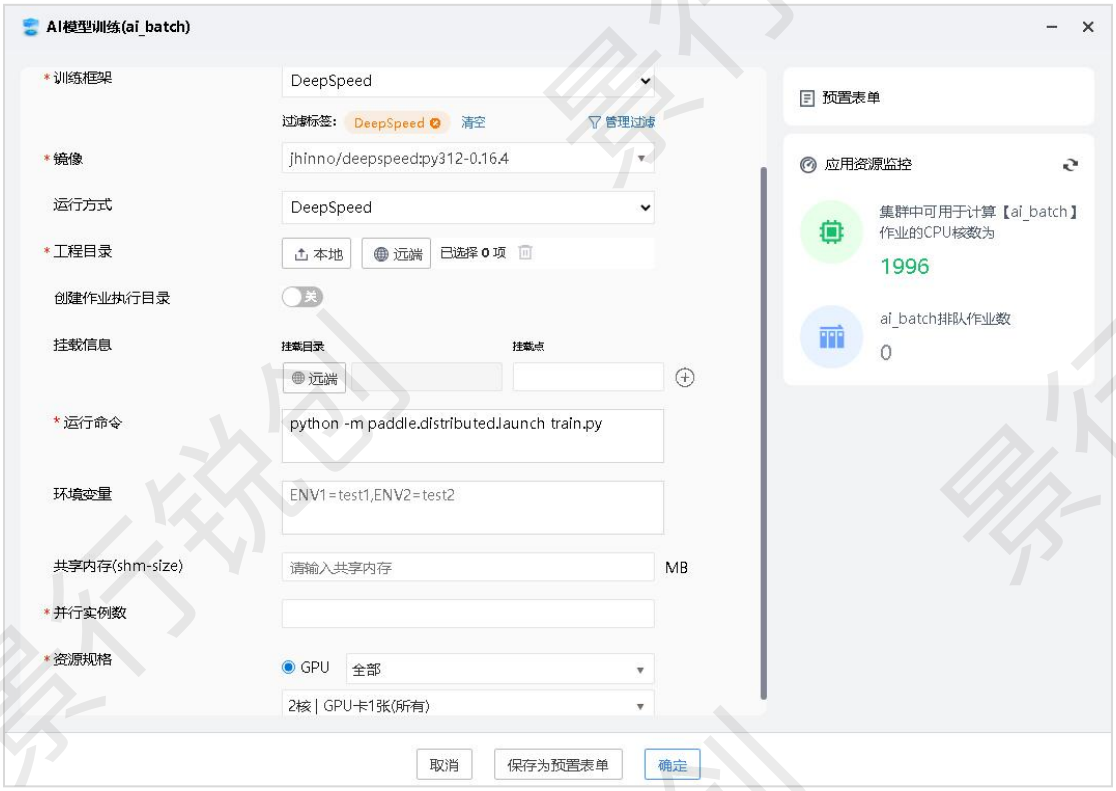
2.2.6 DeepSpeed

DeepSpeed 是由微软开发的开源深度学习优化库，旨在提高大规模模型训练的效率和可扩展性。通过创新的并行化策略、内存优化技术（如 ZeRO）及混合精度训练，DeepSpeed 显著提升了训练速度并降低了资源需求。

2.2.6.1 可视化方式提交

2.2.6.1.1 DeepSpeed 并行模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“DeepSpeed”，如下图所示：



DeepSpeed 并行模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：deepspeed。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“DeepSpeed”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“DeepSpeed”的镜像。

运行方式：默认选择 DeepSpeed 并行运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂

载数据集。

2） **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --data-path=/data --batch=54。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

并行实例数：提交以 DeepSpeed 实现的并行 AI 训练程序，指定并行训练的实例数量。

资源规格：必选项，指定每个并行实例所使用的计算资源。只能选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

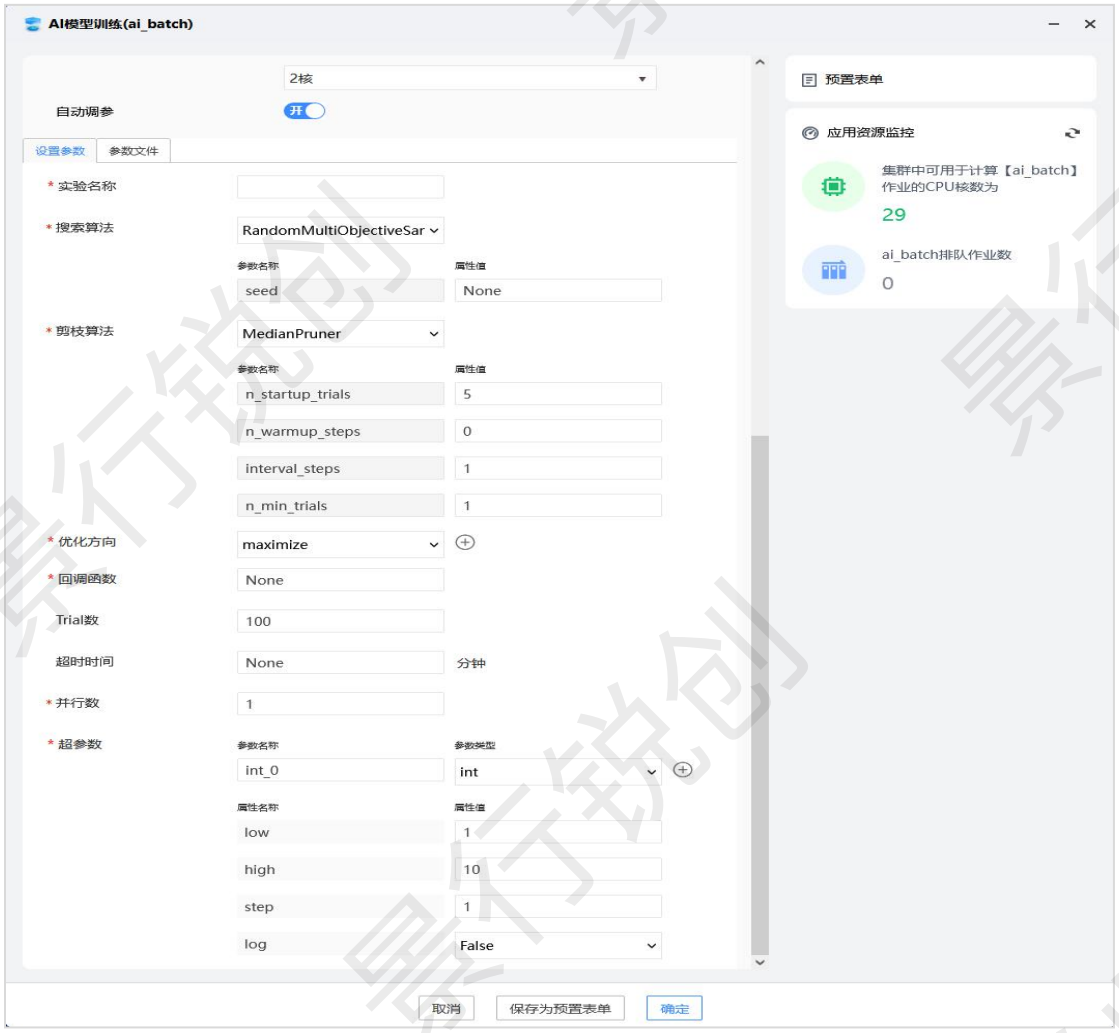
2.2.7 自动调参

景行人工智能软件平台在作业提交模块引入了智能的自动调参功能，旨在解决训练模型过程中繁琐的参数设置和结果统计的问题。传统方式下，每次实验都需要手动设定参数，对于性能较差的模型，更需要手动中止，这不仅工作量大，还容易导致计算资源的浪费。自动调参的引入通过设置试验次数和实验时间，能够及时中止性能较差的模型，实现了对计算资源的有效利用。

此外，自动调参还具备自动保存实验参数和结果的功能。在每次实验结束后，系统会自动记录并保存实验的关键参数和结果，最终提供最优参数组合，从而显著减少了训练模型所需的时间。这一智能调优功能为用户提供了更高效、便捷的超参数搜索体验，使得模型优化过程更加智能和可控。

2.2.7.1 设置参数

打开自动调参开关，展示自动调参配置，如下所示：



设置参数

在“设置参数”标签页中，可查看设置自动调参的所有参数，如下所示：

实验名称：必填项，作业训练过程中创建实验的名称。

搜索算法：人工智能平台中集成的搜索算法。搜索算法主要有“RandomMultiObjectiveSampler”，“NSGAIIMultiObjectiveSampler”，“MOTPEMultiObjectiveSampler”，“TPESampler”，“QMCSampler”，“MOTPESampler”，“RandomSampler”，“GridSampler”，“CmaEsSampler”和“BruteForceSampler”，搜索算法的参数解释见附件八搜索算法。

剪枝算法：选择是否对实验中的试验进行剪枝，当选择“NopPruner”时，表示不剪枝；其他剪枝算法“PercentilePruner”，“HyperbandPruner”，“M

edianPruner”，“PatientPruner”，“ThresholdPruner”，“SuccessiveHalvingPruner”的剪枝的过程中具体剪枝参数，见附件八剪枝算法。

优化方向：优化方向可选参数有“minimize”和 “maximize”。当实验的优化目标有多个时，需要设置多个优化方向。

回调函数：自定义回调函数名称，回调函数的个数应该与优化方向个数相同。此处设置的回调函数名称，可以作为实验管理中查看指标的索引。

Trial 个数：设置实验所有的 Trial 个数，默认为 100 个。当参数的组合数量小于 100 时，运行 Trial 为实际组合数。

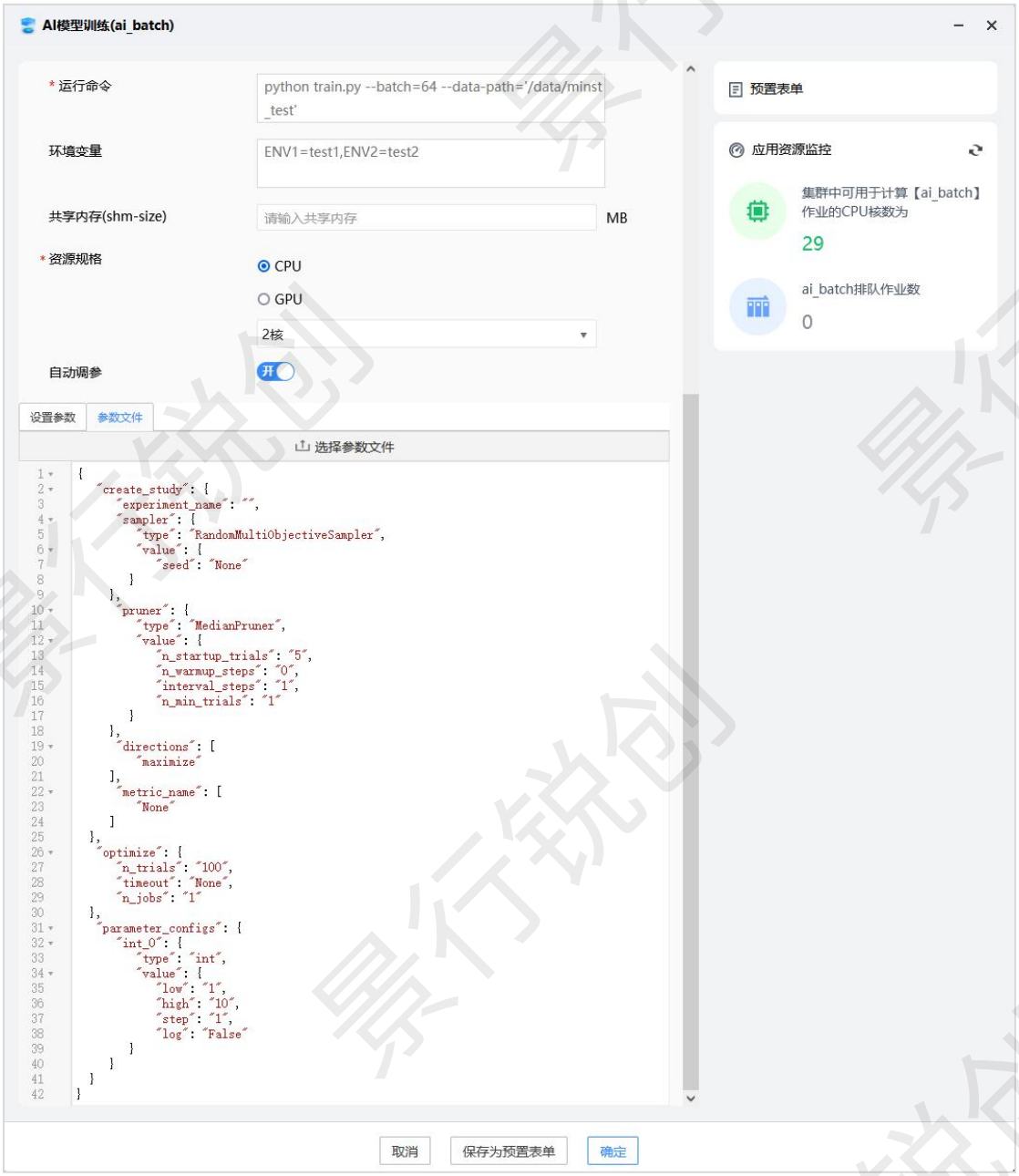
超时时间：本地实验运行的时长，单位为分钟，默认不限制运行时间。

并行数：同时启动的线程数，默认为 1 个。当并行数大于当前环境的核数时，实际并行数为当前的核数。

超参数：设置调参参数和参数搜索范围，可添加的参数不限个数。参数名，调参参数的名称；参数类型，包括 int，float，uniform，loguniform，discrete_uniform 和 categorical。

2.2.7.2 参数文件

点击“参数文件”标签页按钮，切换至参数文件界面。点击“选择参数文件”按钮，可以上传参数文件，如下图所示：



上传文件的格式，如下所示：

```
{
  "create_study": {
    "experiment_name": "",
    "sampler": {
      "type": "RandomMultiObjectiveSampler",
      "value": {
        "seed": "None"
```

```
      }
    },
    "pruner": {
      "type": "MedianPruner",
      "value": {
        "n_startup_trials": "5",
        "n_warmup_steps": "0",
        "interval_steps": "1",
        "n_min_trials": "1"
      }
    },
    "directions": [
      "maximize"
    ],
    "metric_name": [
      "None"
    ]
  },
  "optimize": {
    "n_trials": "100",
    "timeout": "None",
    "n_jobs": "1"
  },
  "parameter_configs": {
    "int_0": {
      "type": "int",
      "value": {
        "low": "1",
        "high": "10",
```

```
        "step": "1",
        "log": "False"
      }
    }
  }
}
```

自动调参的参数：

experiment_name：对应设置参数中的实验名称；

sampler：对应设置参数中的搜索算法；

pruner：对应设置参数中的剪枝算法；

directions：对应设置参数中的优化方向；

metric_name：对应设置参数中的回调函数；

n_trials：对应设置参数中的 trial 个数；

timeout：对应设置参数中的超时时间；

n_jobs：对应设置参数中的并行数；

parameter_configs：对应设置参数中的超参数。

2.2.7.3 脚本 API 设置

为了实现自动调参功能，除了设置上述参数或者文件之外，需要在运行脚本中添加相关内容：

（1）添加“parse_arg”模型，作为参数入口，如下所示：

```
def parse_arg(args=None):
    """parse arguments"""
    parser = argparse.ArgumentParser(description='tensorflow MNIST Example')
    parser.add_argument('--filters', type=int, default=32,
                        help='number of the filters')
    parser.add_argument('--kernel_size', type=int, default=2,
```

```
                        help='number of the kernel size')
    parser.add_argument('--strides', type=int, default=1,
                        help='strides size')
    parser.add_argument('--activation', type=str, default='softmax',
                        help='activation function')
    parser.add_argument('--learning_rate', type=float, default=0.01,
                        help='learning rate (default: 0.01)')
    parser.add_argument('--trial', type=object,
default=optuna.trial._trial.Trial,
                        help='trial')
    args = parser.parse_args(args)
    return args
```

参数名称应该与超参数中设置的参数保持一致，另外需要额外添加以下内容。

```
 Import optuna
 parser.add_argument('--trial', type=object,
default=optuna.trial._trial.Trial,
                        help='trial')
```

（2）如果设置自动剪枝算法，需要添加相应框架的回调，如下所示：

```
 from libaiflow.intergration.callback import TFKerasPruningCallback
    callbacks = [
        tf.keras.callbacks.TensorBoard(log_dir=logdir),

tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_prefix,verbose
= 0,save_best_only=True),
        tf.keras.callbacks.EarlyStopping(patience=3),
        TFKerasPruningCallback(args.trial, 'val_accuracy'), #自动调
参剪枝回调
    ]
```

每个框架具有不同的剪枝回调，具体说明见附件八回调函数。

（3）脚本返回值的个数及目标方向和优化方向保持一致，此处的返回值也就是回调函数显示的内容。整个脚本如下所示：

```
import argparse
import os
import tensorflow as tf
from libaiflow.intergration.callback import TFKerasPruningCallback
import optuna

def parse_arg(args=None):
    parser = argparse.ArgumentParser(description='tensorflow MNIST Example')
    parser.add_argument('--filters', type=int, default=32,
                        help='number of the filters')
    parser.add_argument('--kernel_size', type=int, default=2,
                        help='number of the kernel size')
    parser.add_argument('--strides', type=int, default=1,
                        help='strides size')
    parser.add_argument('--activation', type=str, default='softmax',
                        help='activation function')
    parser.add_argument('--learning_rate', type=float, default=0.01,
                        help='learning rate (default: 0.01)')
parser.add_argument('--trial', type=object,
default=optuna.trial._trial.Trial,help='trial')
    args = parser.parse_args(args)
    return args

def load_data():
  path = "./data/mnist.npz"
```

```
    import numpy as np
    with np.load(path) as f:
      x_train, y_train = f['x_train'], f['y_train']
      x_test, y_test = f['x_test'], f['y_test']
      return (x_train, y_train), (x_test, y_test)

  def main(args):
      from tensorflow.keras.backend import clear_session
      from tensorflow.keras.datasets import mnist
      from tensorflow.keras.layers import Conv2D, Dense, Flatten
      from tensorflow.keras.models import Sequential
      from tensorflow.keras.optimizers import RMSprop
      clear_session()
      N_TRAIN_EXAMPLES = 3000
      N_VALID_EXAMPLES = 1000
      BATCHSIZE = 128
      CLASSES = 10
      EPOCHS = 10

      (x_train, y_train), (x_valid, y_valid) = load_data()
      img_x, img_y = x_train.shape[1], x_train.shape[2]
      x_train = x_train.reshape(-1, img_x, img_y,
1)[:N_TRAIN_EXAMPLES].astype("float32") / 255
      x_valid = x_valid.reshape(-1, img_x, img_y,
1)[:N_VALID_EXAMPLES].astype("float32") / 255
      y_train = y_train[:N_TRAIN_EXAMPLES]
      y_valid = y_valid[:N_VALID_EXAMPLES]
      input_shape = (img_x, img_y, 1)
```

```
    model = Sequential()
    model.add(
        Conv2D(
            filters=args.filters,kernel_size=args.kernel_size,
            strides=args.strides,activation=args.activation,
            input_shape=input_shape,
        )
    )
    model.add(Flatten())
    model.add(Dense(CLASSES, activation="softmax"))
    model.compile(
        loss="sparse_categorical_crossentropy",
        optimizer=RMSprop(learning_rate=args.learning_rate),
        metrics=["accuracy"],
    )
    checkpoint_dir = 'model'
 if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)
    logdir = os.path.join('logs', str(args.trial.number))
    checkpoint_prefix = os.path.join(checkpoint_dir,
str(args.trial.number))
    callbacks = [
        tf.keras.callbacks.TensorBoard(log_dir=logdir),

tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_prefix,verbose
= 0,save_best_only=True),
        tf.keras.callbacks.EarlyStopping(patience=3),
        TFKerasPruningCallback(args.trial, 'val_accuracy'), #自动调
参剪枝回调
```

```
    ]
    model.fit(
        x_train,
        y_train,
        validation_data=(x_valid, y_valid),
        shuffle=True,
        batch_size=BATCHSIZE,
        epochs=EPOCHS,
        verbose=False,
        callbacks=callbacks,
    )
    score = model.evaluate(x_valid, y_valid, verbose=0)
    return score[1]


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

2.3 强化学习

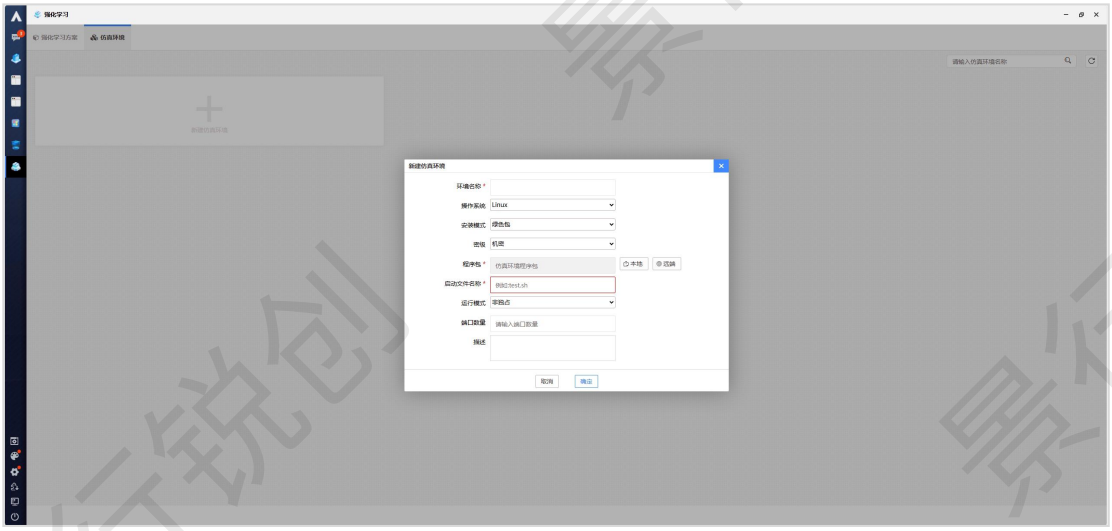
强化学习（Reinforcement Learning，RL）正逐渐成为推动技术进步的关键力量。它通过模拟试错学习的过程，使智能体能够在复杂环境中自主学习并做出最优决策。为了满足日益增长的市场需求，现将 Ray RLlib 的强化学习功能集成平台，旨在简化 RL 模型的开发、训练和部署流程。用户登录应用门户后，点击桌面“强化学习”图标，进入强化学习页面。

2.3.1 构建方案设计流程

2.3.1.1 新建仿真环境

在强化学习应用选择仿真环境，点击“新建仿真环境”卡片，弹出新建方案的表单页。

当安装模式选择“绿色包”，表单页如下图所示：



新建绿色包仿真环境

图中每个参数的具体含义如下：

环境名称：用户自定义，但是不能与现有仿真环境名称一致。

安装模式：仿真环境的安装模式，分为绿色包和已安装环境，默认为绿色包。

密级：管理员开启密级功能后，显示此选项，默认为机密，用于数据安全、保密。

程序包：支持 zip、tar、tar.gz、tar.bz2、tar.xz 结尾的压缩包，可本地或者远端上传，必填项。

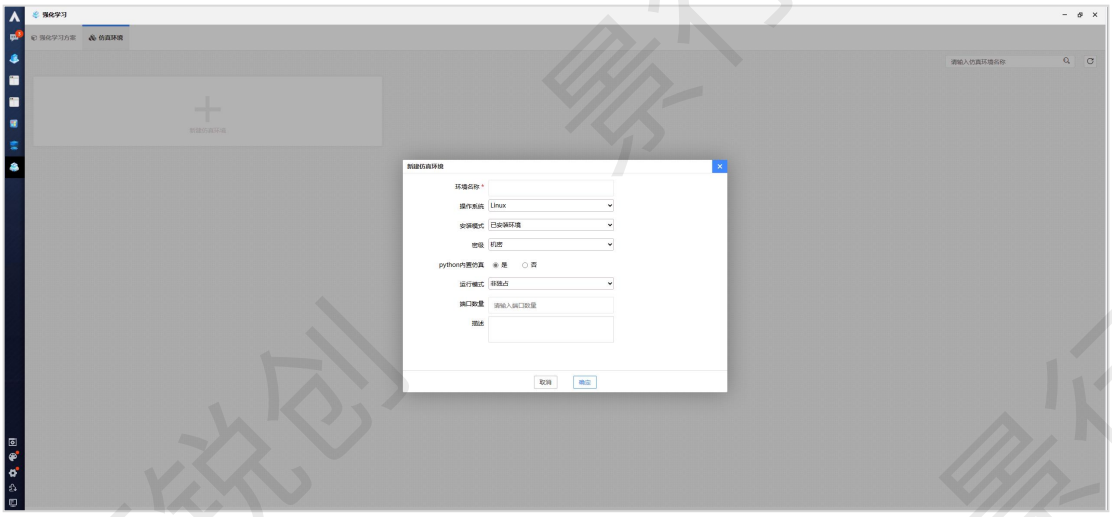
启动文件名称：用户自定义，支持 bat、exe、cmd、sh 结尾的文件全称，必填项。

运行模式：仿真环境的运行模式，默认为非独占。

端口数量：仿真环境需要的端口数量，非必填项，端口数只能输入 1-5 的正整数。

描述：用于描述仿真环境的相关内容，选填。

当安装模式选择“已安装环境”，表单页如下图所示：



新建已安装仿真环境

图中每个参数的具体含义如下：

环境名称：用户自定义，但是不能与现有仿真环境名称一致。

安装模式：仿真环境的安装模式，分为绿色包和已安装环境，默认为绿色包。

密级：管理员开启密级功能后，显示此选项，默认为机密，用于数据安全、保密。

python 内置仿真：是否为 python 内置仿真环境，如果选择是，页面参数不变；如果选择否，增加“启动文件绝对路径”参数。

启动文件绝对路径：用户自定义，输入启动文件的绝对路径，例如：/home/jhadmin/test.sh，必填项。

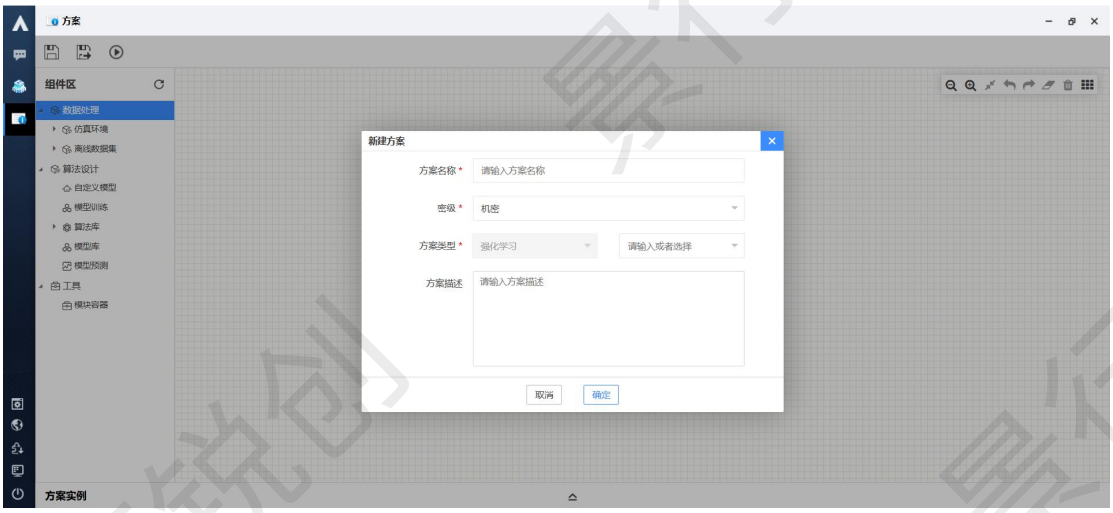
运行模式：仿真环境的运行模式，默认为非独占。

端口数量：仿真环境需要的端口数量，非必填项，端口数只能输入 1-5 的正整数。

描述：用于描述仿真环境的相关内容，选填。

2.3.1.2 新建设计方案

在强化学习应用选择强化学习方案，点击“新建方案”卡片，弹出新建方案的表单页，如下图所示：



新建强化学习方案

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有强化学习方案名称一致。

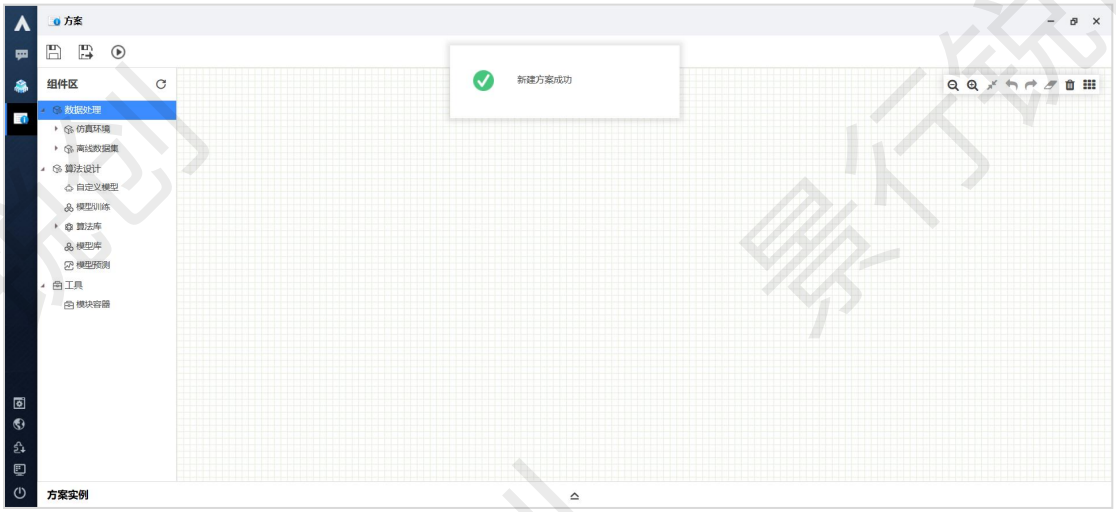
密级：管理员开启密级功能后，显示此选项，默认为机密，用于数据安全、保密。

方案类型：默认为已选的方案类型，也可以自定义方案类型。

方案描述：用于描述方案设计的相关内容，选填。

2.3.1.3 可视化设计

填写完成新建强化学习方案表单后，点击“确定”按钮，进入方案设计页面。

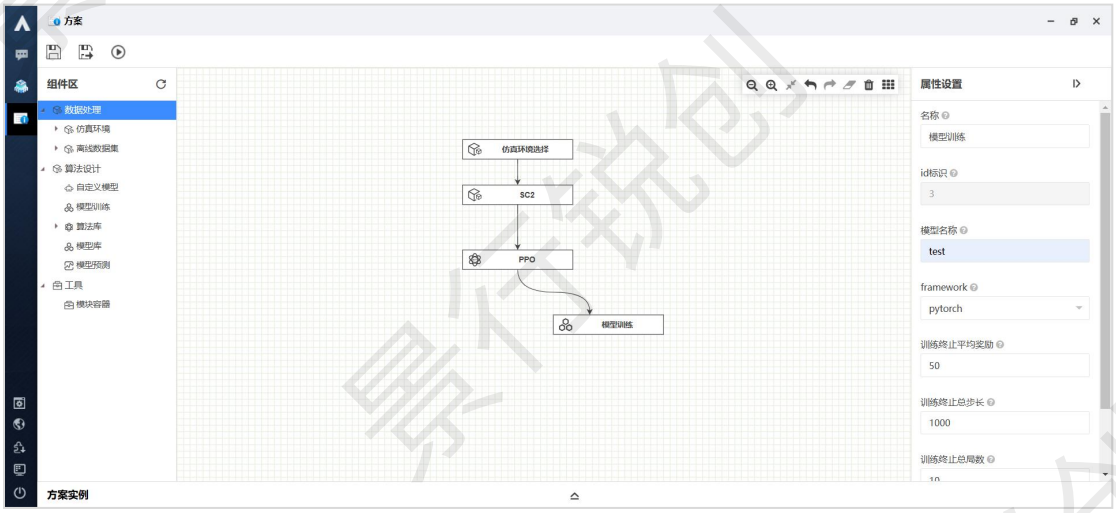


强化学习方案设计页面

“强化学习方案设计”左侧区域为组件区，组件区又根据组件属性分为“数据处理”，“算法设计”和“工具”。每一个组件都是一个模块的封装。

“强化学习方案设计”右侧区域为属性区，组件拖拽到画布区域后，属性设置面板就会从画布右侧滑出，用户可自行配置参数。当属性设置有误或未设置属性时，设计区的组件右侧会显示黄色叹号的图标，当鼠标移至此图标上时会显示出错误信息，必须纠正错误后才能够开始运行此方案。

画布右上角的工具栏可对画布区域和画布中的组件进行快捷操作，功能包含缩小、放大、实际比例、后退、前进、删除和清空。其中“删除”按钮仅会删除画布中选中的任意组件，“清空”则会删除整个画布中的内容，长按鼠标右键可拖动整个画布。

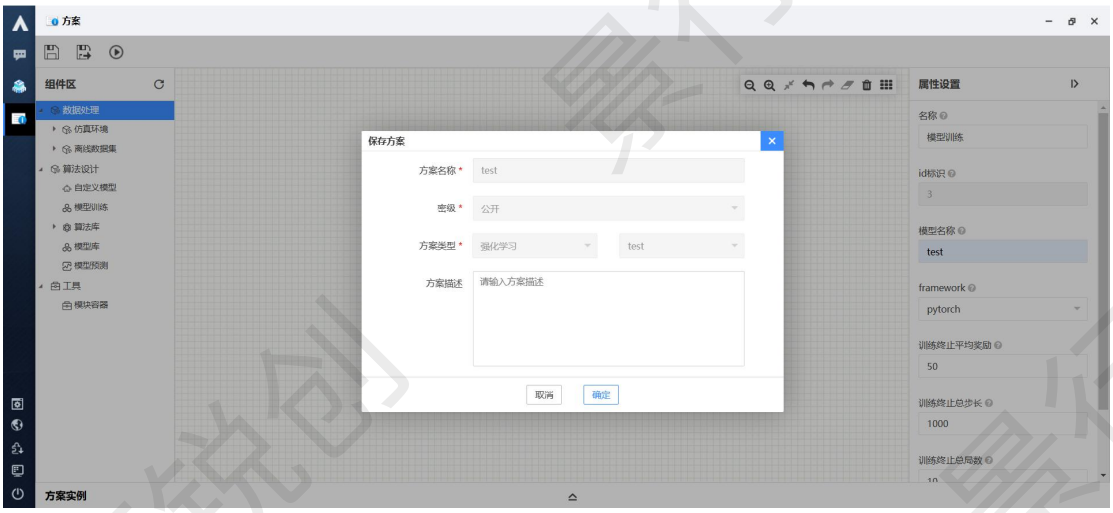


画布工具及组件属性配置

2.3.2 操作

2.3.2.1 方案保存

点击工具栏中的“保存”按钮，即可将方案描述和画布上的强化学习方案设计保存至数据库，当再次打开该方案时，画布上会显示最近一次保存的强化学习方案。“保存方案”页面，如下图所示：



保存方案页面

图中每个参数的具体含义如下：

方案名称：任何状态下均不可修改。

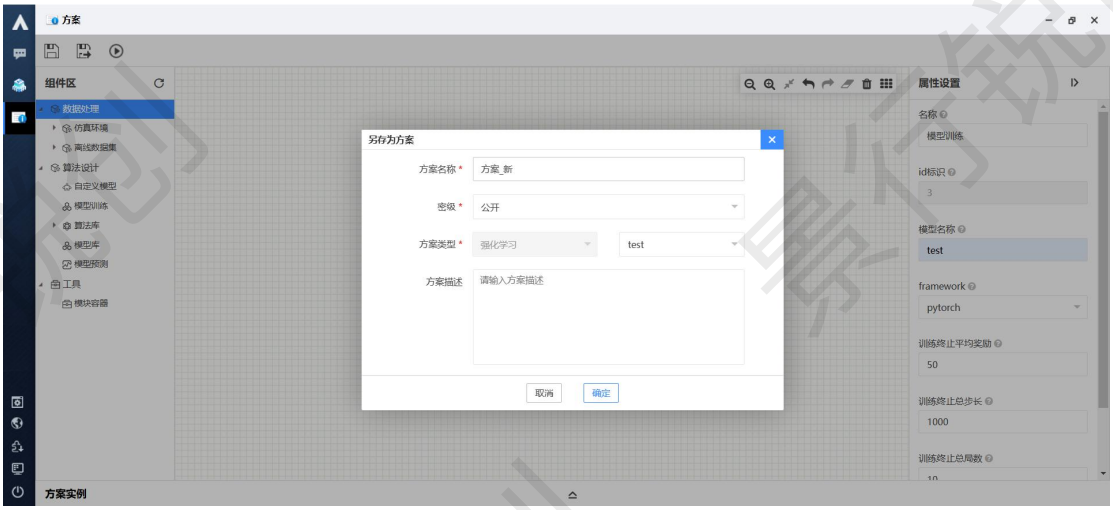
密级：任何状态下均不可修改。

方案类型：任何状态下均不可修改。

方案描述：用于描述方案的相关内容，选填。

2.3.2.2 方案另存为

点击工具栏中的“另存为”按钮，即可将方案另存为一个新的方案，方案可以是已保存的和未保存的。“另存为方案”页面。如下图所示：



另存为方案页面

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有的方案名称一致，必填。

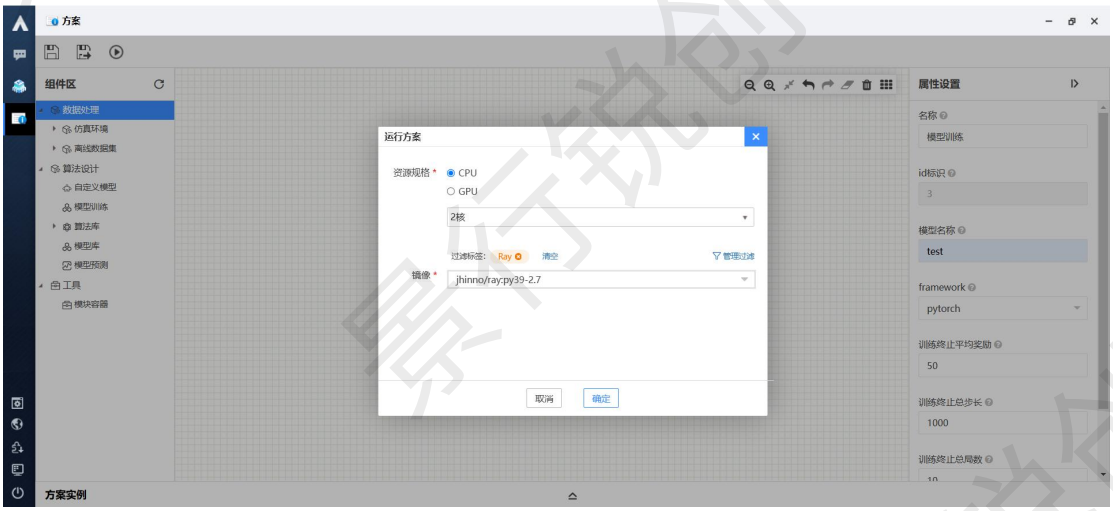
密级：对于另存的方案可根据需要修改密级。

方案类型：默认为当前方案的方案类型，可以选择，也可以自定义方案类型，必填。

方案描述：用于描述方案的相关内容，选填。

2.3.2.3 方案运行

方案设计完成后，点击“运行”按钮，弹出“运行方案”窗口，选择训练方案需要使用的资源，资源选择后，选择方案使用的镜像，点击“确定”按钮，即可开始训练。如下图所示：



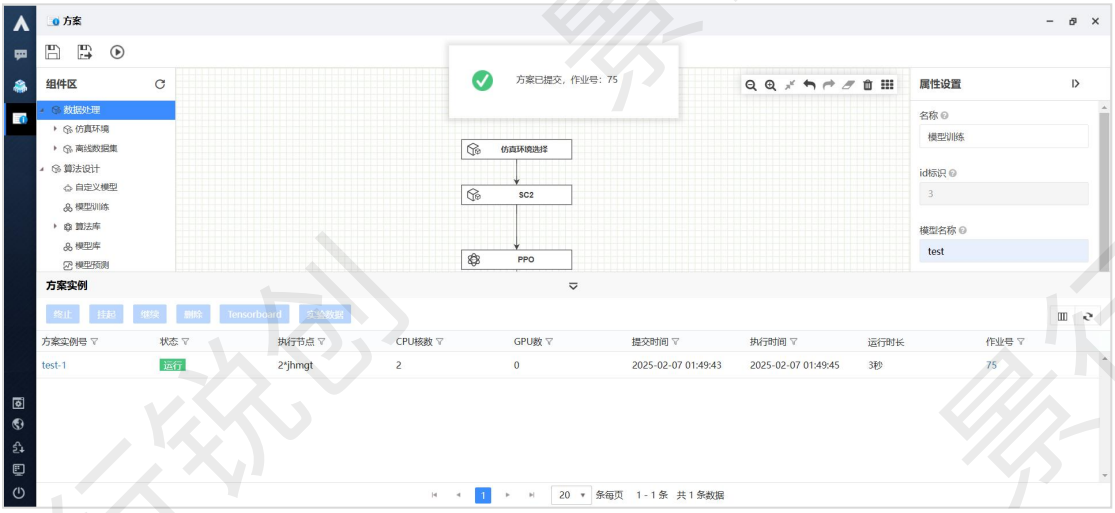
资源配置页面

点击运行按钮，打开选择资源页面，同时进行方案保存。选择资源后，点击确定按钮，即可开始训练方案。

注意：在方案设计页面中，点击“运行”按钮时，错误和问题对应如下：

- 画布中不存在组件时，提示“组件不能为空”。
- 组件的锚点没有连线、必填项属性为空时，提示“请检查异常组件并清除异常”。

开始训练方案后，会自动打开底部方案实例滑块，如下图所示：

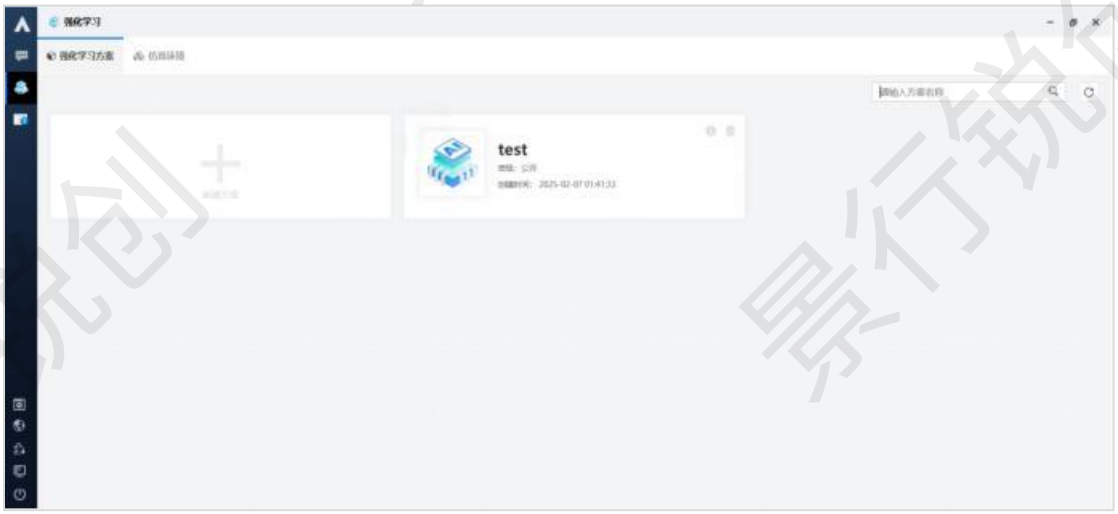


方案实例滑块

底部方案实例滑块除了只能显示该方案的所有实例外，功能操作与方案实例一致，具体操作详情请看“方案实例”章节。

2.3.3 管理

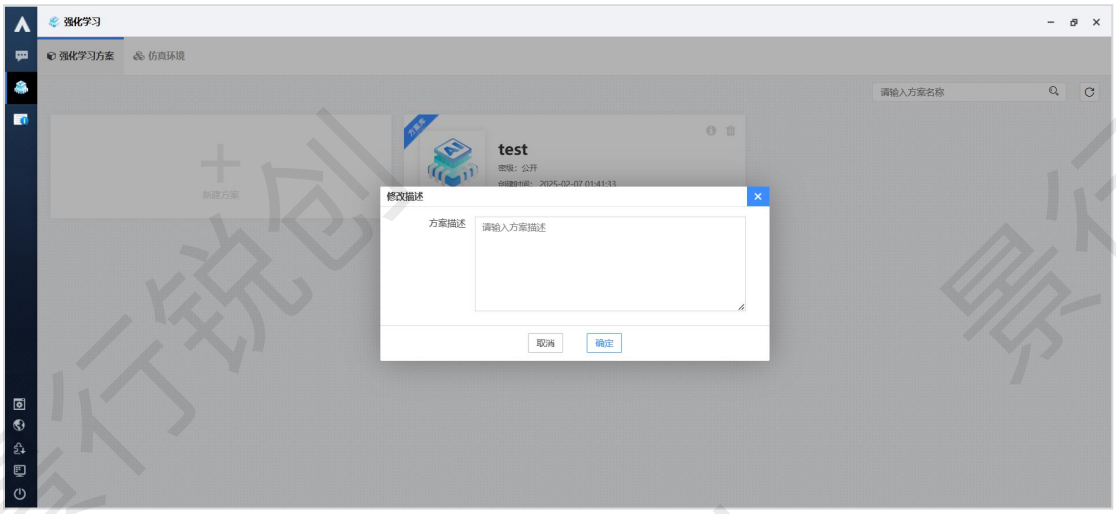
方案新建成功后，在强化学习方案中会自动创建方案卡片，卡片内容包含方案名称、方案 ID、最后更新时间，点击方案卡片的右上角功能图标，支持对方案进行“修改描述”和“删除”操作。



方案设计管理

2.3.3.1 修改描述

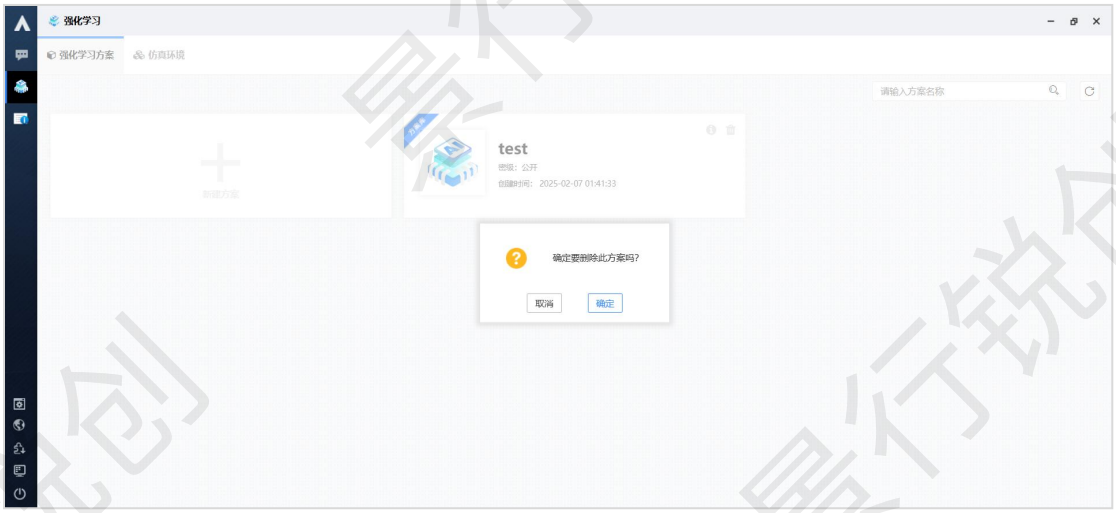
点击“描述”的图标按钮，可在弹出窗口中修改方案的描述信息，如下图所示：



修改方案描述

2.3.3.2 删除

点击“删除”按钮，可以把该方案卡片删除。

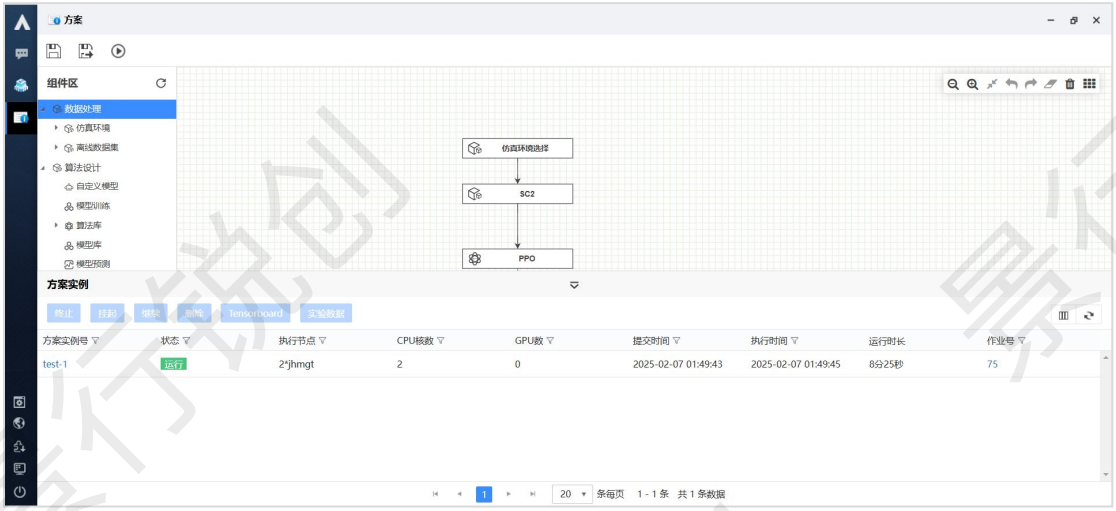


删除方案

2.3.4 方案实例

方案实例中以列表的形式记录了所有方案的历史运行记录，用户可以通过实例号、名称等筛选功能快速过滤实例。

用户可在方案实例应用中查看自己的所有实例，或在具体的方案设计页面中，通过点击画布下方的方案实例滑块，在展开的面板中查看当前方案实例的信息，亦可对其进行终止、挂起等操作。

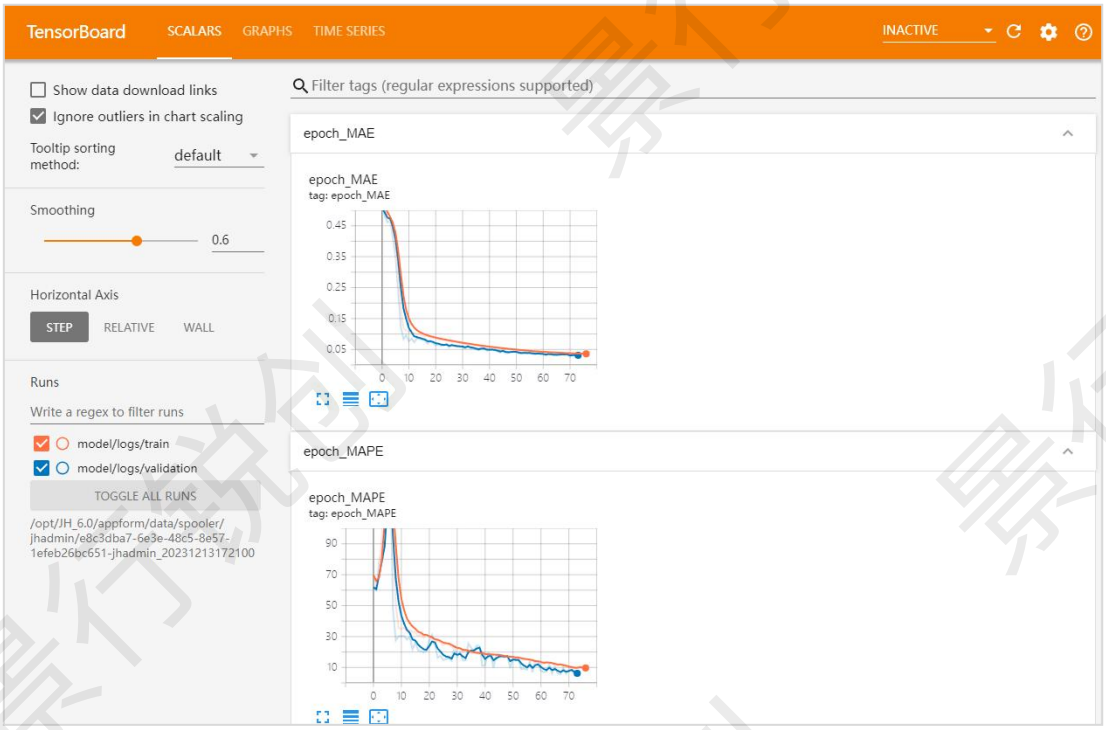


方案实例列表

➢ 功能按钮：

方案实例支持对方案实例进行“终止”“挂起”“继续”“删除”以及打开 Tensorboard 操作。

当强化学习的方案运行完成后，点击“Tensorboard”按钮可以以图表形式查看当前实例运行结果。

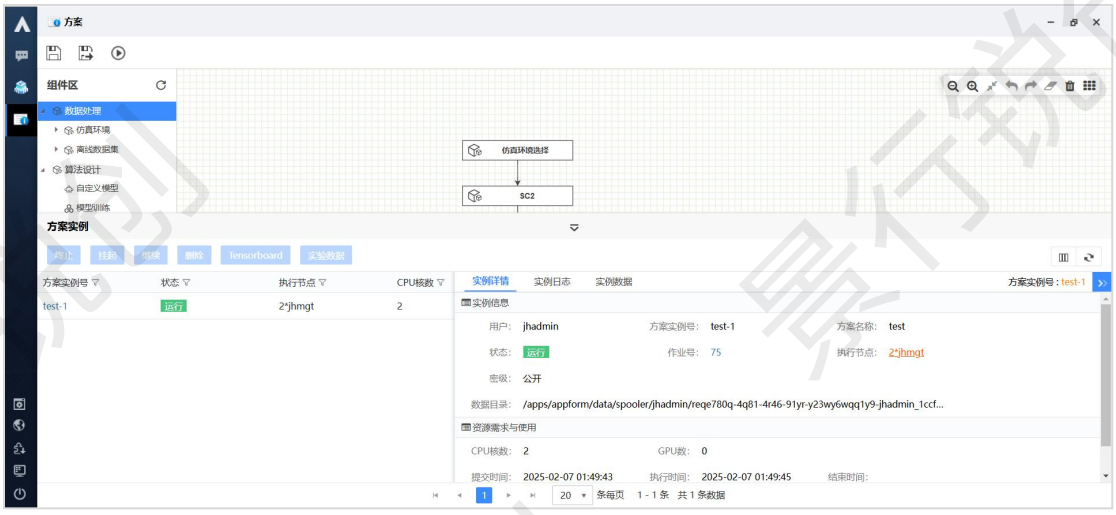


Tensorboard

➢ 方案实例滑块：

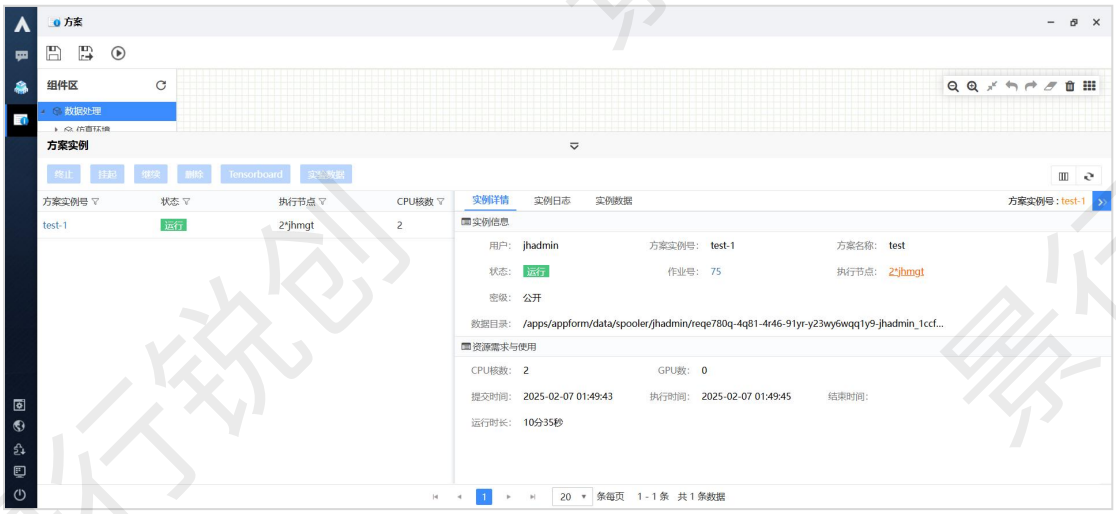
点击“方案实例号”或者双击方案实例列表中的某条记录，会打开右侧滑块。滑块有三个标签页，分别为实例详情、实例日志和实例数据。

当组件中包含“模型评估”组件，且方案运行成功，滑块会有四个标签页，分别为实例详情、实例日志、实例数据。如下图所示：



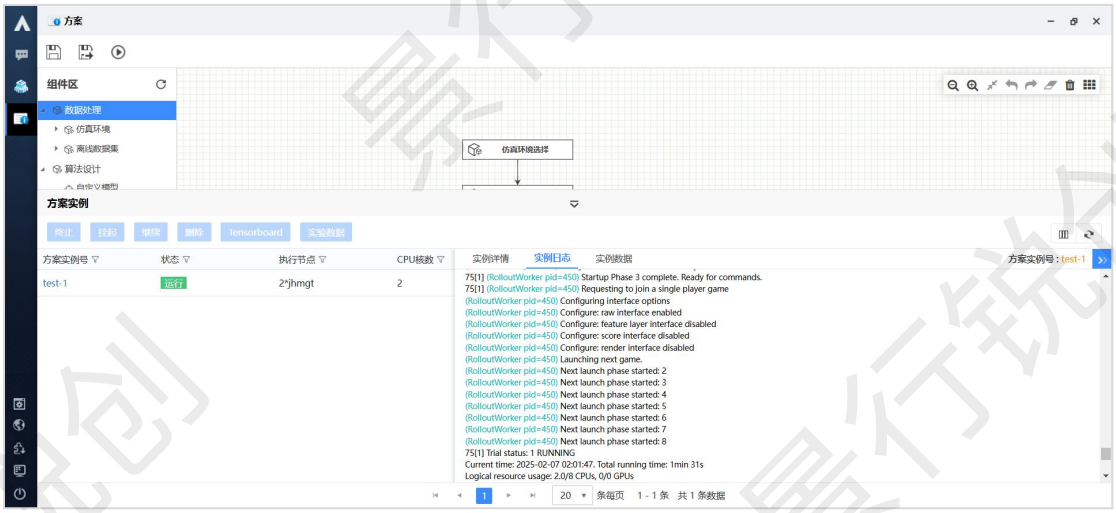
方案设计

实例详情：显示方案实例的基本信息、资源信息。点击作业号，可以查看该实例关联的作业信息。实例详情中包含方案实例关联的数据保存目录。



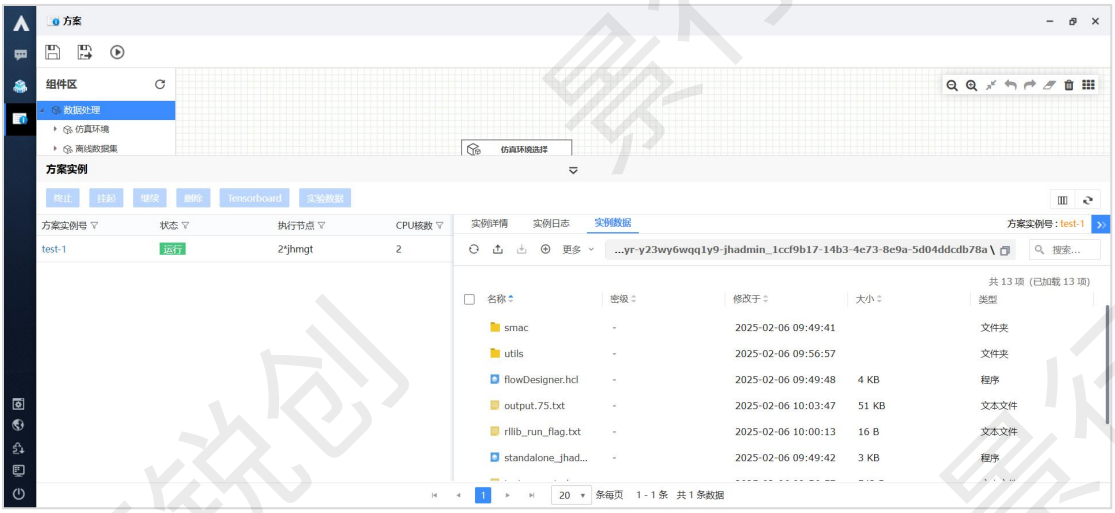
方案实例滑块

实例日志：点击实例日志按钮，打开实例日志界面。方案实例的状态处于正在运行时，该实例日志会实时输出。方案实例的状态处于完成时，会显示该方案实例的所有日志。



方案实例日志

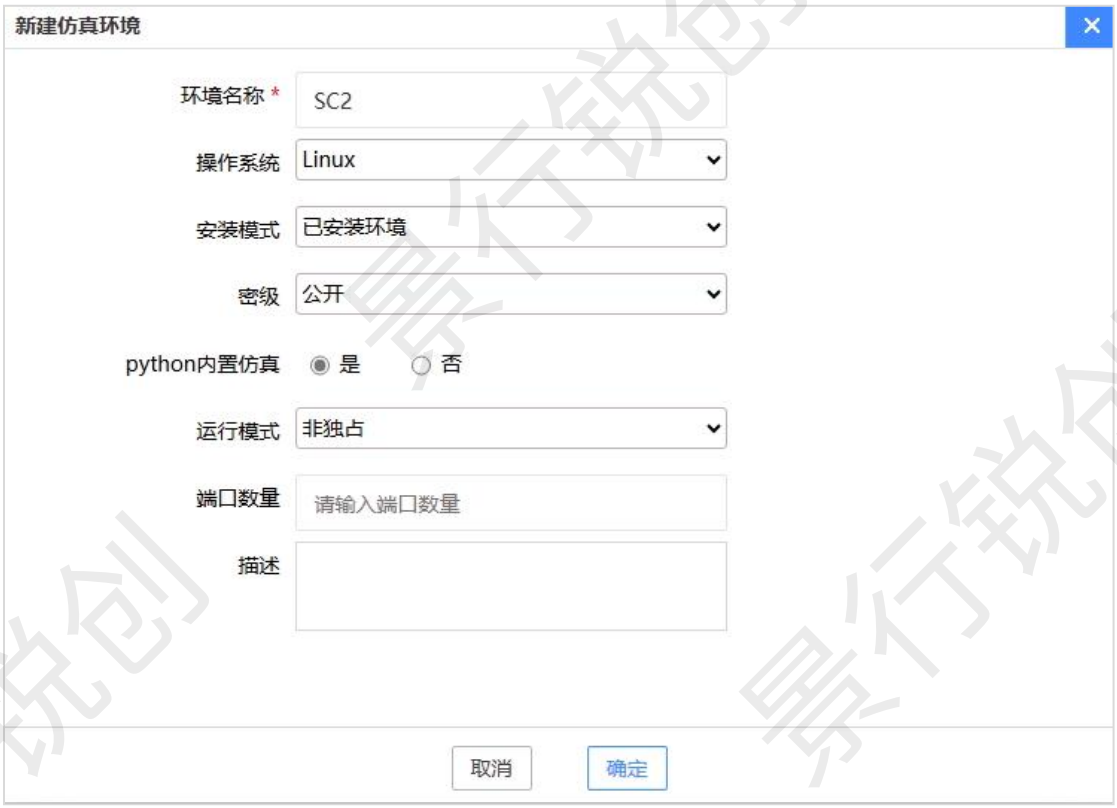
实例数据：点击实例数据按钮，打开实例数据界面，显示该方案实例包含的所有相关文件。



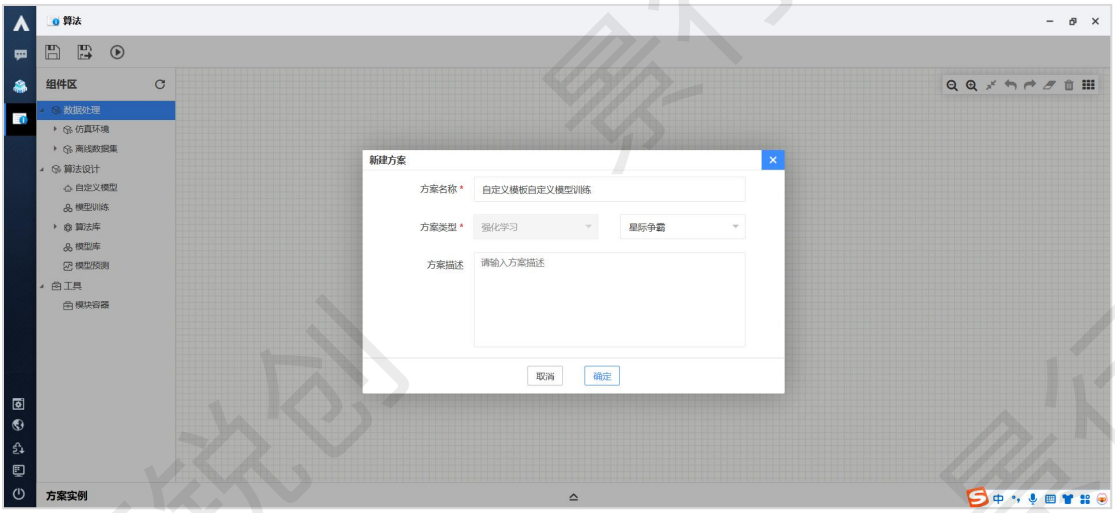
方案实例数据

2.3.5 使用自定义开发模板，自定义模型进行训练/测试

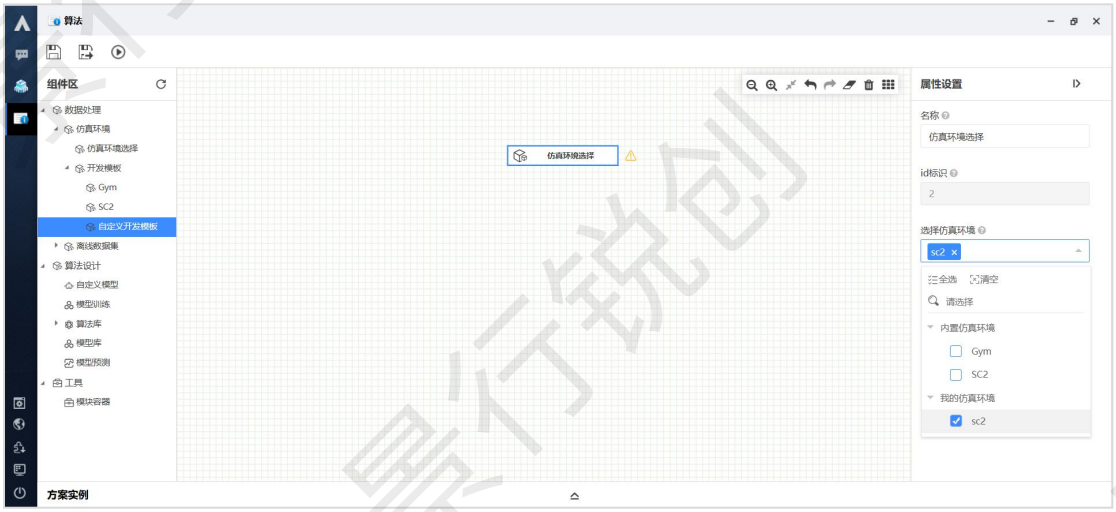
- 新建 python 内置仿真环境



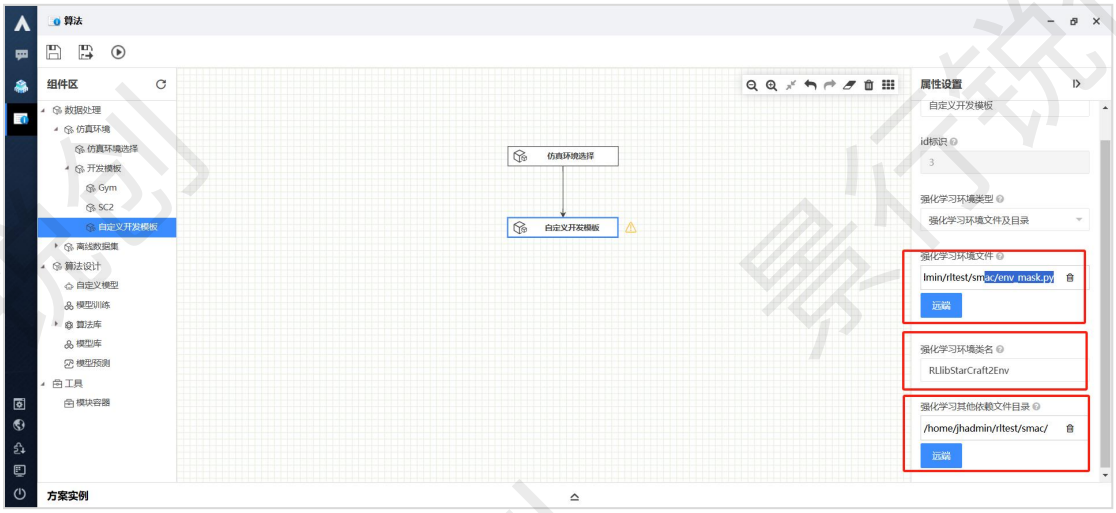
- 新建强化学习方案



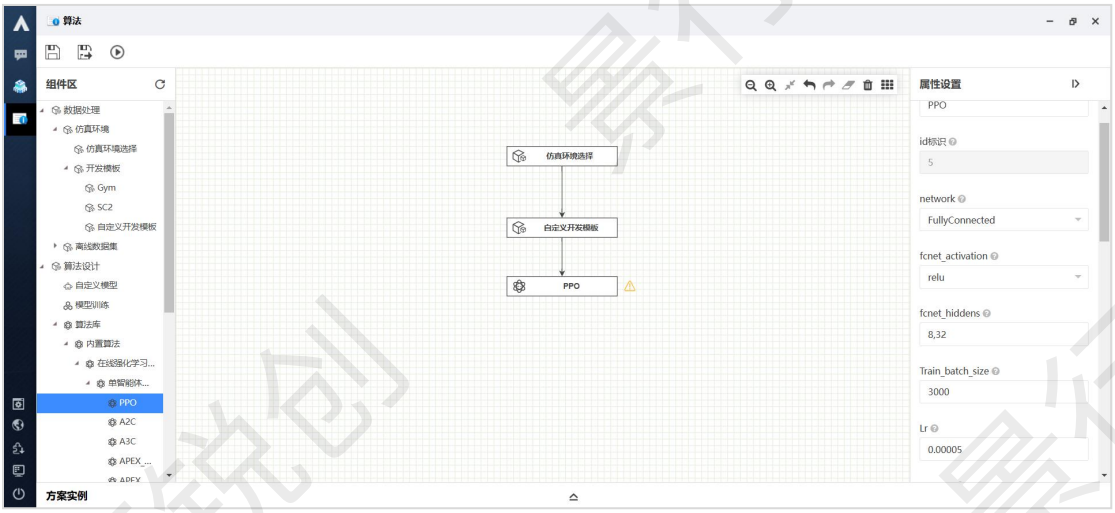
➢ 选择“仿真环境选择”组件，我的仿真环境中选中新增的仿真环境



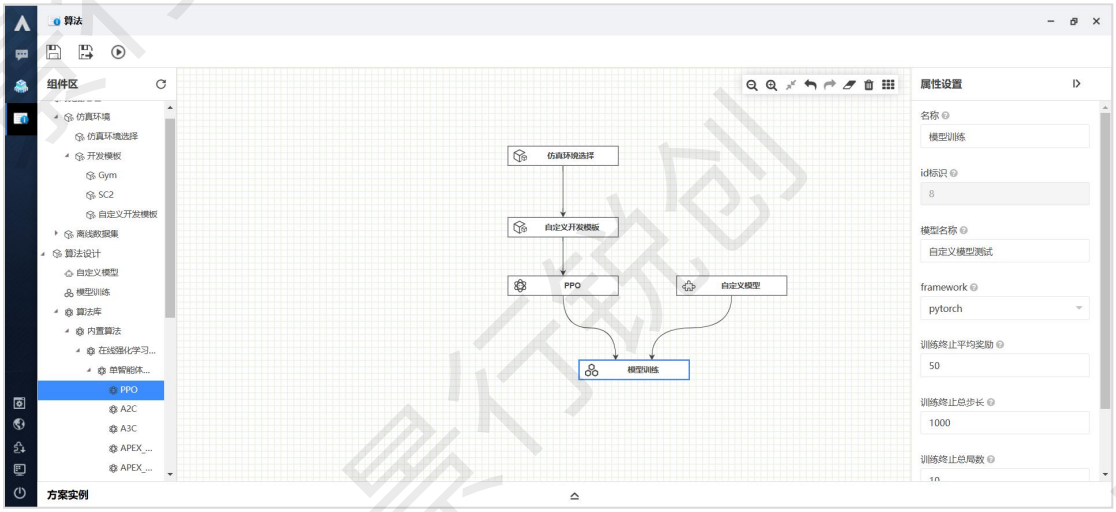
➢ 选择“自定义开发模板”组件，上传强化学习环境及目录



➢ 选择算法库-在线强化学习算法-单智能体算法-PPO



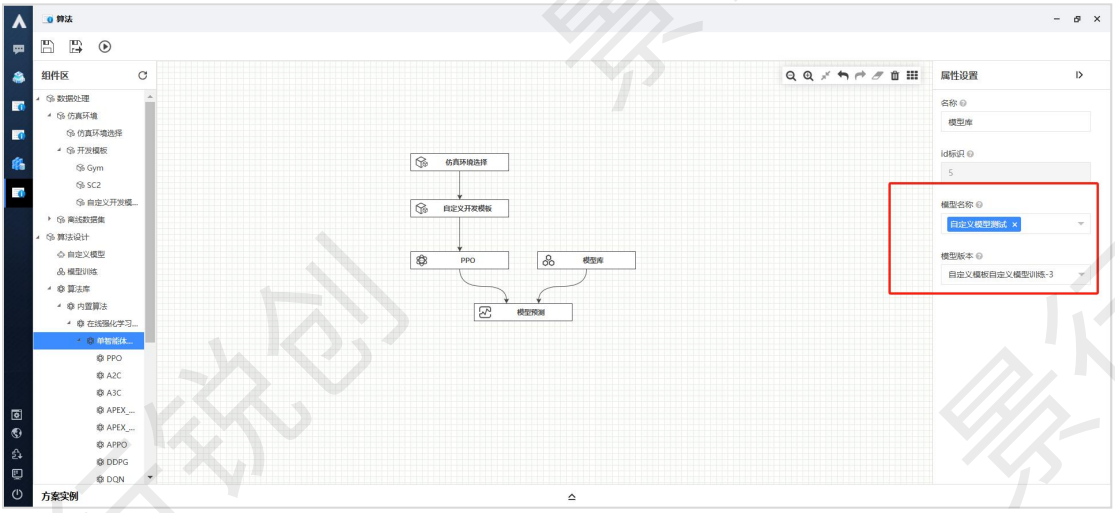
➢ 选择自定义模型组件，选择模型训练



➢ 训练完成后，模型库生成强化学习模型

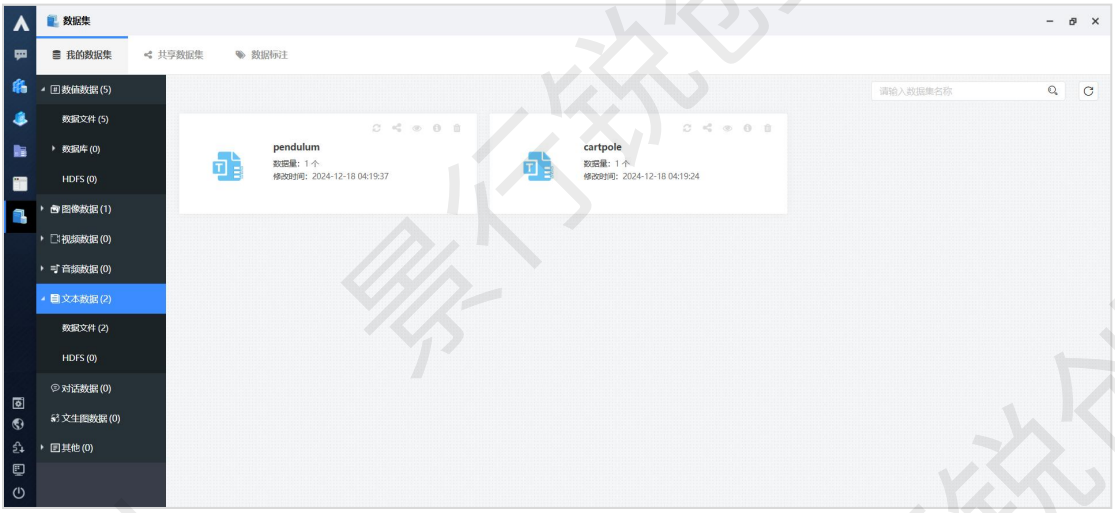
➢ 和上述步骤类似，加载已生成的模型库模型选择模型预测组件，进行强

化学习模型预测

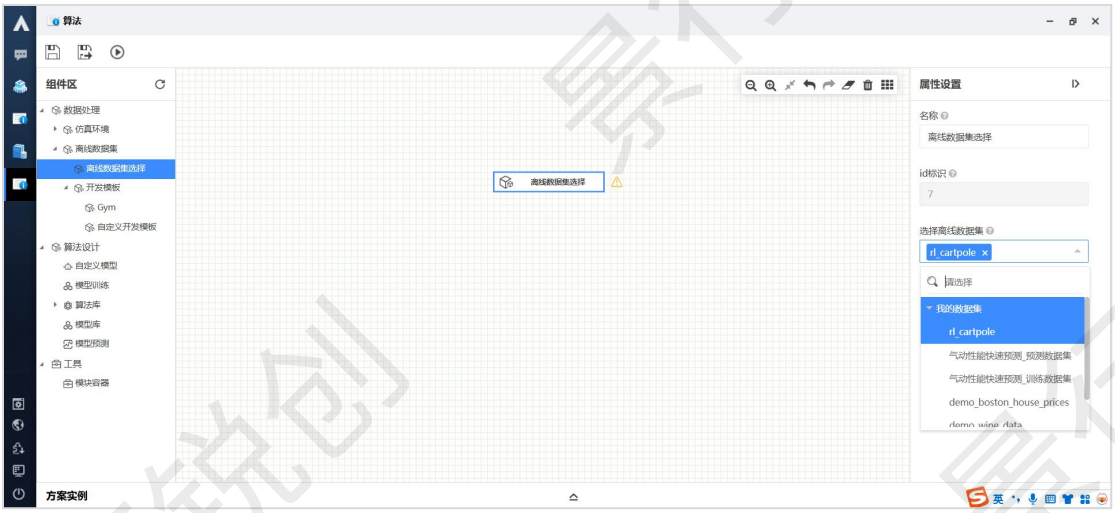


2.3.6 内置离线数据集进行模型训练、模型预测

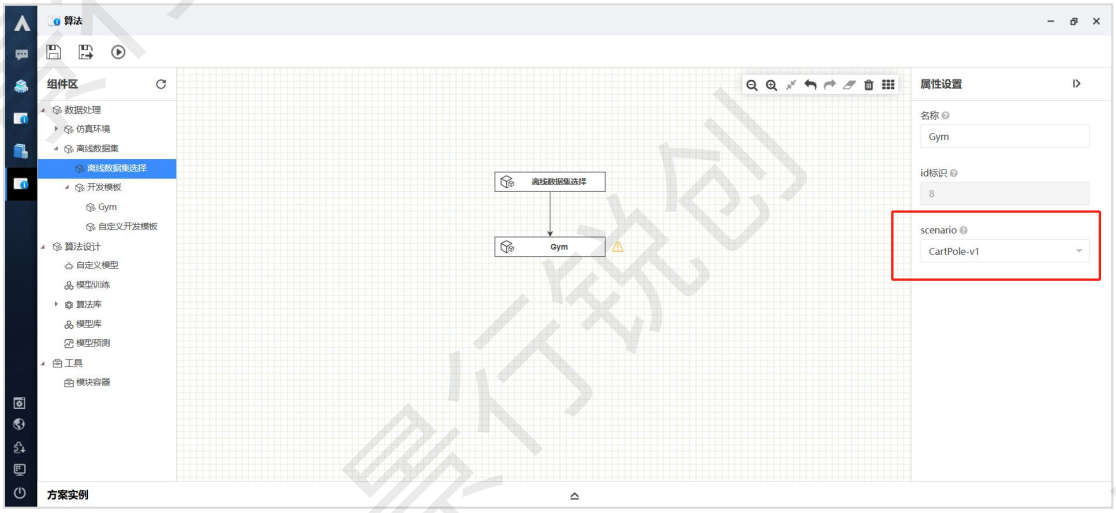
- 新建数据集-文本数据-数据文件



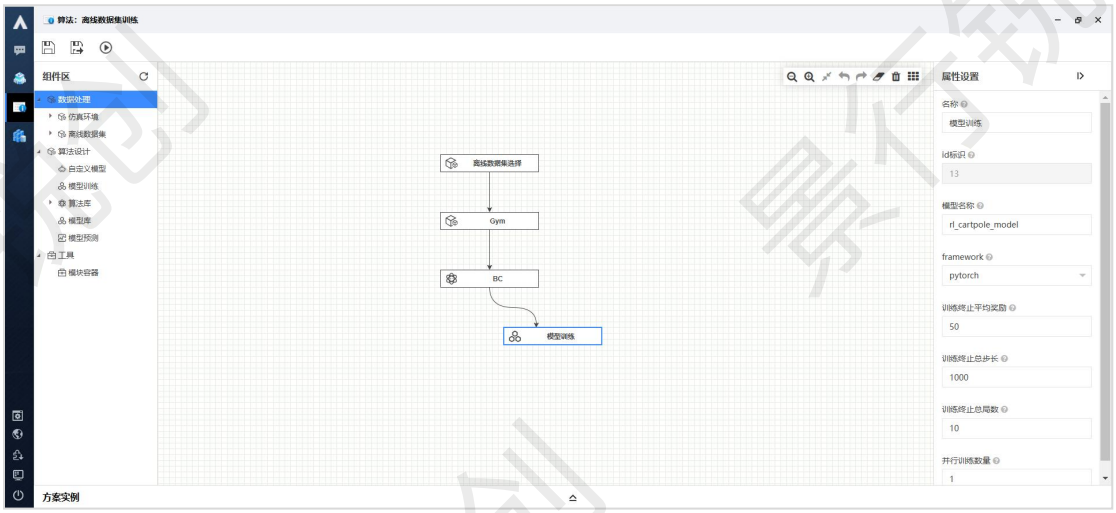
- 选择“离线数据集”组件，我的离线数据集中选中新增的离线数据集



➢ 选择开发模板-Gym，注意右侧 scenario 应与数据集相对应。

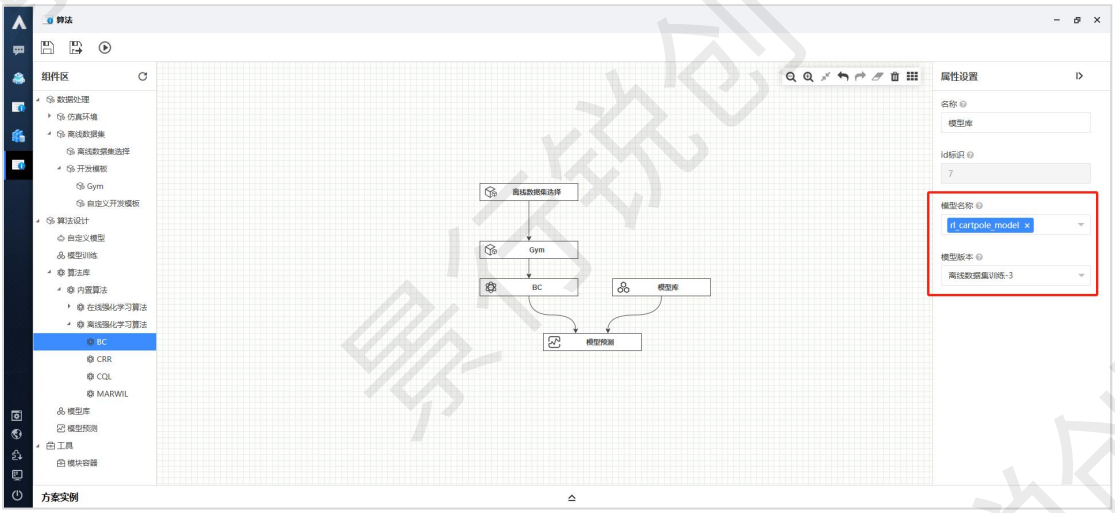


➢ 选择算法库-离线强化学习算法-BC，选择模型训练（离线强化学习算法不支持自定义模型训练）



➢ 训练完成后，在模型库生成离线强化学习模型

➢ 和上述步骤类似，加载已生成的模型库模型选择模型预测组件，进行强化学习模型预测



2.4 数据集

景行锐创人工智能平台提供了丰富的数据集管理功能，满足了不同类型数据在不同训练场景下的管理需求，支持数值、图像、视频、音频、文本以及其他类型数据的管理，也支持进行用户数据的在线标注。

数据集中包含“我的数据集”“共享数据集”和“数据标注”功能，在“我的数据集”和“共享数据集”下可以创建“数值数据”“图像数据”“视频数据”“音频数据”“文本数据”和“其他”数据集。

数据集同时支持数据集的共享功能，让不同用户之间可以相互共享数据，并

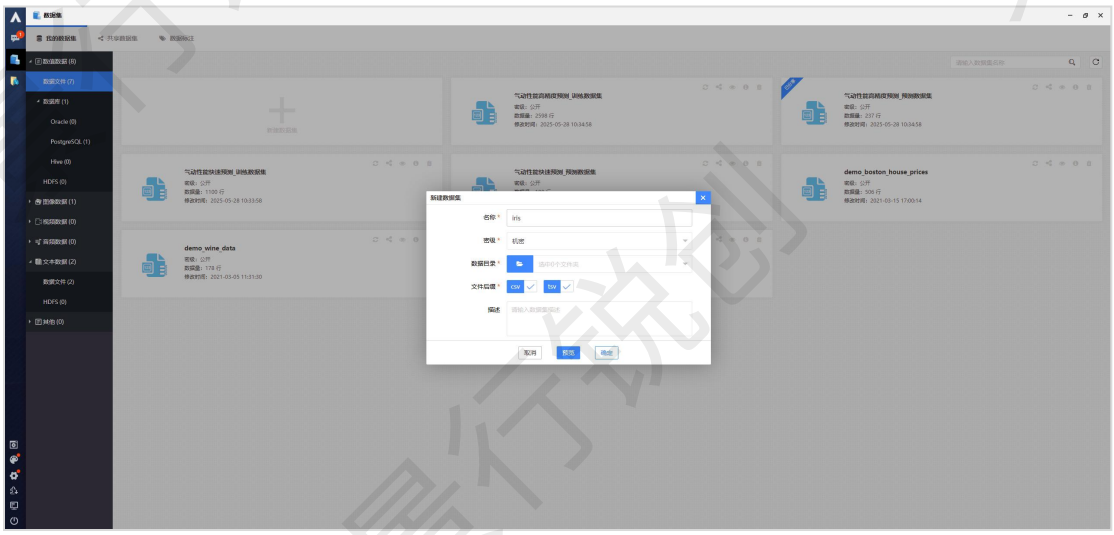
支持设置共享的权限。

方案设计中支持以可视化方式引用数据集，可以使用已有的数据集进行数据处理和模型训练。

2.4.1 数值数据

2.4.1.1 “数值数据-数据文件”数据集

新建“数值数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“数值数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“数值数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集名称，输入任意符合命名规则的名称即可。

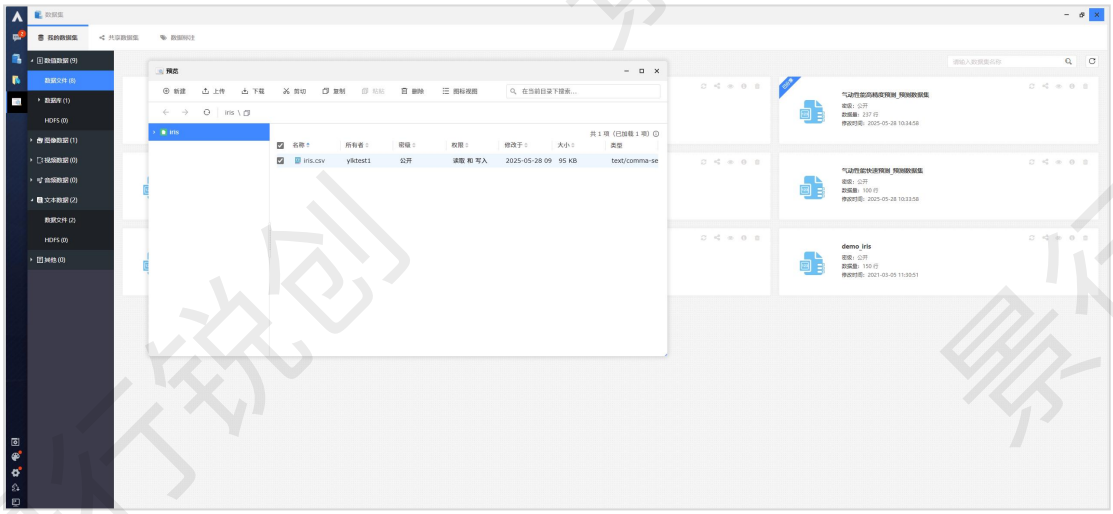
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中选择数据目录即可。当数据目录中包含多个数据文件时，需要保证每个数据文件的表头相同，否则会导致新建数据集失败。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 2 种，分别是 csv 和 tsv，可单选和全选，至少选择其中一项。

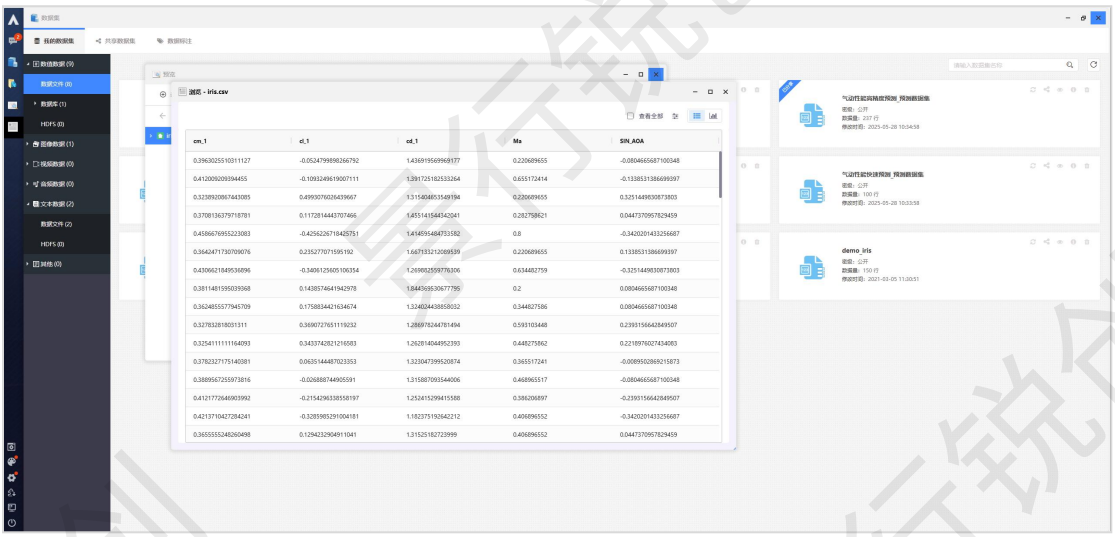
描述：输入该数据集的描述信息，可选。

配置完成后，点击“预览”按钮，即可预览数据集的数据文件，如下图所示：



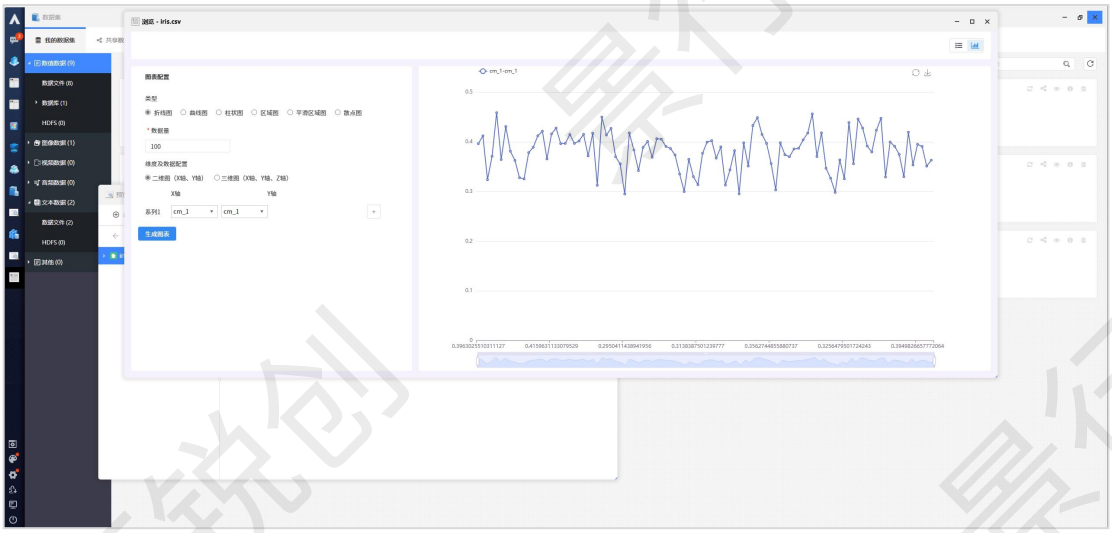
预览“数值数据-数据文件”数据集

双击文件，可以预览数据文件的内容，如下图所示：



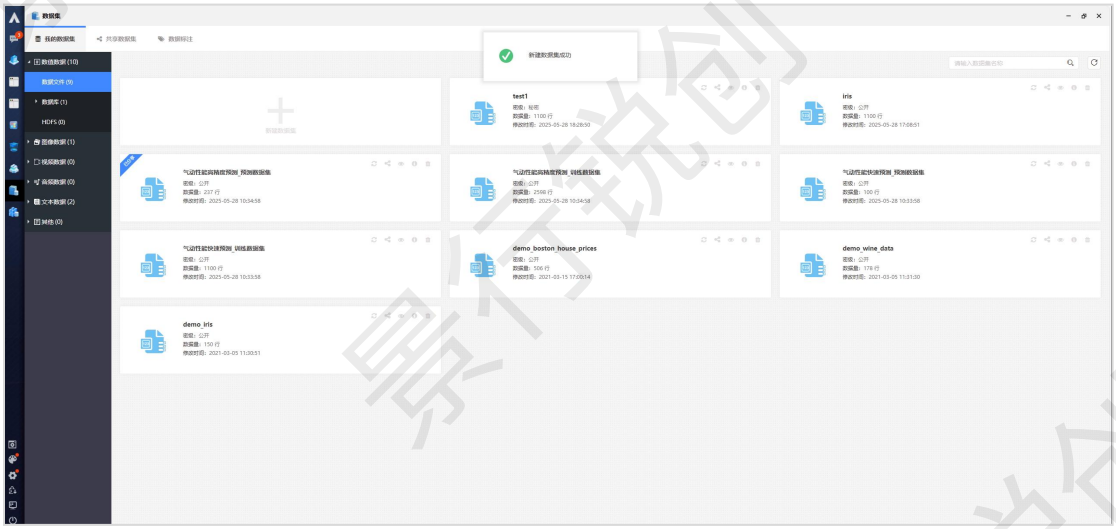
以列表视图形式，预览“数值数据-数据文件”数据集

点击浏览窗口右上角的“图表视图”按钮，将列表视图界面切换至图表视图界面，如下图所示：



以图表视图形式，预览“数值数据-数据文件”数据集

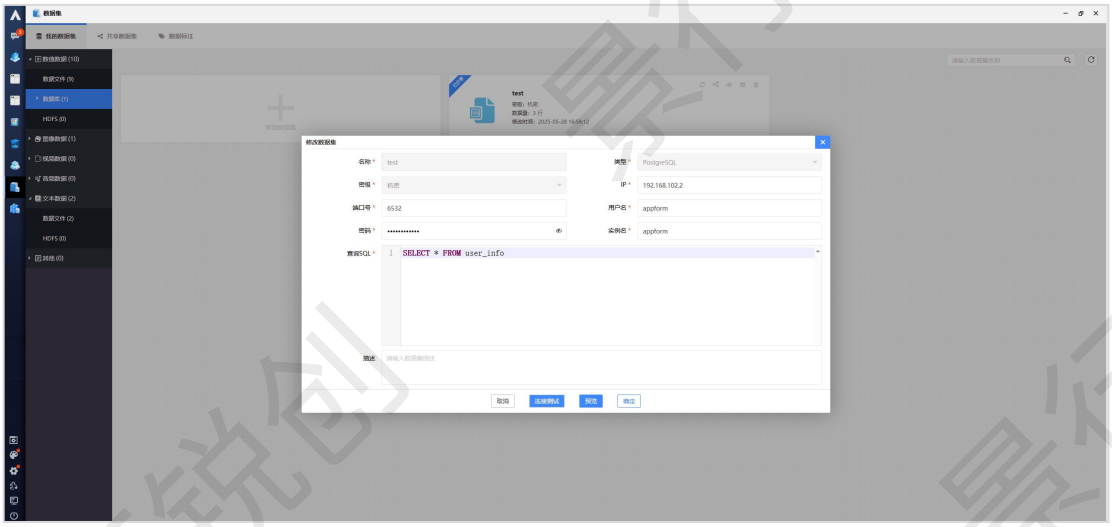
点击“确定”按钮，新建数据集，如下图所示：



新建“数值数据-数据文件”数据集成功

2.4.1.2 “数值数据-数据库”数据集

以新建“PostgreSQL 数据库”数据集为例，操作步骤如下：依次点击“我的数据集”-“数值数据”-“数据库”-“PostgreSQL”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“数值数据-PostgreSQL 数据库”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

类型：选择数据库的类型，默认回填当前数据分类的类型，如：PostgreSQL。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

IP：输入数据库所在机器的 IP。

端口号：输入数据库使用的端口号。

用户名：输入数据库的用户名。

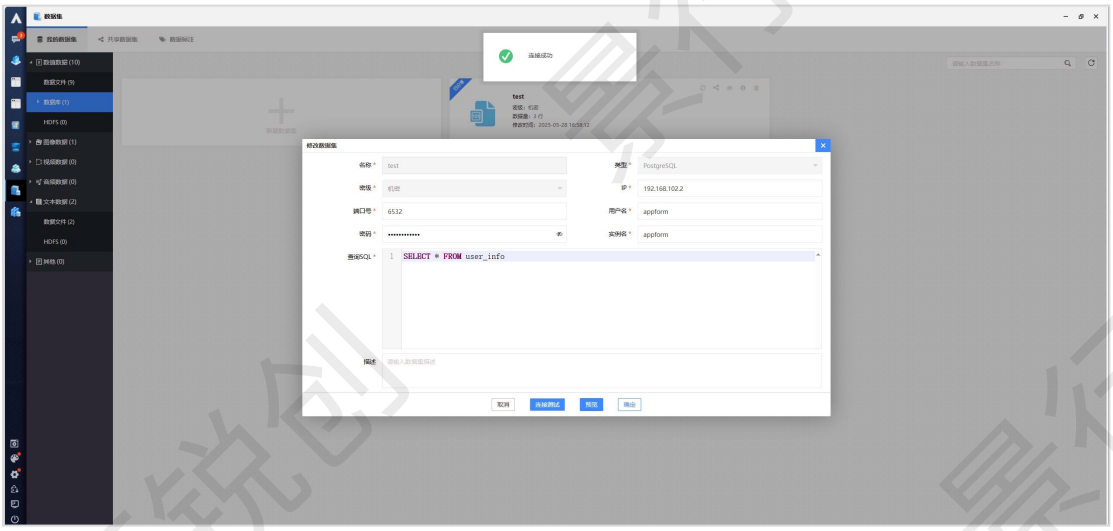
密码：输入数据库用户对应的密码。

实例名：输入数据库名。

查询 SQL：输入 SQL 查询语句，例如：select * from data_set（注意：不要在查询语句后面加分号）。

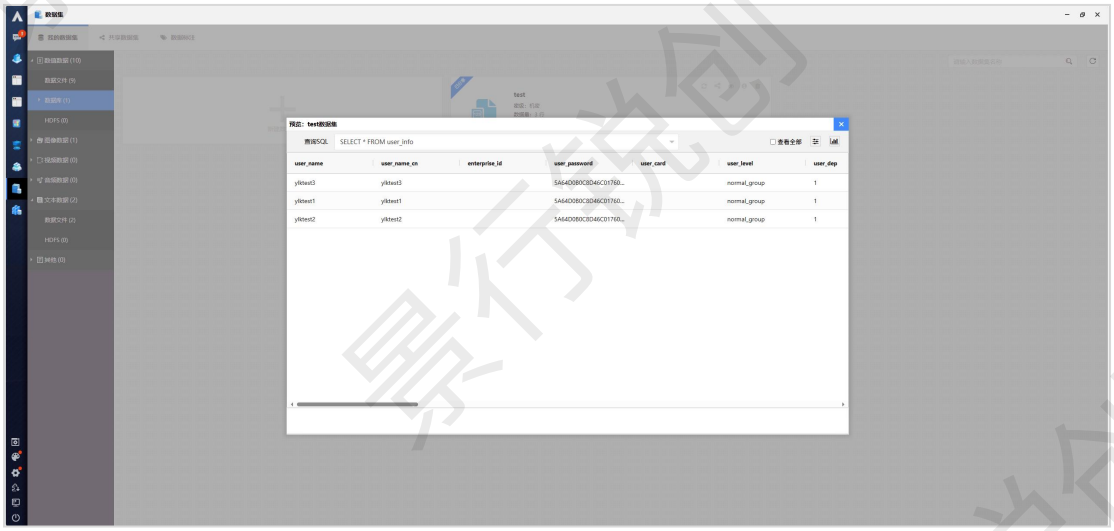
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试数据库的连通性，会有成功和失败提示。连接测试效果如下图所示：



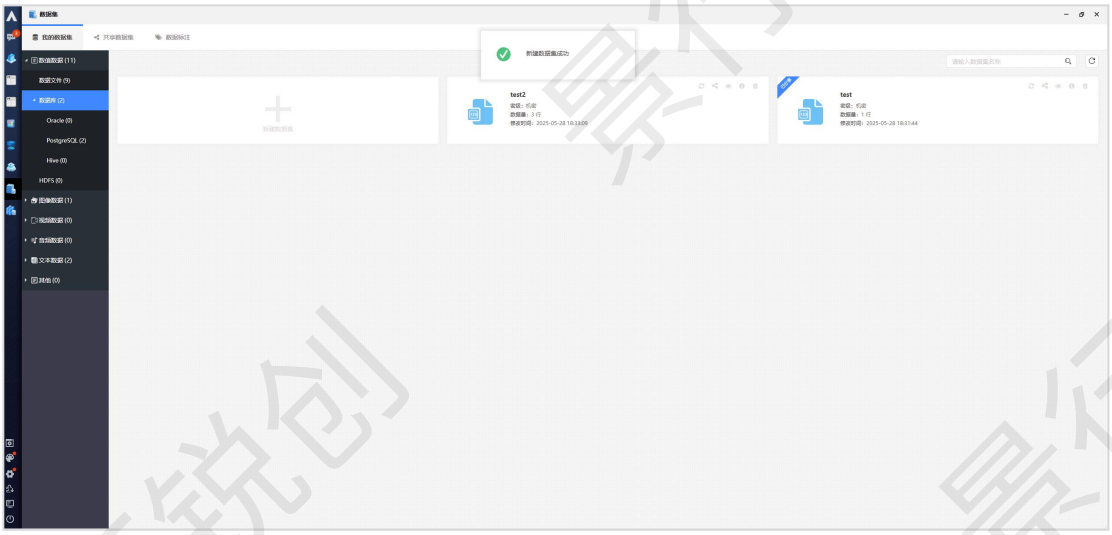
数据库连接测试

点击“预览”按钮，即可查看“查询 SQL”的结果。如下图所示：



预览“数值数据-PostgreSQL 数据库”数据集

点击“确定”按钮，新建数据集。如下图所示：

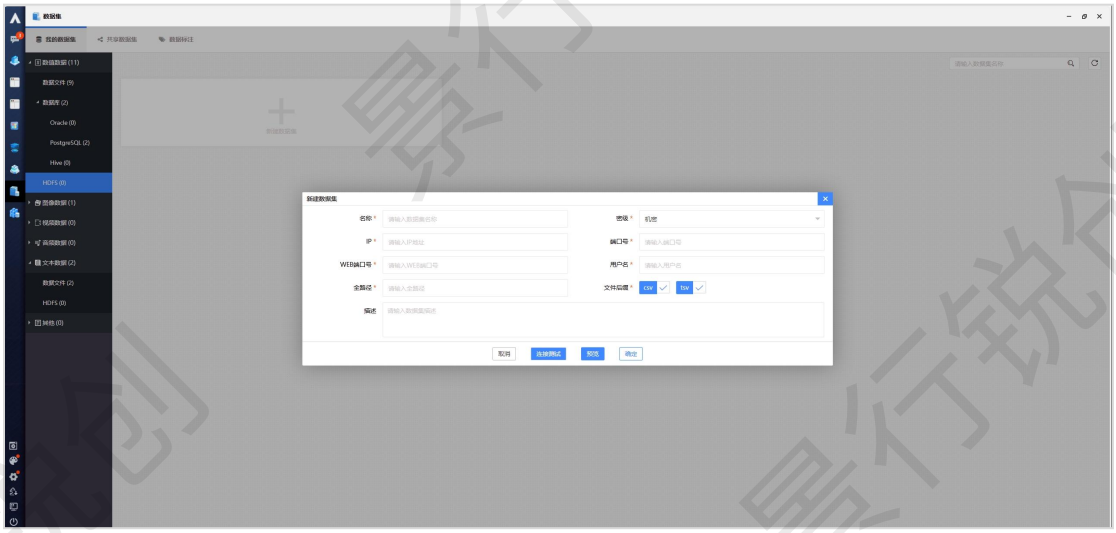


新建“数值数据-PostgreSQL 数据库”数据集成功

2.4.1.3 “数值数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“数值数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“数值数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“数值数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安

全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

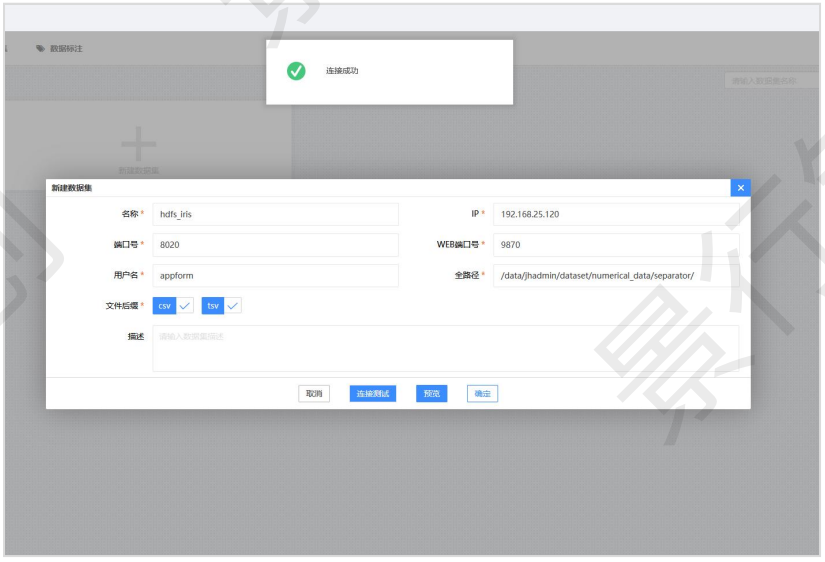
用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 2 种，分别是 csv 和 tsv，可单选和全选，至少选择其中一项。

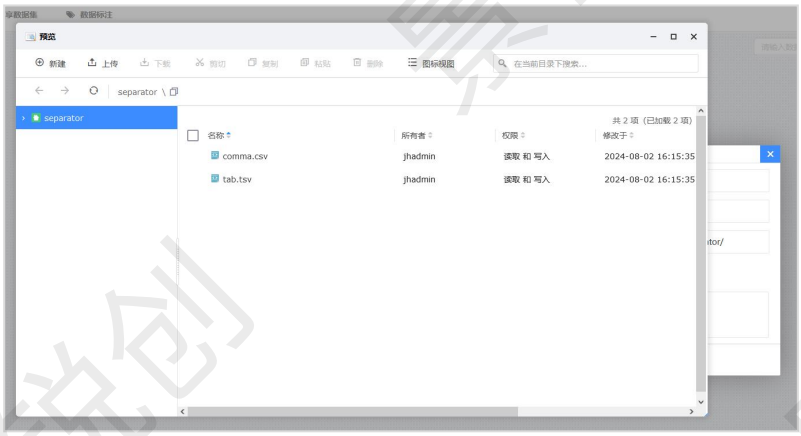
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



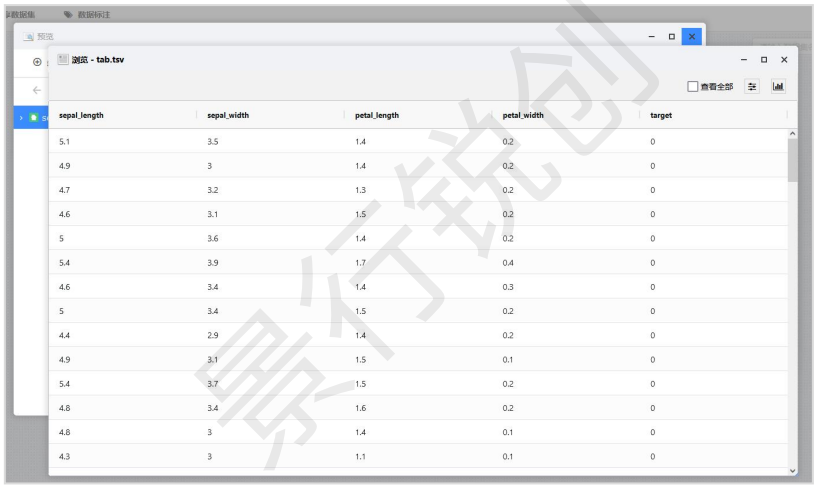
连接测试

点击“预览”按钮，即可预览数据集的数据文件。如下图所示：



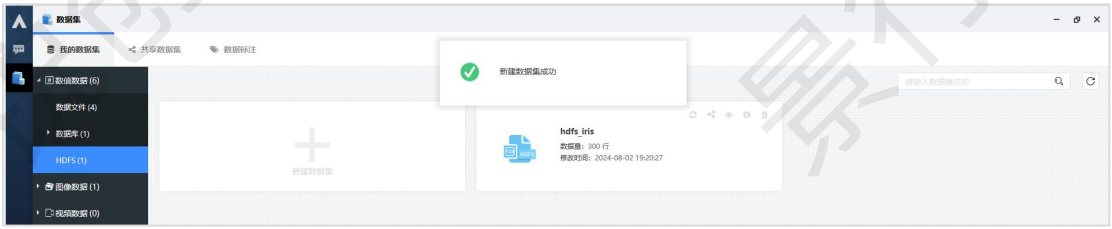
预览“数值数据-HDFS”数据集

双击文件，可以预览数据文件的内容，如下图所示：



预览“数值数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：

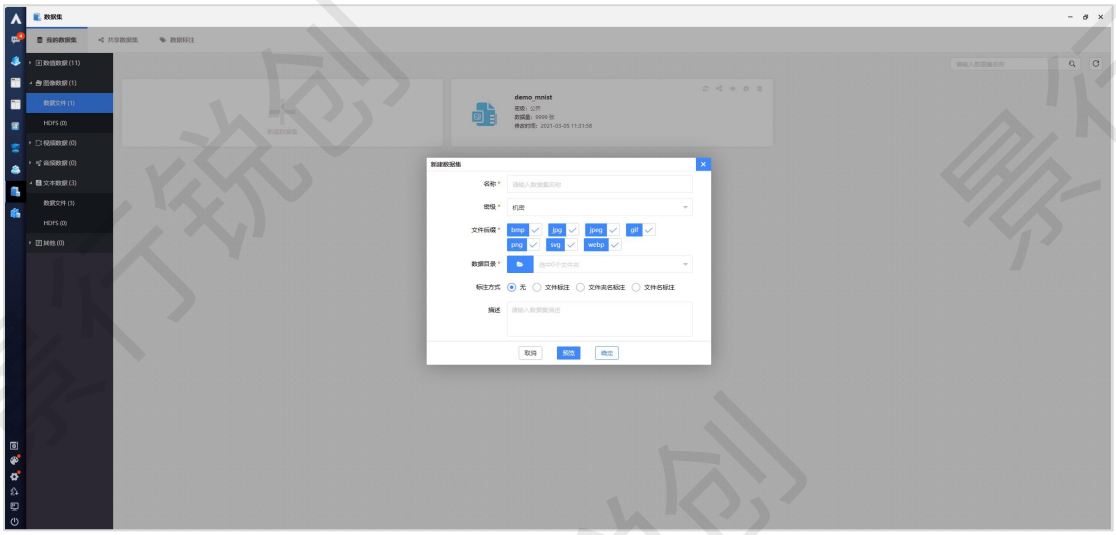


新建“数值数据-HDFS”数据集成功

2.4.2 图像数据

2.4.2.1 “图像数据-数据文件”数据集

新建“图像数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“图像数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“图像数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 bmp、jpg、jpeg、gif、png、svg 和 webp，可单选和全选，至少选择其中一项。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择图像文件夹即可。

标注方式：有 4 种标注方式，如下所示：

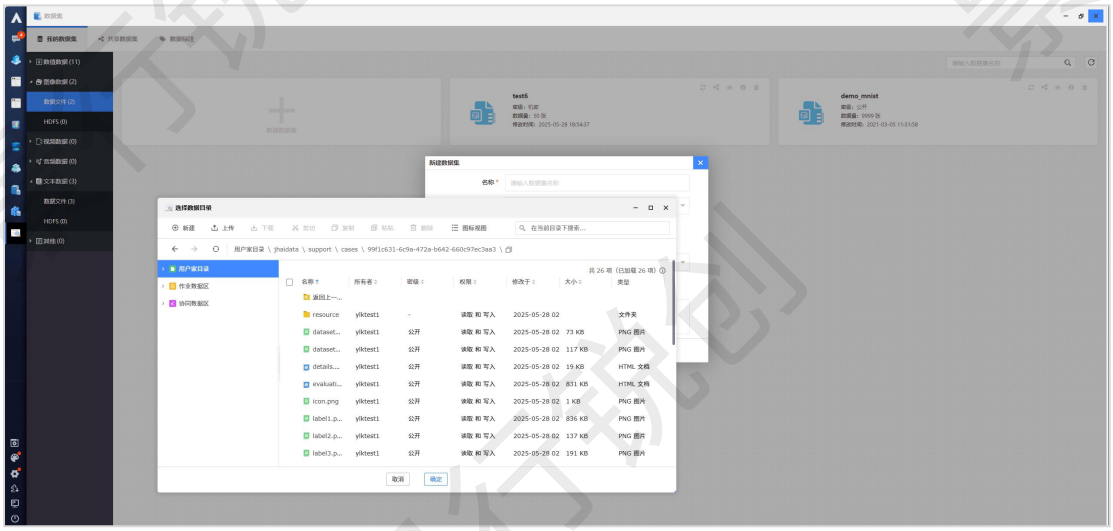
1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文

件夹名来标注类别；

4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

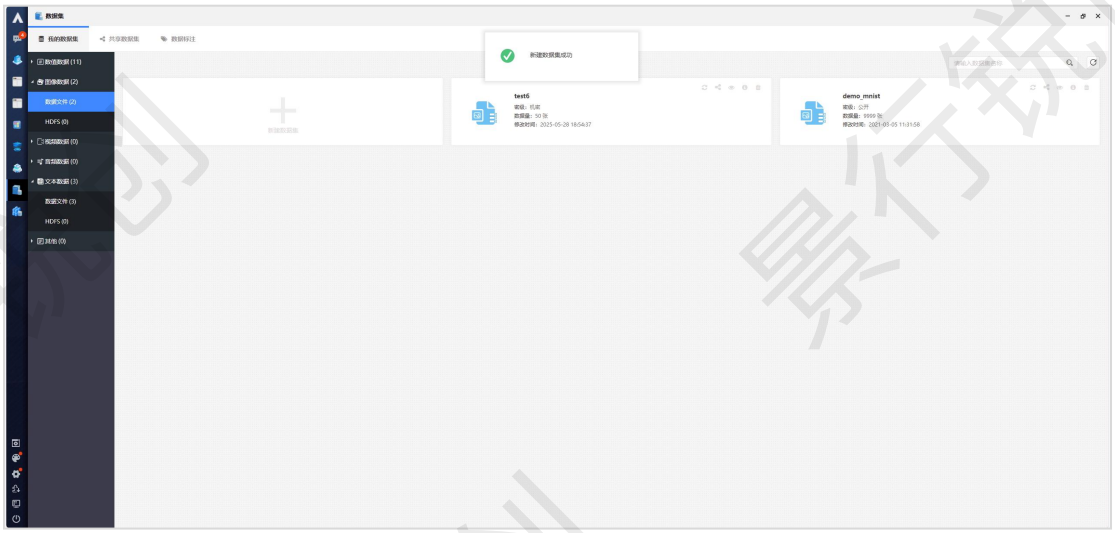
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可预览数据集的数据文件。预览效果如下图所示：



预览“图像数据-数据文件”数据集

点击“确定”按钮，新建数据集。如下图所示：

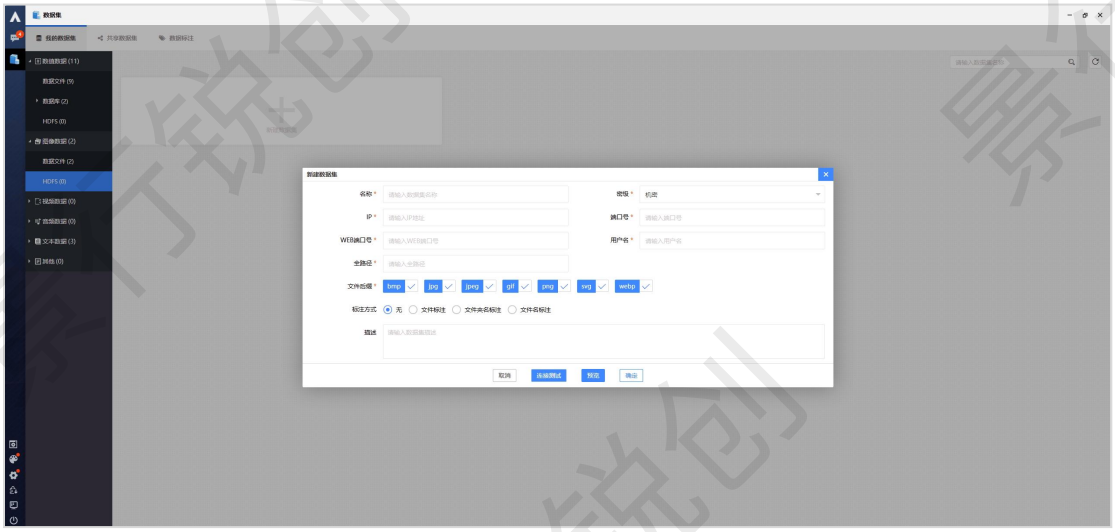


新建“图像数据-数据文件”数据集成功

2.4.2.2 “图像数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“图像数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“图像数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“图像数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管

理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

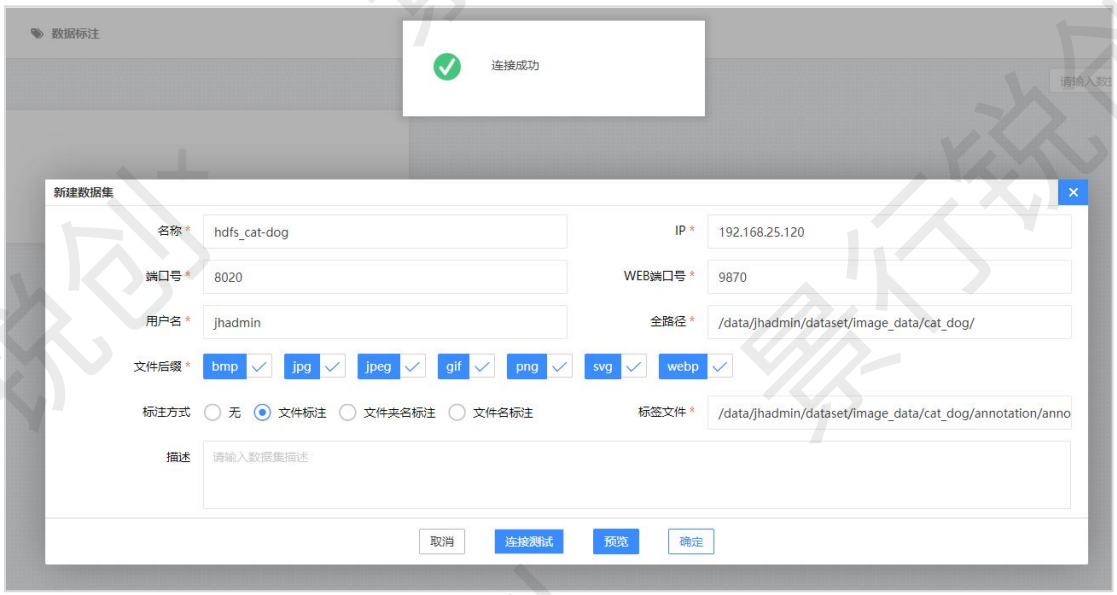
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 bmp、jpg、jpeg、gif、png、svg 和 webp，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

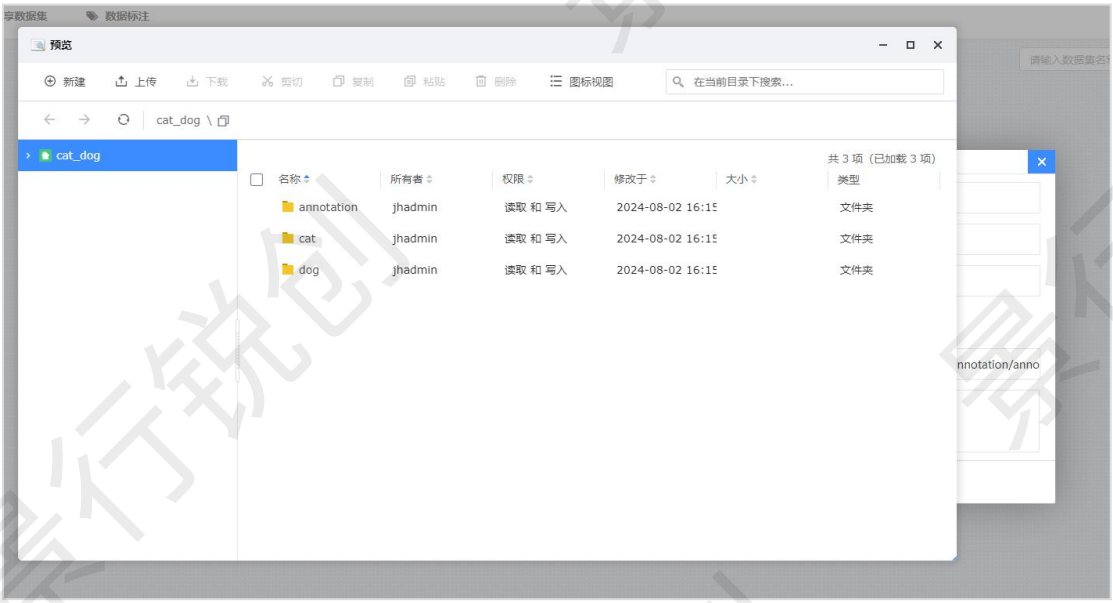
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



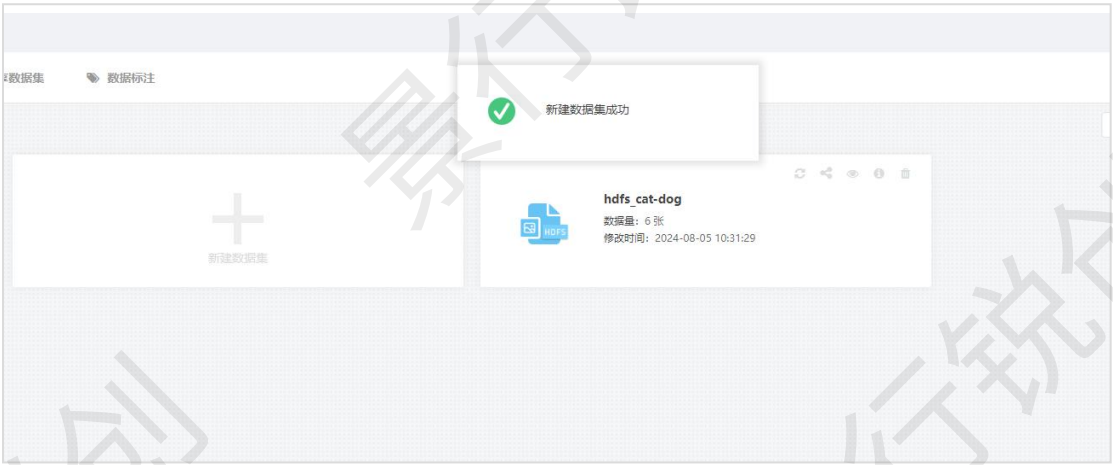
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。如下图所示：



预览“图像数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：



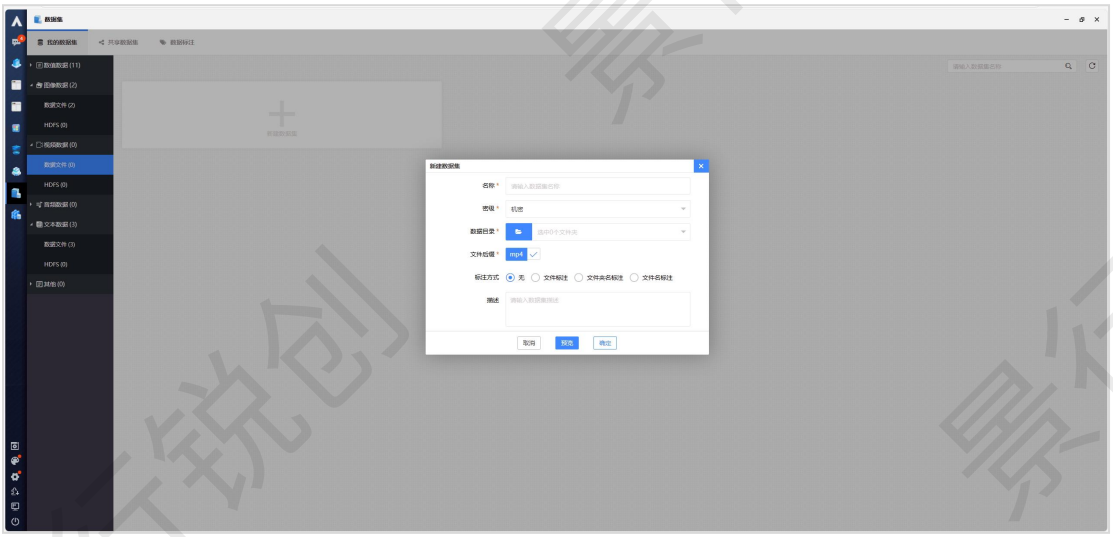
新建“图像数据-HDFS”数据集成功

2.4.3 视频数据

2.4.3.1 “视频数据-数据文件”数据集

新建“视频数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“视频数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建

数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“视频数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择视频文件夹即可。

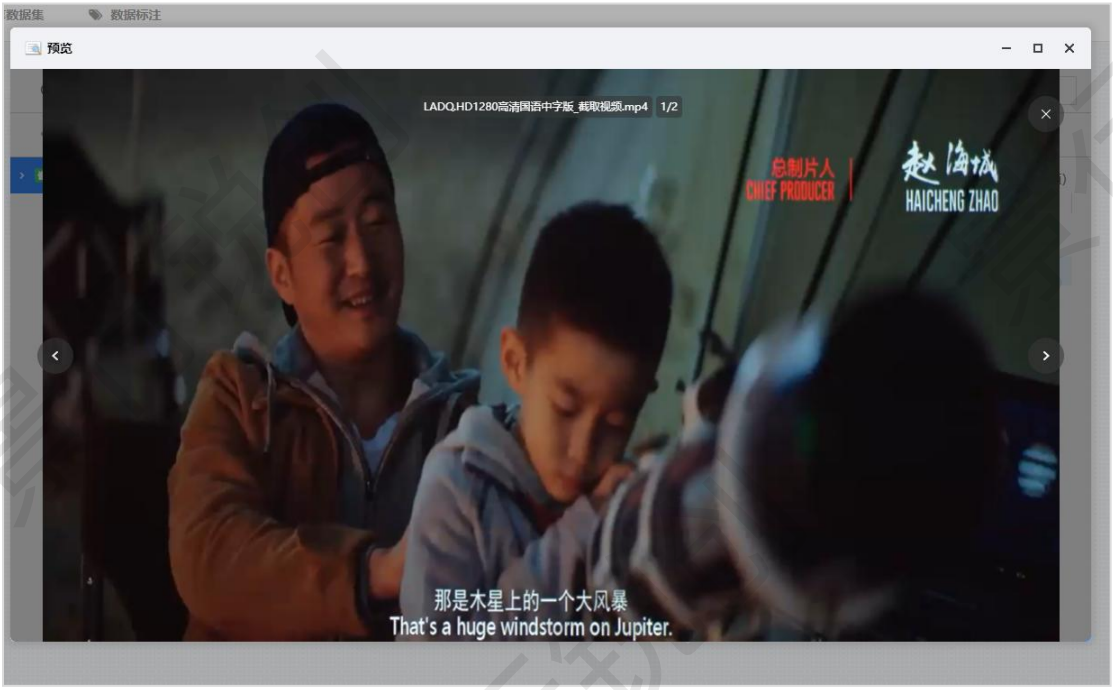
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型仅有 mp4，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

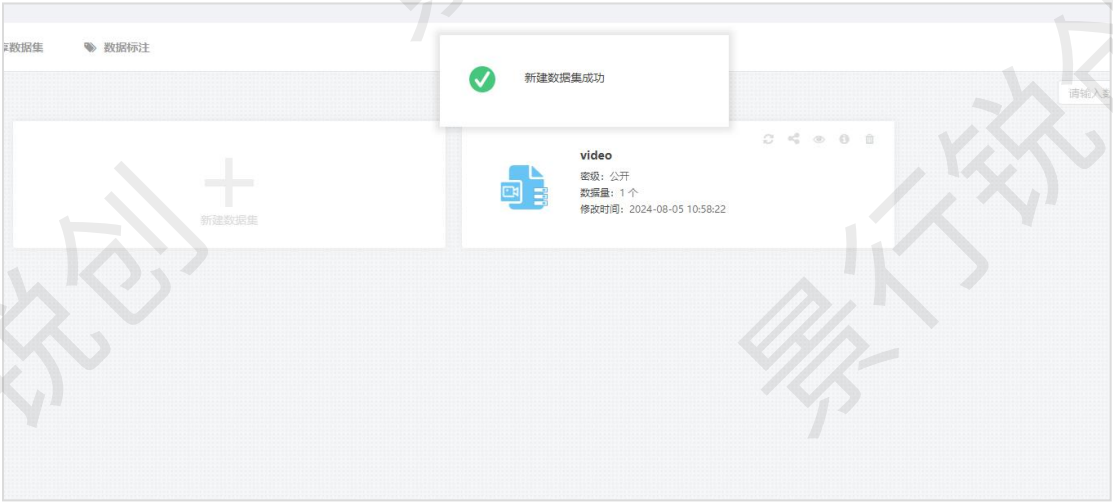
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“视频数据-数据文件”数据集

点击“确定”按钮，新建数据集。如下图所示：

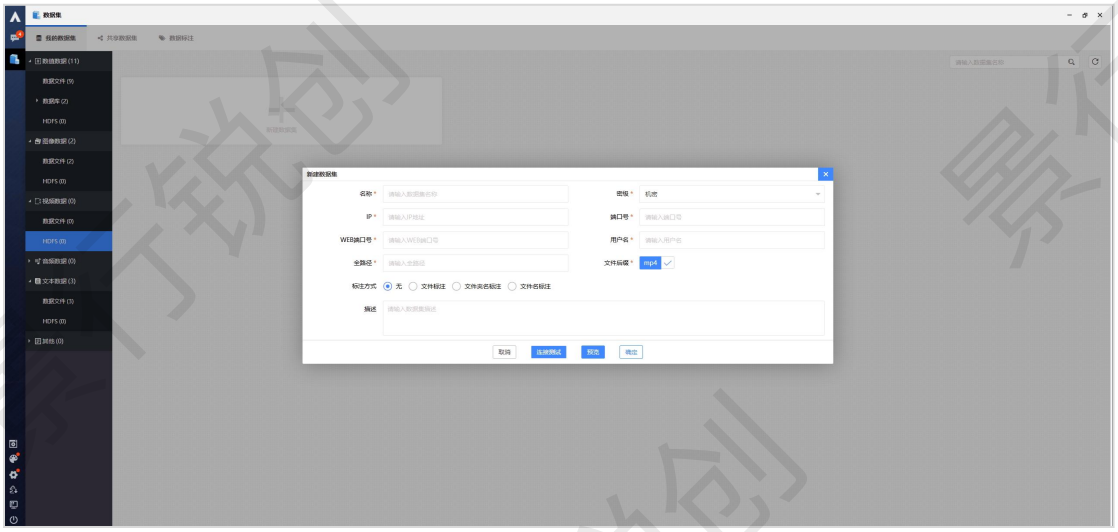


新建“视频数据-数据文件”数据集成功

2.4.3.2 “视频数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“视频数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“视频数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“视频数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”

的值。

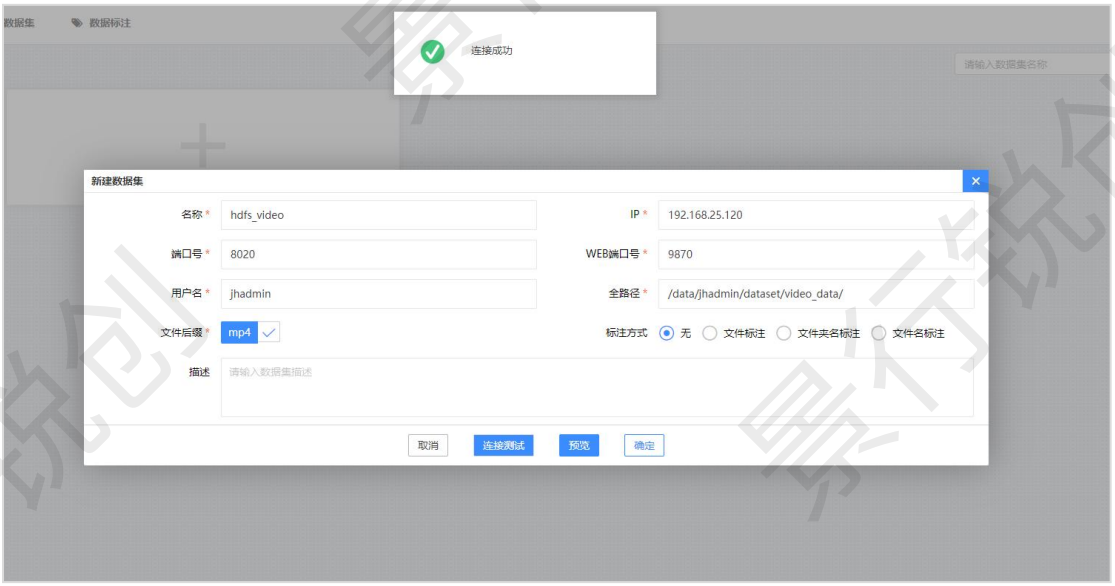
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型仅有 mp4，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：

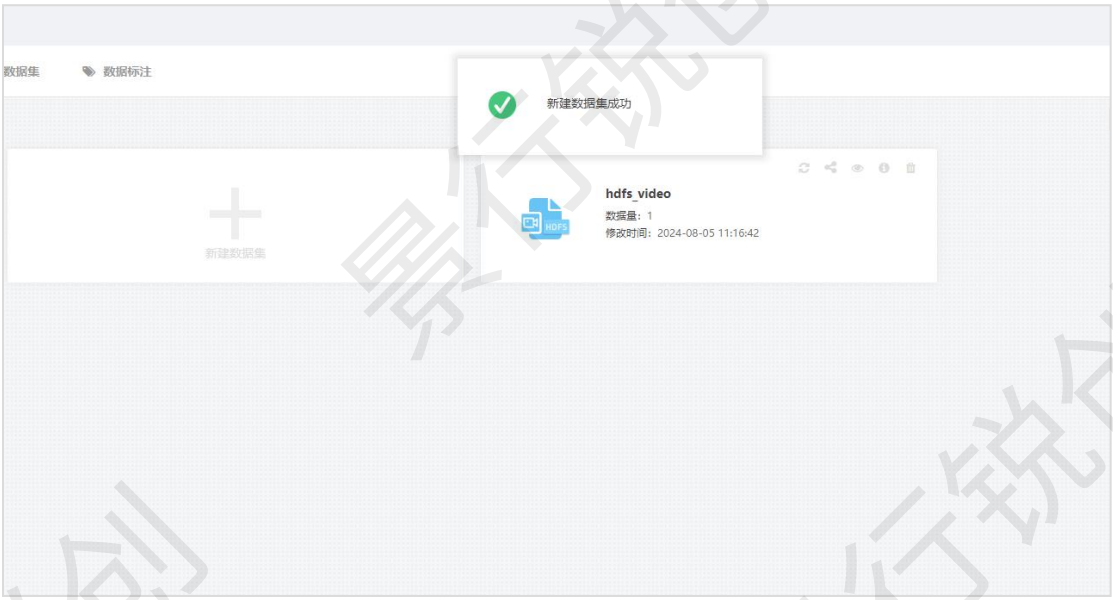


连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。如下图所示：

预览“视频数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：



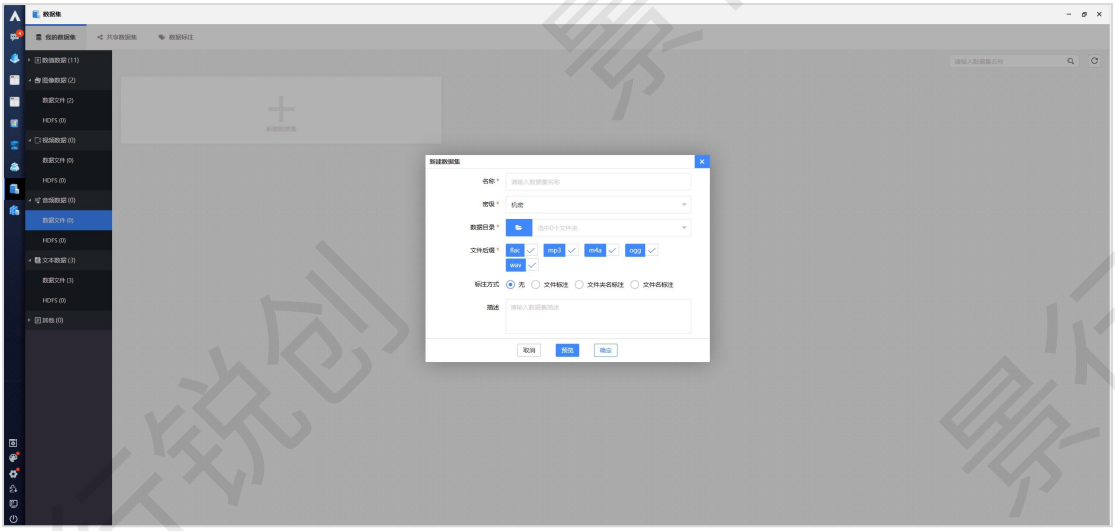
新建“视频数据-HDFS”数据集成功

2.4.4 音频数据

2.4.4.1 “音频数据-数据文件”数据集

新建“音频数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“音频数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建

数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“音频数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择音频文件夹即可。

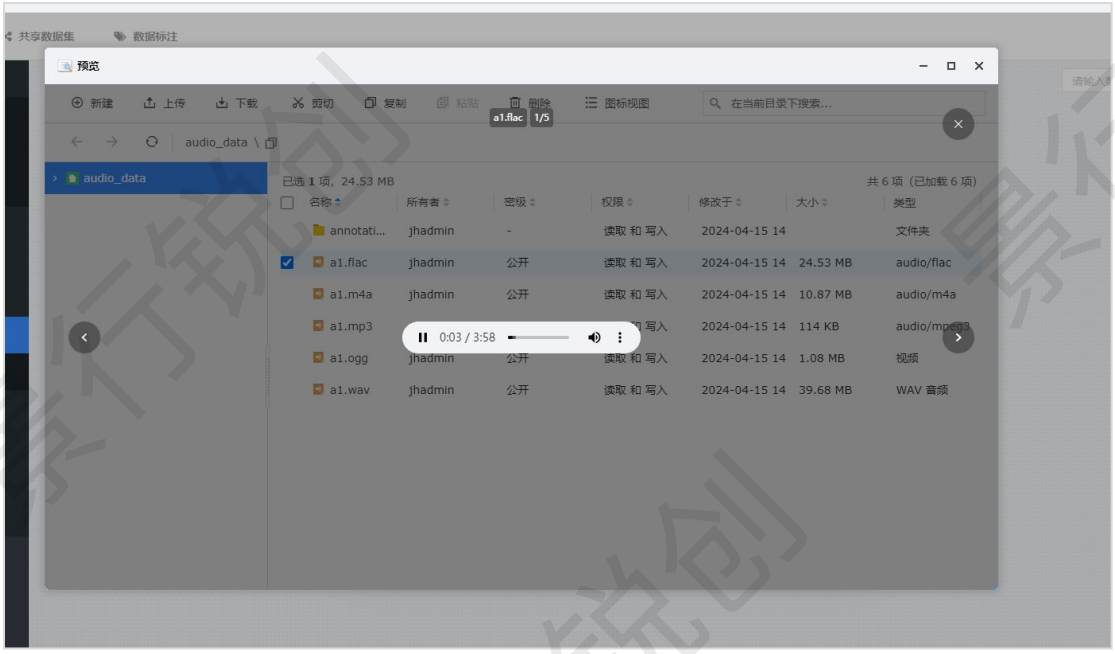
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 flac、mp3、m4a、ogg 和 wav，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“音频数据-数据文件”数据集

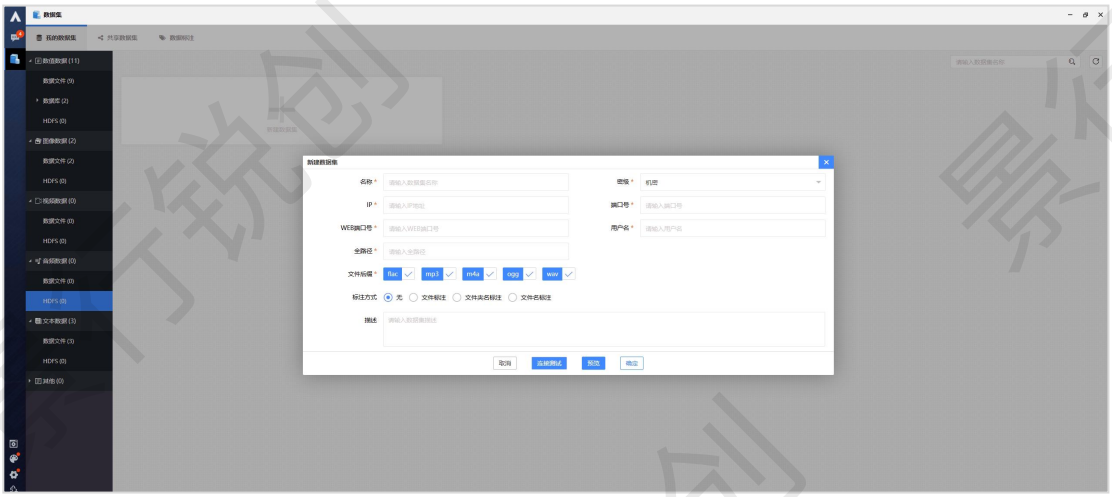
点击“确定”按钮，新建数据集。如下图所示：

新建“音频数据-数据文件”数据集成功

2.4.4.2 “音频数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“音频数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“音频数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“音频数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”

的值。

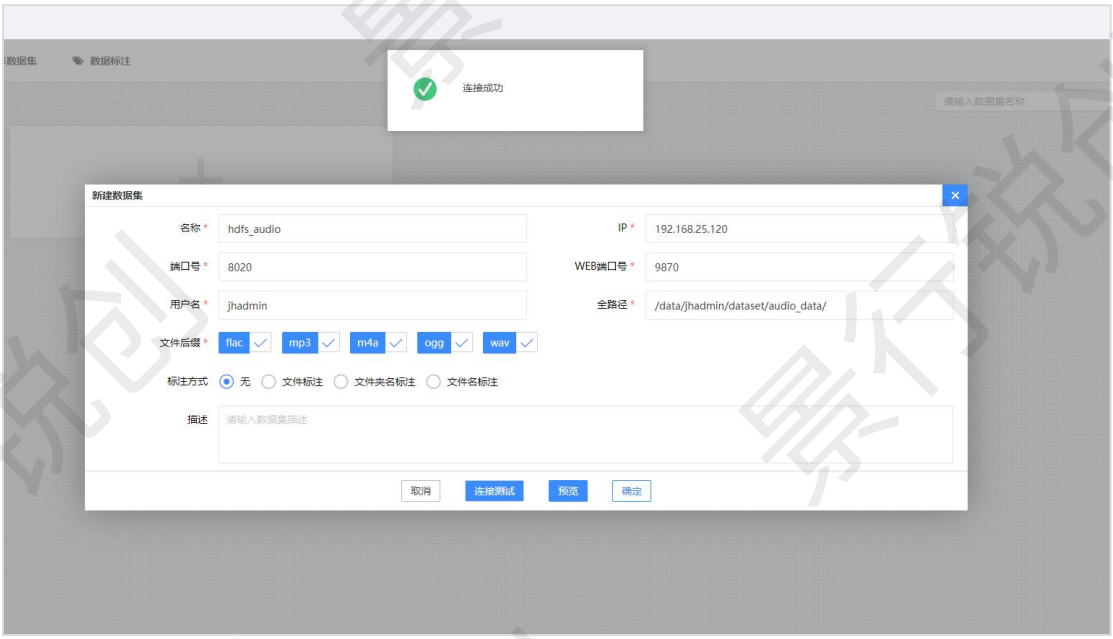
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 flac、mp3、m4a、ogg 和 wav，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

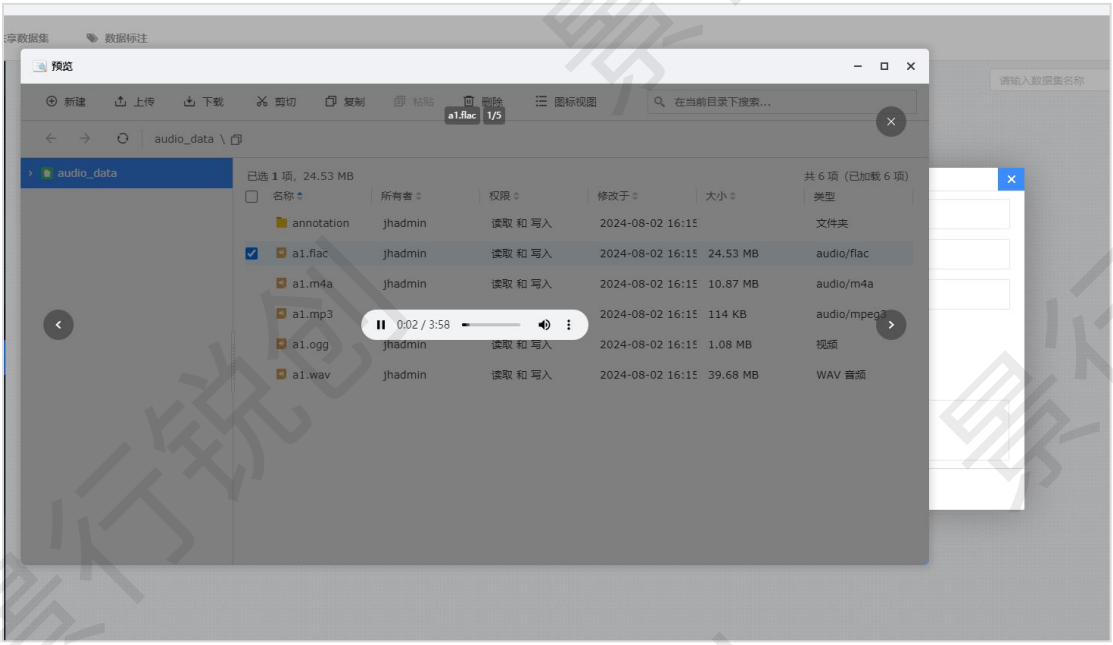
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



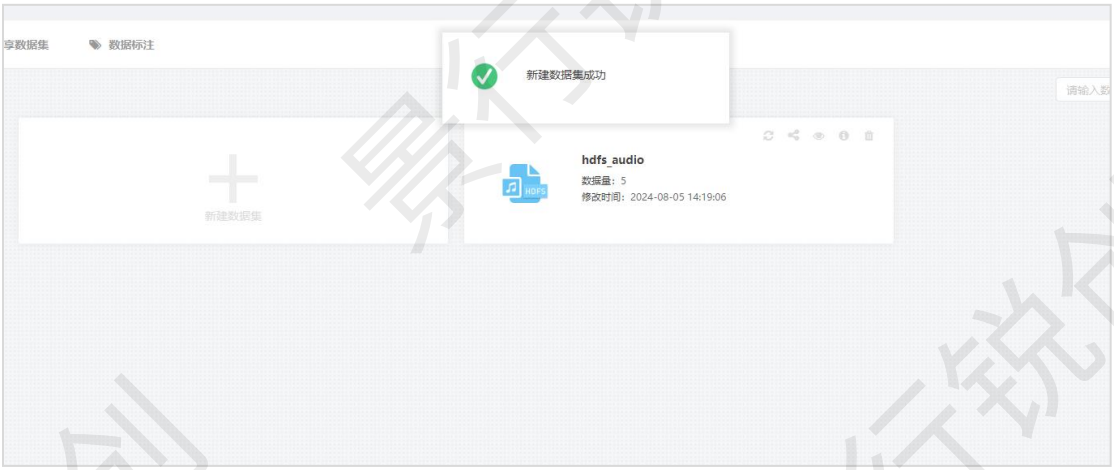
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“音频数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：

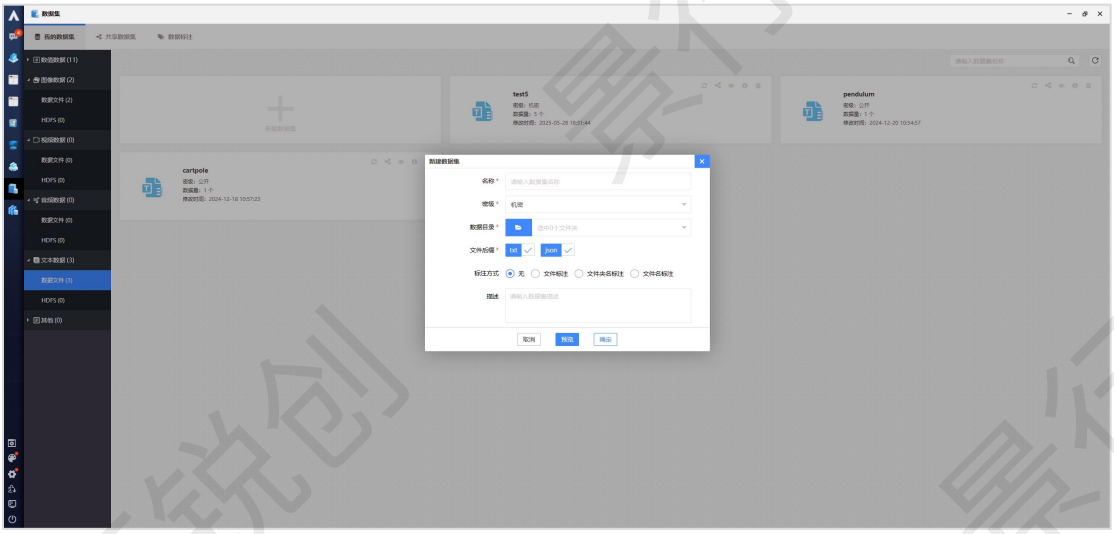


新建“音频数据-HDFS”数据集成功

2.4.5 文本数据

2.4.5.1 “文本数据-数据文件”数据集

新建“文本数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“文本数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“文本数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择文本文件夹即可。

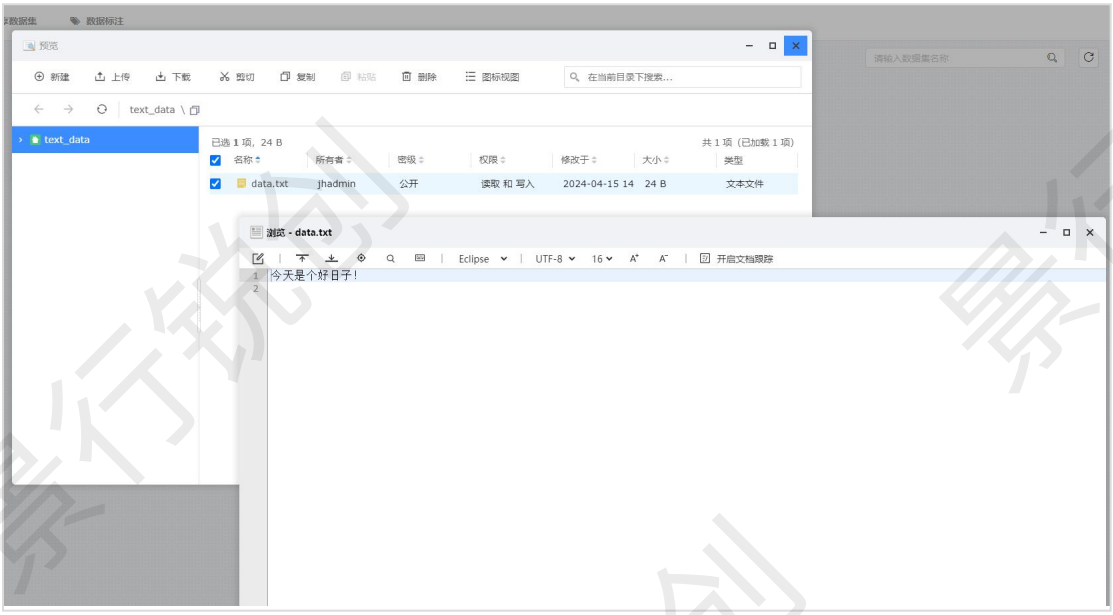
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 txt 和 json，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

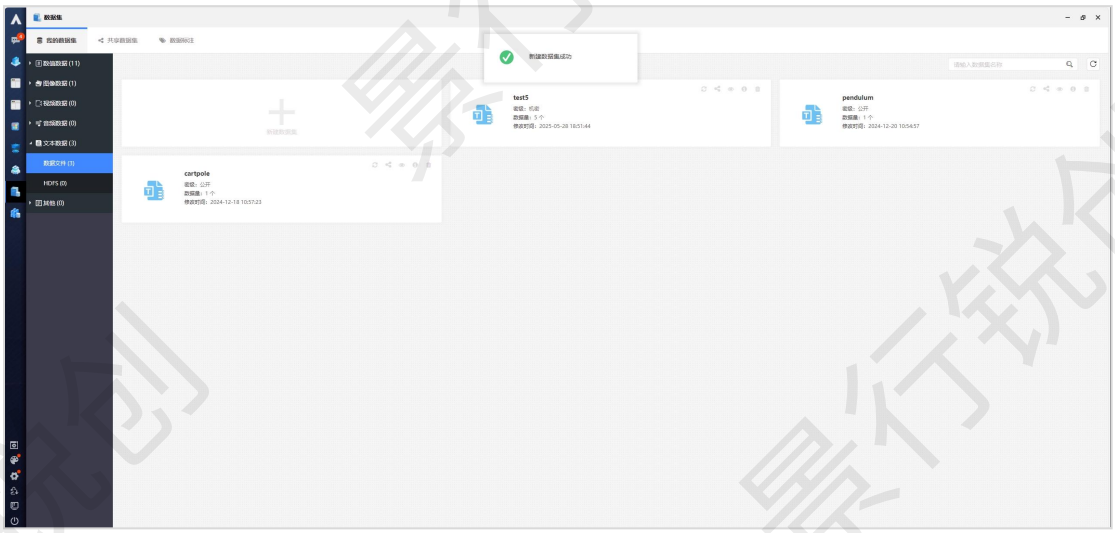
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“文本数据-数据文件”数据集

点击“确定”按钮，新建数据集。如下图所示：



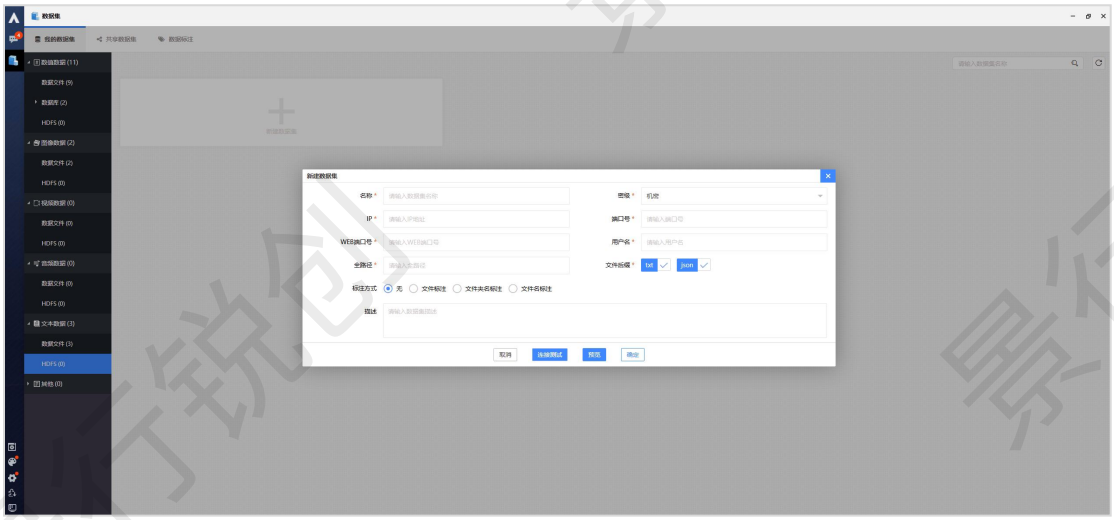
新建“文本数据-数据文件”数据集成功

2.4.5.2 “文本数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“文本数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-

“文本数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“文本数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

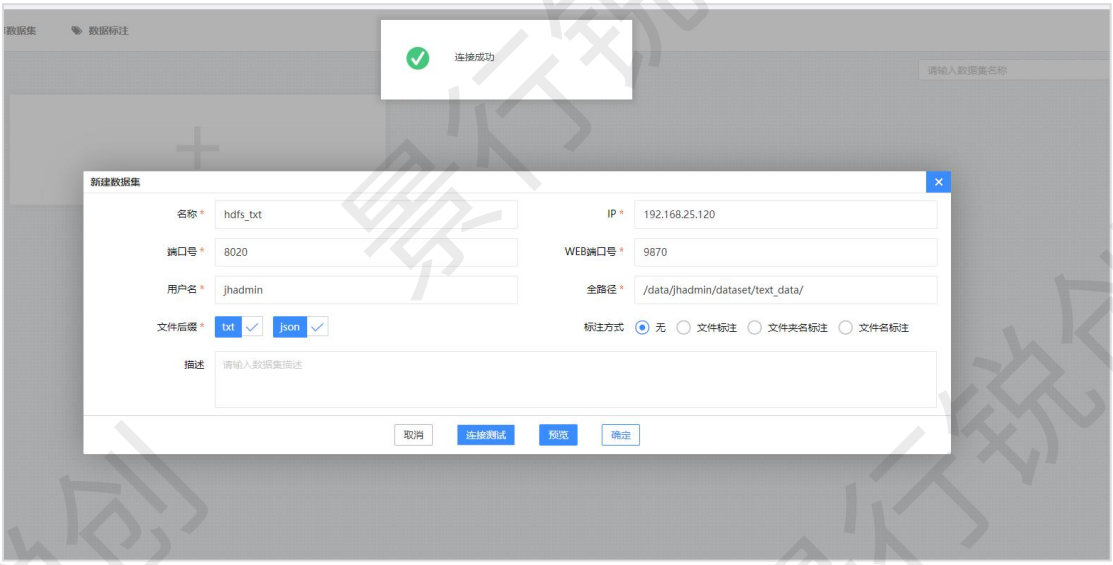
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 txt 和 json，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

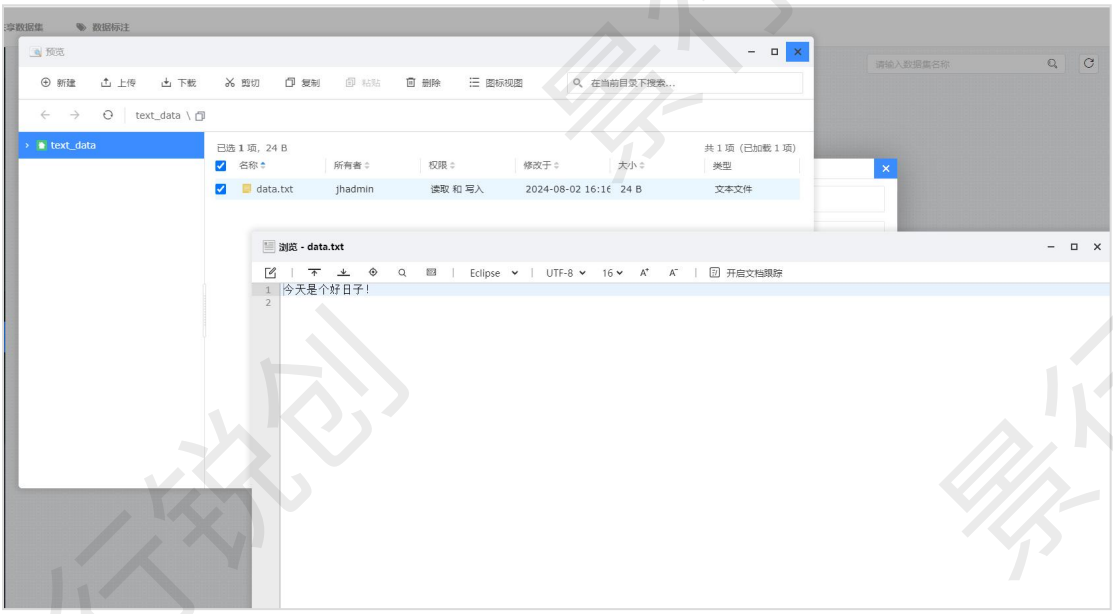
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



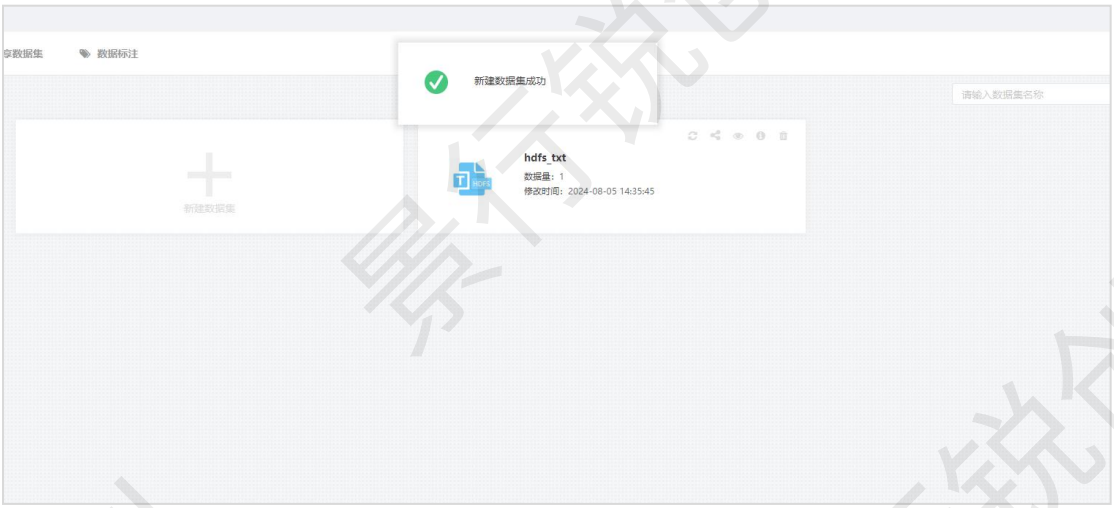
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“文本数据-HDFS”数据集

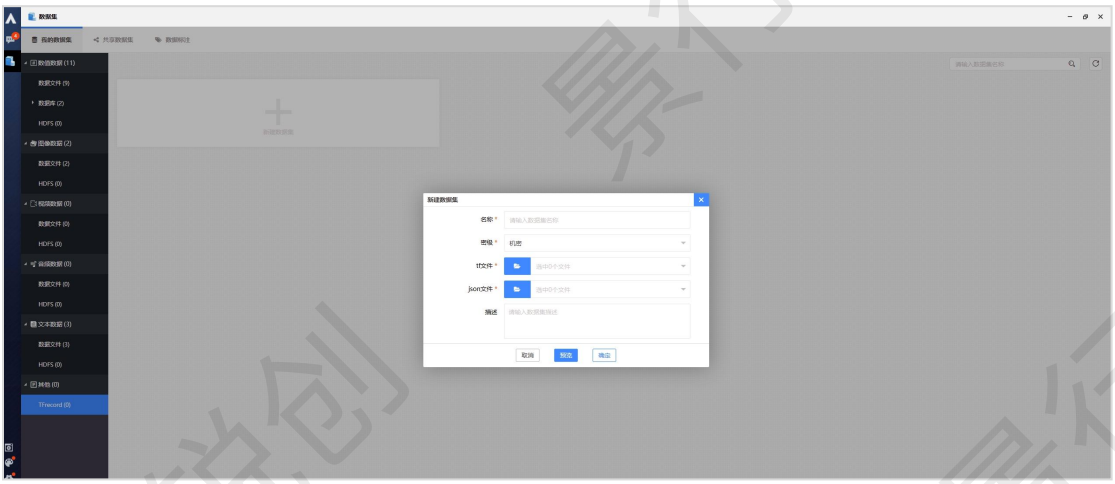
点击“确定”按钮，新建数据集。如下图所示：



新建“文本数据-HDFS”数据集成功

2.4.6 其他

新建“其他-TFrecord”数据集，操作步骤如下：依次点击“我的数据集”-“其他”-“TFrecord”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“其他-TFrecord”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

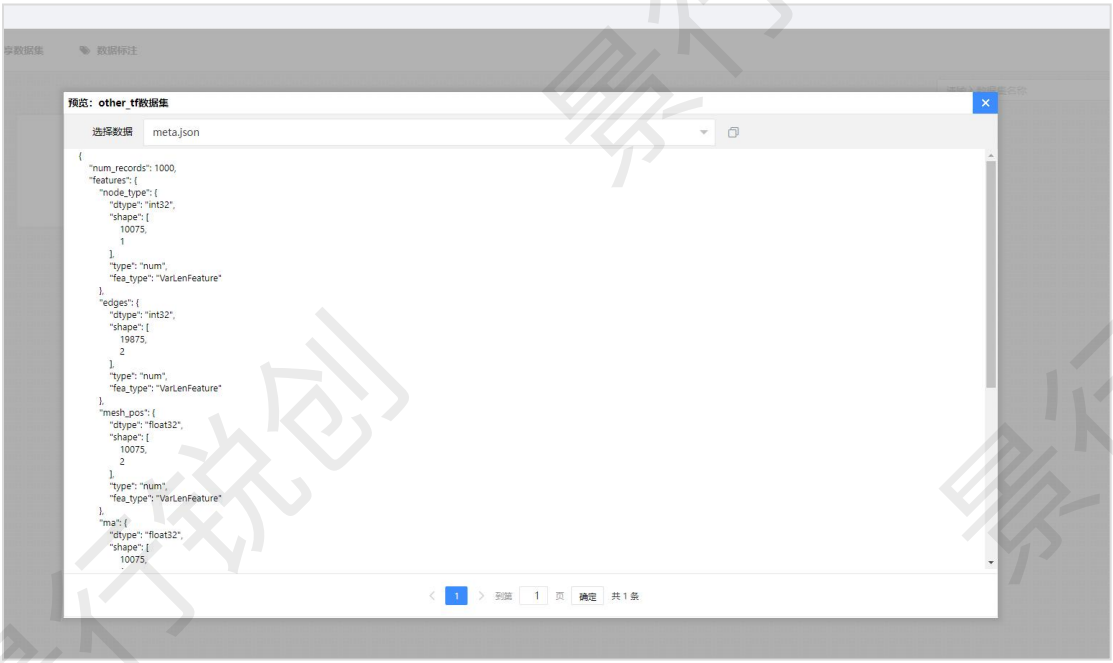
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

tf 文件：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择 tf 文件”窗口中，选择 tfrecord 文件即可。只允许选择一个 tfrecord 文件。

json 文件：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择 json 文件”窗口中，选择 json 文件即可。只允许选择一个 json 文件。

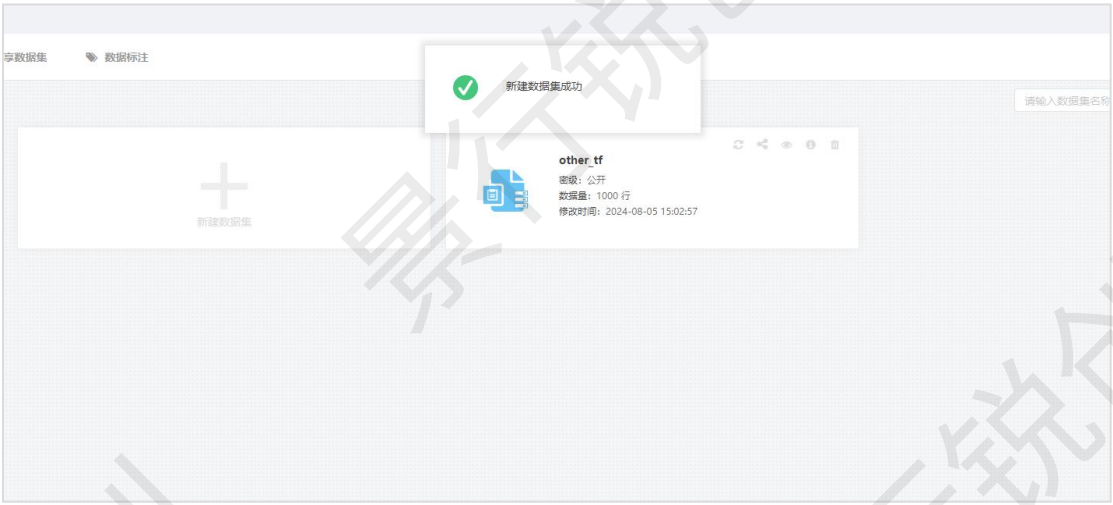
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。如下图所示：



预览“其他-TFrecord”数据集

点击“确定”按钮，新建数据集。如下图所示：



新建“其他-TFrecord”数据集成功

2.4.7 共享数据集

在我的数据集中，可以将我的数据集共享给其他用户使用，共享数据集基于共享数据区功能实现，共享数据集的操作权限与共享数据区保持一致各种角色对共享数据集的操作权限，如下所示：

共享者：对共享数据集有修改、预览、使用和删除的权限；

共享组用户：对共享数据集有预览、使用的权限；

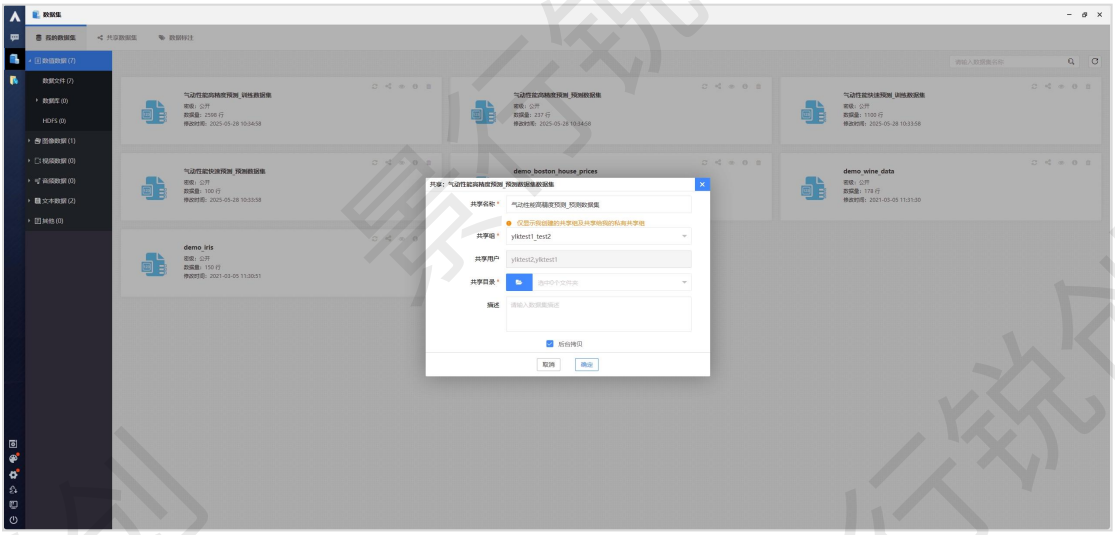
管理员：对共享数据集仅有删除权限。

2.4.7.1 将我的数据集共享至共享数据集

共享“数据文件”类型的数据集，如下所示：

- “数值数据-数据文件”数据集
- “图像数据-数据文件”数据集
- “视频数据-数据文件”数据集
- “音频数据-数据文件”数据集
- “文本数据-数据文件”数据集
- “其他-TFrecord”数据集

以共享“数值数据-数据文件”数据集为例，步骤如下：依次点击“我的数据集”-“数值数据”-“数据文件”分类按钮，然后在右侧的工作区，点击数据集卡片上的“共享”图标按钮，此时会弹出“共享”窗口，如下图所示：



共享“数值数据-数据文件”数据集

图中每个参数的具体含义如下：

共享名称：共享后的数据集名称，输入任意符合命名规则的名称即可。

共享组：选择可见的共享组，选项从我的数据→共享数据区获取。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

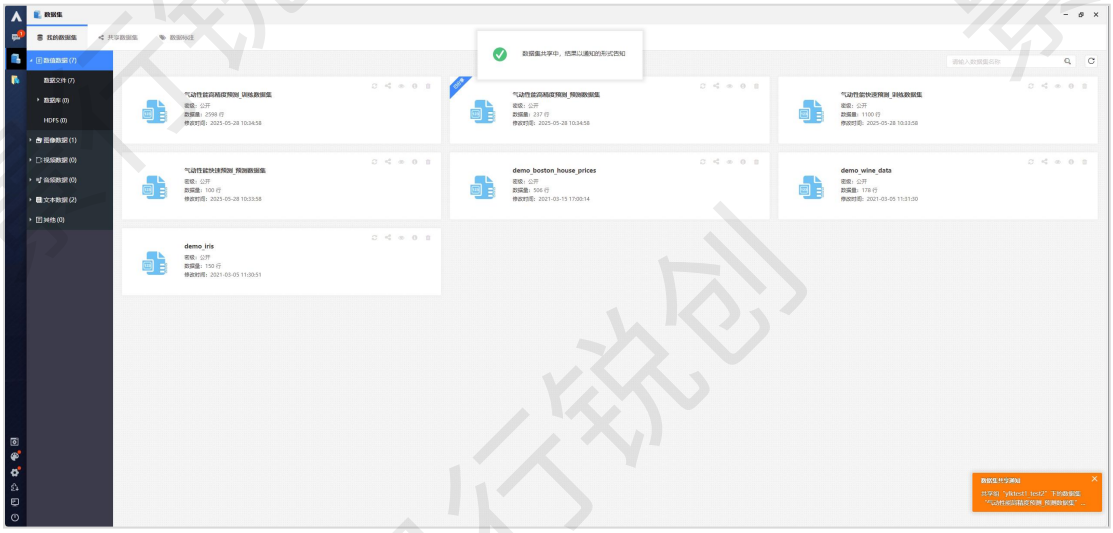
共享目录：选择共享组之后显示，只能在所选共享组的共享数据区选择文件

夹，点击蓝色的“文件夹”图标按钮，然后在弹出的“选择共享目录”窗口中选择文件夹即可，用作存放源数据文件目录，只允许选择一个文件夹。

描述：共享后的数据集描述信息，可选。

后台拷贝：默认勾选，采用后台形式将数据集的数据文件拷贝至共享目录，当前共享页面退出；不勾选则采用前台拷贝方式，当前共享页面直至共享结束退出。

配置完成后，点击“确定”按钮，进行数据集共享，如下图所示：



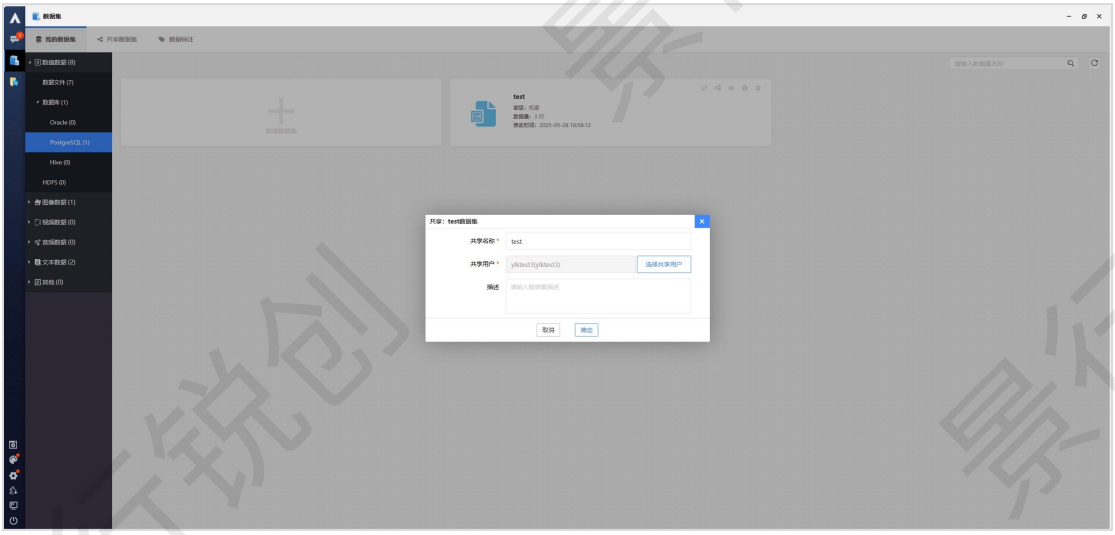
共享数据集

共享非“数据文件”类的数据集，如下所示：

- “数值数据-数据库”数据集
- “数值数据-HDFS”数据集
- “图像数据-HDFS”数据集
- “视频数据-HDFS”数据集
- “音频数据-HDFS”数据集
- “文本数据-HDFS”数据集

以共享“数值数据-数据库-PostgreSQL”数据集为例，操作步骤如下：依次点击“我的数据集”-“数值数据”-“数据库”-“PostgreSQL”分类按钮，然后在右侧的工作区，点击数据集卡片上的“共享”按钮，此时会弹出“共享”窗

口，如下图所示：



共享“数值数据-数据库-PostgreSQL”数据集

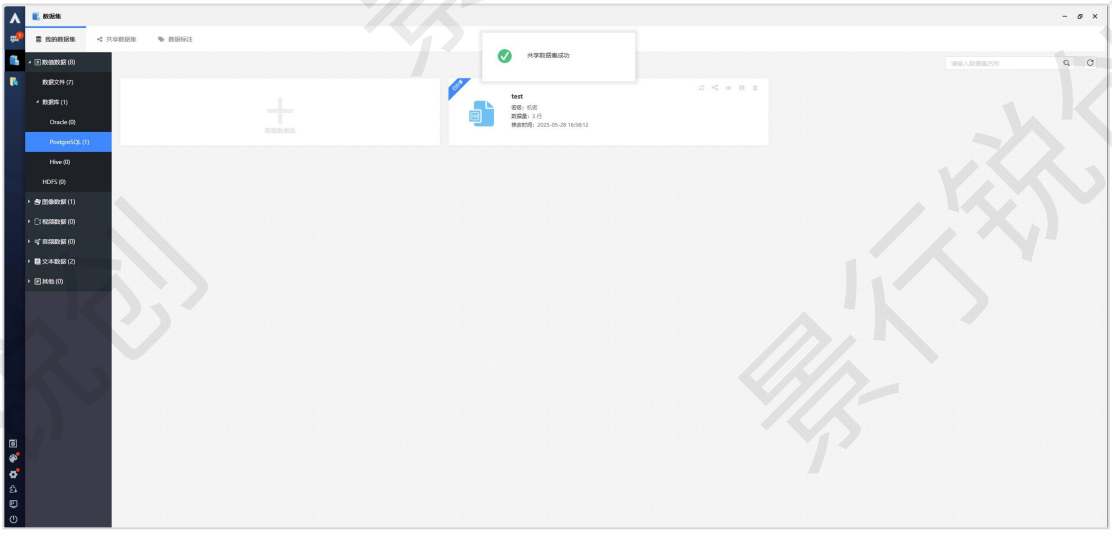
图中每个参数的具体含义如下：

共享名称：共享后的数据集名称，输入任意符合命名规则的名称即可。

共享用户：选择共享用户。

描述：共享后的数据集描述信息，可选。

配置完成后，点击“确定”按钮，共享数据集，如下图所示：



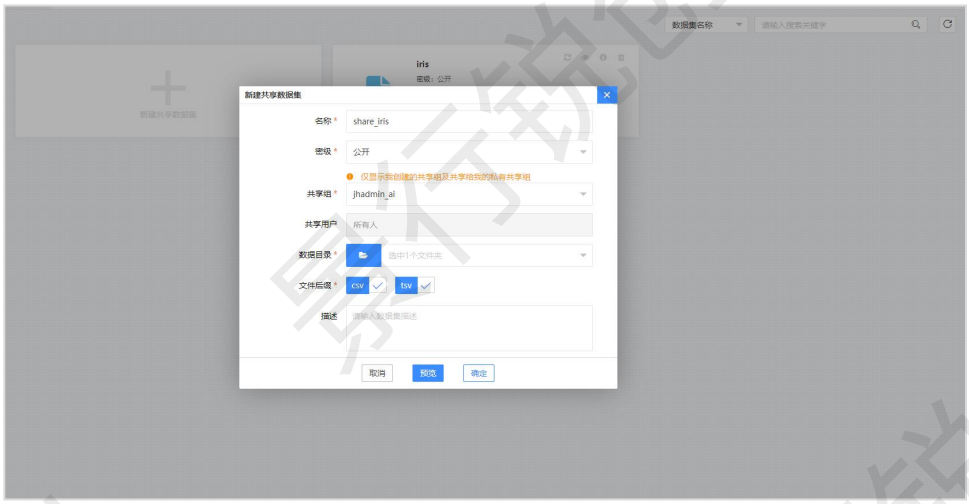
共享数据集

2.4.7.2 新建共享数据集

新建数据文件类的共享数据集，如下所示：

- “数值数据-数据文件”数据集
- “图像数据-数据文件”数据集
- “视频数据-数据文件”数据集
- “音频数据-数据文件”数据集
- “文本数据-数据文件”数据集
- “其他-TFrecord”数据集

以新建“数值数据-数据文件”共享数据集为例，操作步骤如下：依次点击“共享数据集”-“数值数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建共享数据集”卡片按钮，此时会弹出“新建共享数据集”窗口，如下图所示：



新建“数值数据-数据文件”共享数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

共享组：选择可见的共享组，选项从我的数据→共享数据区获取。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

数据目录：选择共享组之后显示，只能在所选共享组的共享数据区选择数据目录，同我的数据集中数值数据→数据文件。

描述：输入该数据集的描述信息，可选。

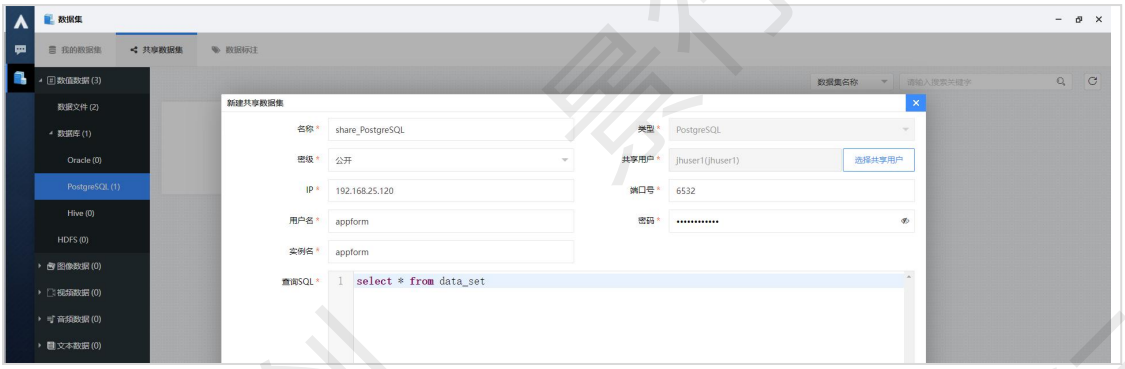
配置完成后，点击“确定”按钮，新建共享数据集。如下图所示：

新建共享数据集

共享非“数据文件”类的数据集，如下所示：

- “数值数据-数据库”数据集
- “数值数据-HDFS”数据集
- “图像数据-HDFS”数据集
- “视频数据-HDFS”数据集
- “音频数据-HDFS”数据集
- “文本数据-HDFS”数据集

以新建“数值数据-数据库-PostgreSQL”共享数据集为例，操作步骤如下：依次点击“共享数据集”-“数值数据”-“数据库”-“PostgreSQL”分类按钮，然后在右侧的工作区，点击“新建共享数据集”卡片按钮，此时会弹出“新建共享数据集”窗口，如下图所示：



新建“数值数据-数据库-PostgreSQL”共享数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

类型：选择数据库的类型。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

IP：输入数据库所在机器的 IP。

端口号：输入数据库使用的端口号。

用户名：输入数据库的用户名。

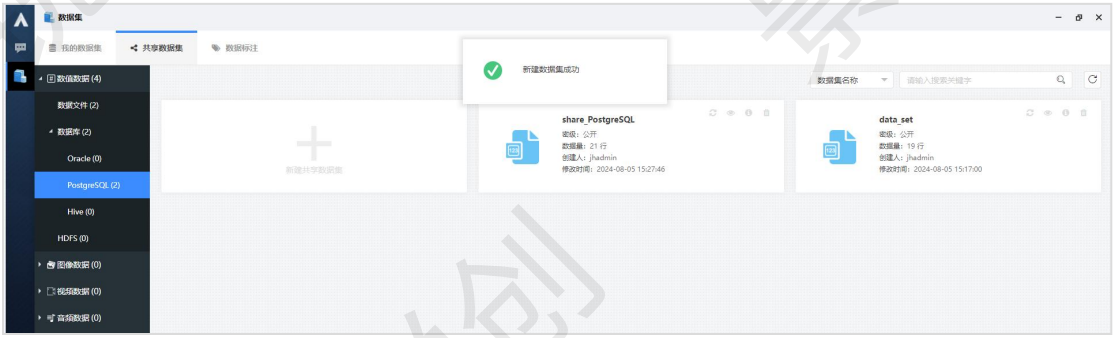
密码：输入数据库用户对应的密码。

实例名：输入数据库名。

查询 SQL：输入 SQL 查询语句（注意：不要在查询语句后面加分号，eg：select * from tbl）。

描述：输入该数据集的描述信息，可选。

配置完成后，点击“确定”按钮，新建共享数据集。如下图所示：



新建共享数据集

2.4.8 数据标注

数据标注功能为用户提供了灵活、可扩展的在线数据标注工具。以下是数据标注的主要功能：

多数据类型支持：支持多种数据类型的标注，包括图像、文本、音频、视频等，使其适用于各种人工智能任务。

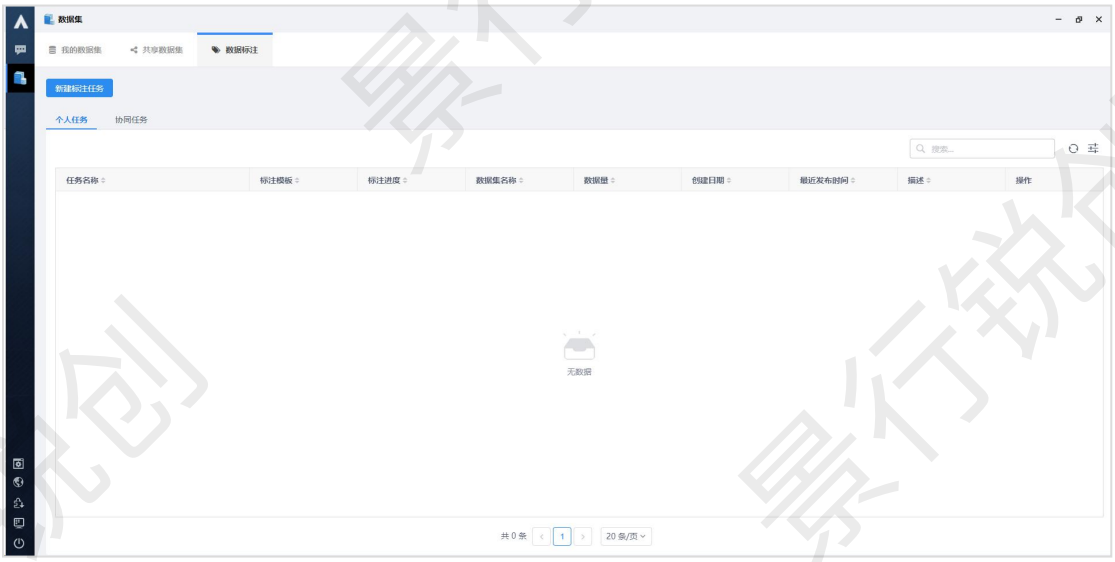
灵活的标注工具：提供丰富的标注工具，包括矩形框、多边形、点标注、文本标注等，以满足不同任务的标注需求。

可视化界面：具备直观的可视化界面，使标注过程更加用户友好，同时支持标注者在标注过程中实时预览标注效果。

自定义标签：用户可以定义和配置标签集，以适应不同项目的特定标注任务，实现个性化的标注需求。

协作与团队功能：提供协作功能，支持多人团队在同一数据集上进行标注工作。

标注格式导出：支持将标注后的数据以各种标准格式导出。



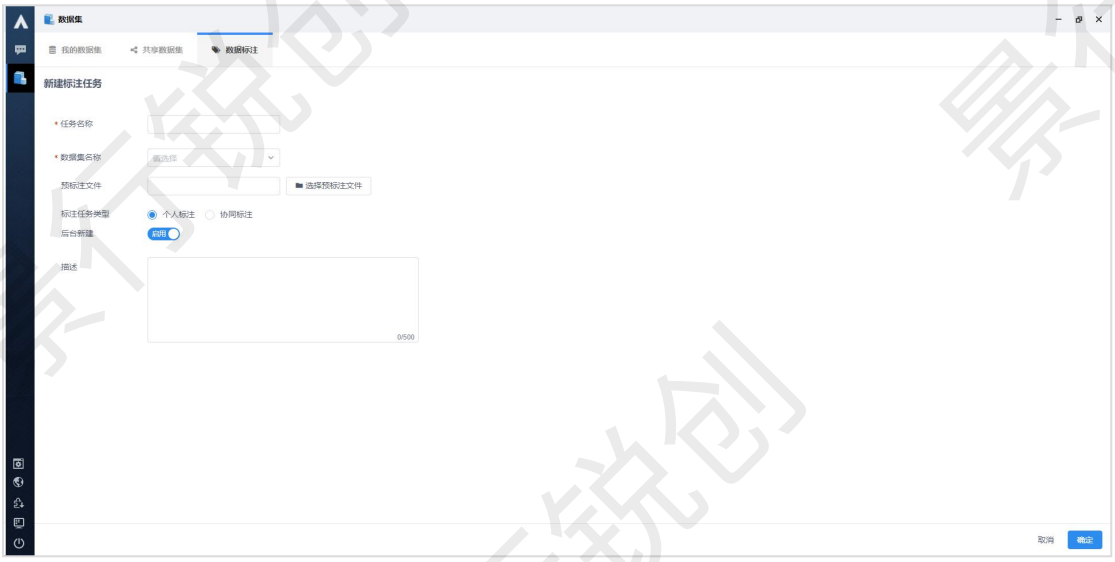
数据标注

2.4.8.1 标注任务管理

2.4.8.1.1 新建

您可以点击数据标注页面顶部的“新建标注任务”按钮，进入新建标注任务的页面。根据实际需要完成此页面的表单后，点击“确定”按钮即可完成标注任务的创建。

注意：“数值数据”和“其他”类型的数据集不支持标注。



新建标注任务

图中每个参数的具体含义如下：

任务名称：标注任务的名称，输入任意符合命名规则的名称即可。

数据集名称：选择已有数据集。

预标注文件：选择该数据集数据的标注文件，主要用于对已标注的标注数据，可持续标注。

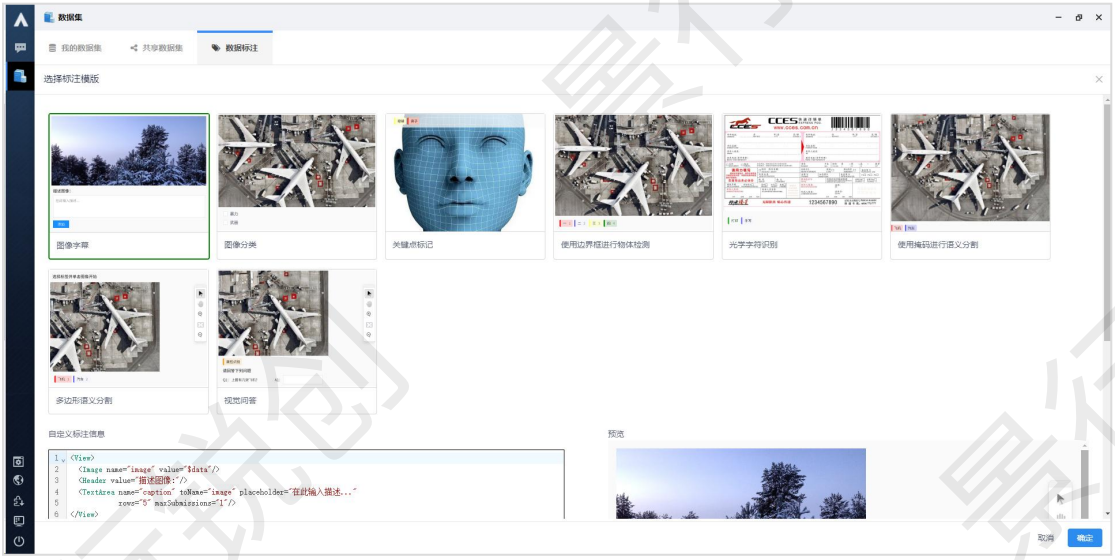
标注模板：根据当前选择数据集类型选择不同场景下的标注模板。

标注任务类型：可根据需要选择“个人标注”或“协同标注”。

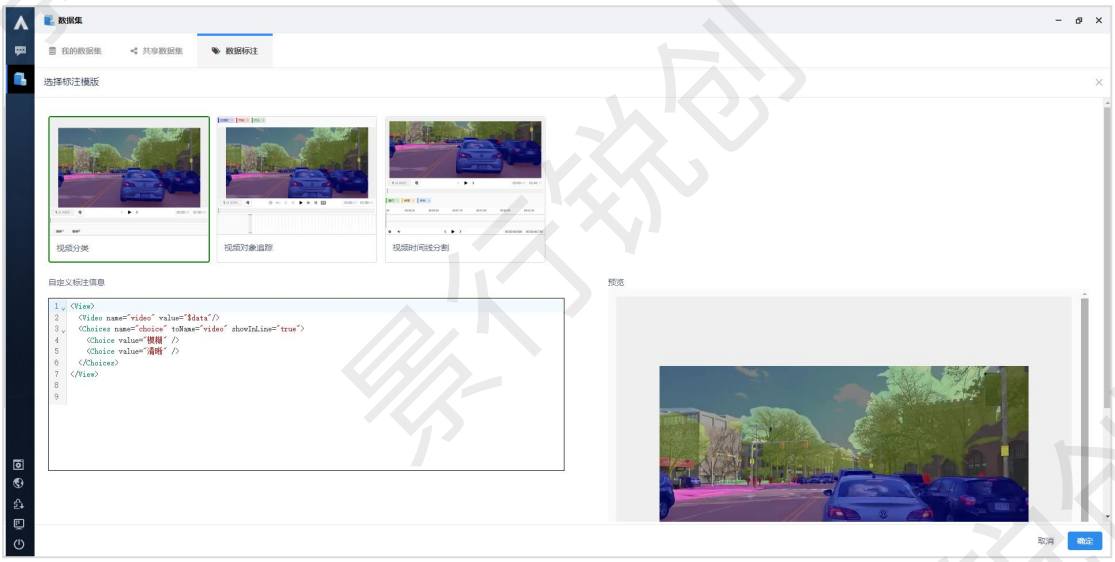
后台新建：默认开启，采用后台形式新建标注任务；新建成功后，通过右下角的气泡信息通知用户；关闭后则采用前台形式拷贝方式。

描述：输入该标注任务的描述信息，可选。

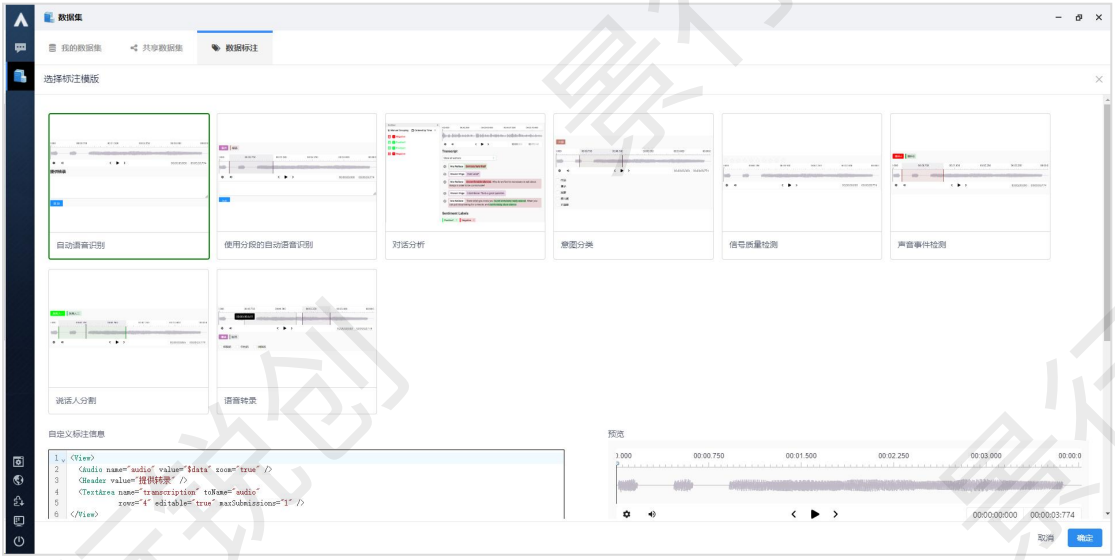
标注模板页面如下所示：



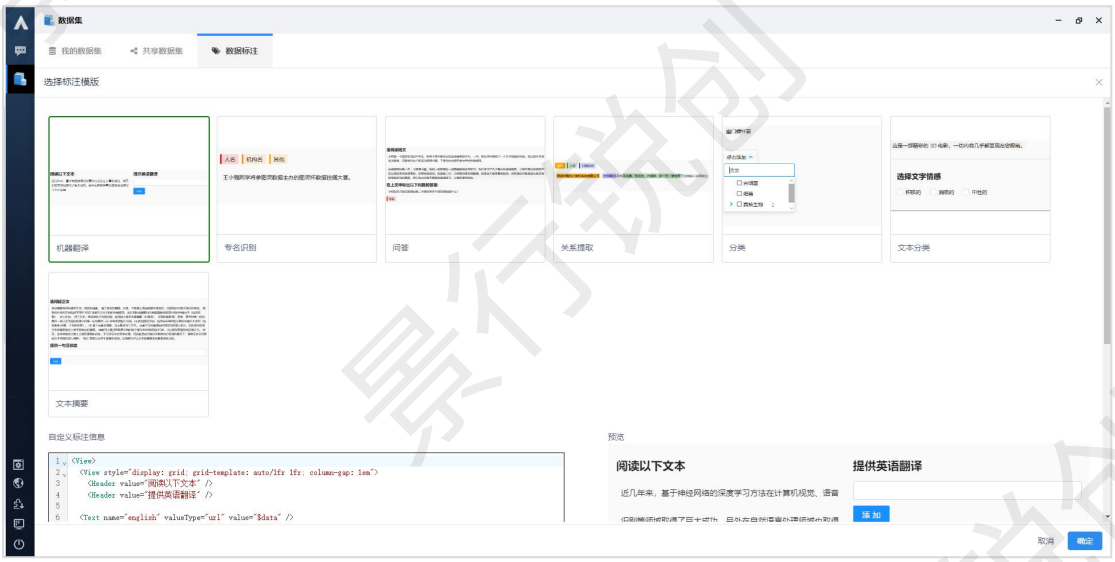
图像标注模板



视频标注模板



音频标注模板



文本标注模板

2.4.8.1.2 修改、删除

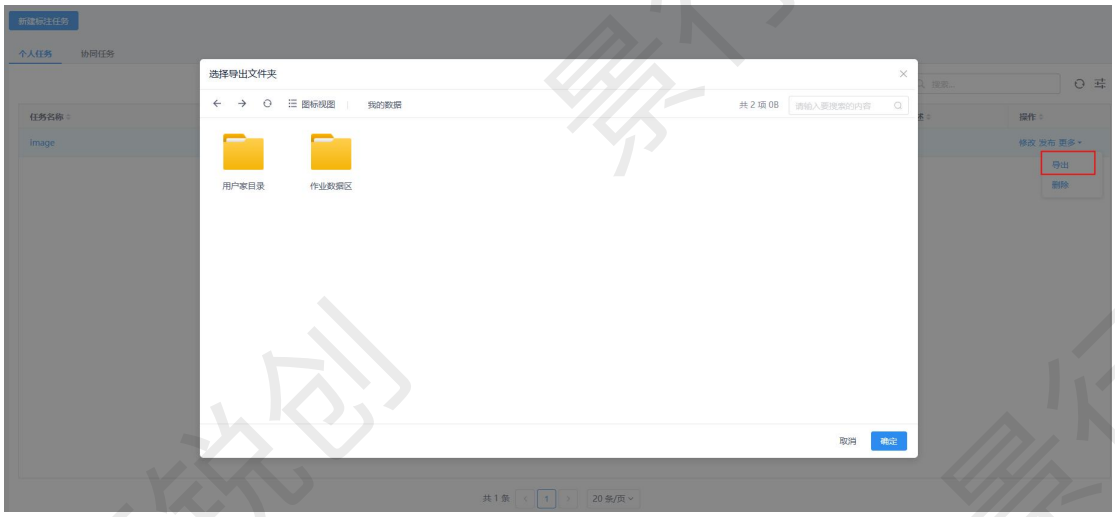
您可以对标注任务执行以下操作：修改、发布、导出和删除。其中，修改操作仅限于修改预标注文件、标注模板以及描述信息。导出操作是将标注数据导出至指定目录。发布操作是将标注数据发布至与标注数据同级的目录。最后，删除操作将彻底删除标注任务。

标注任务的操作

2.4.8.1.3 发布、导出

对创建完成的标注任务进行发布和导出。

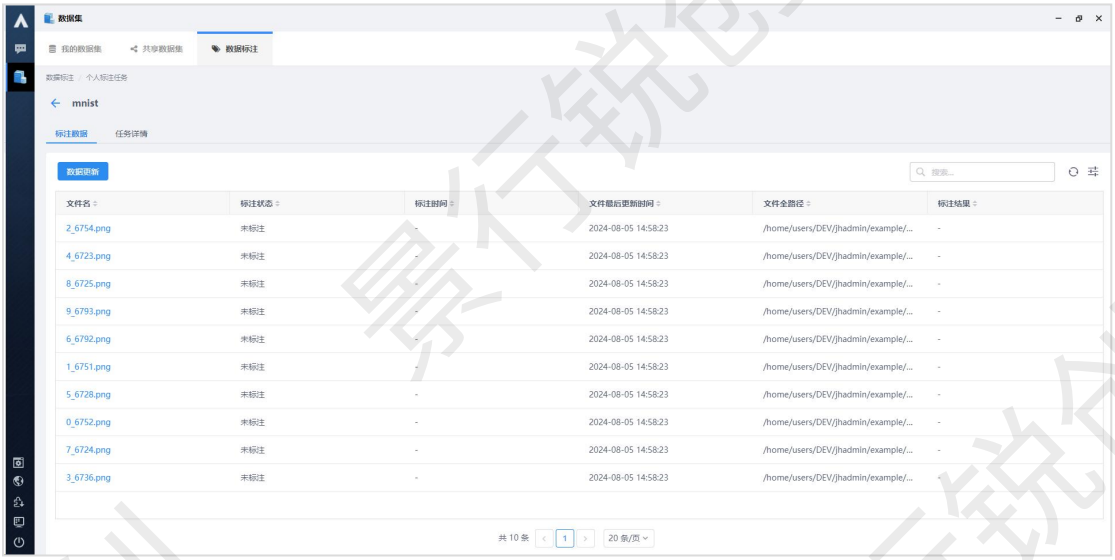
发布标注任务



导出标注任务

2.4.8.1.4 任务数据

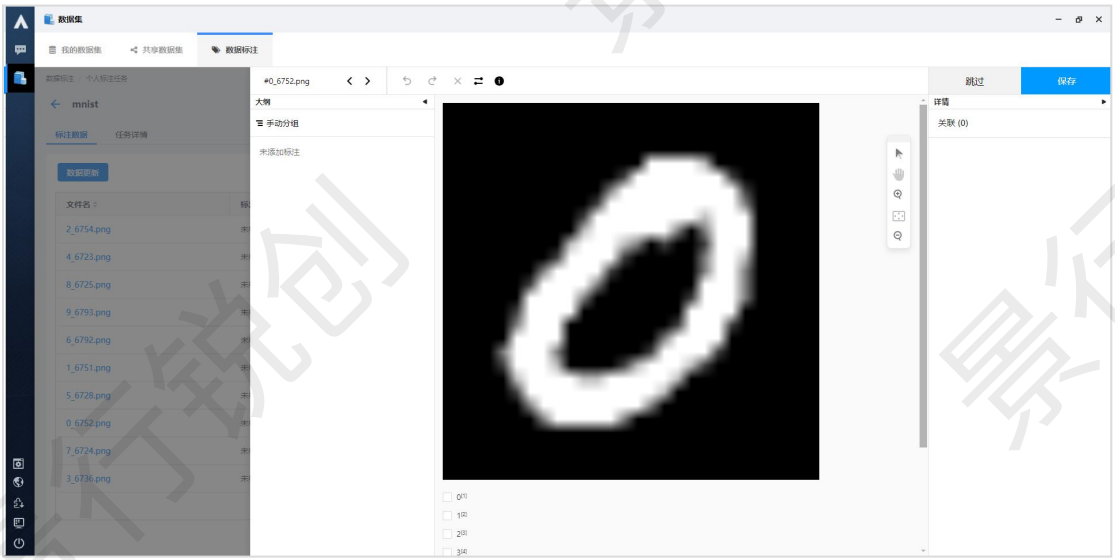
通过点击任务列表中任务名称，可以进入任务数据列表。



任务数据列表

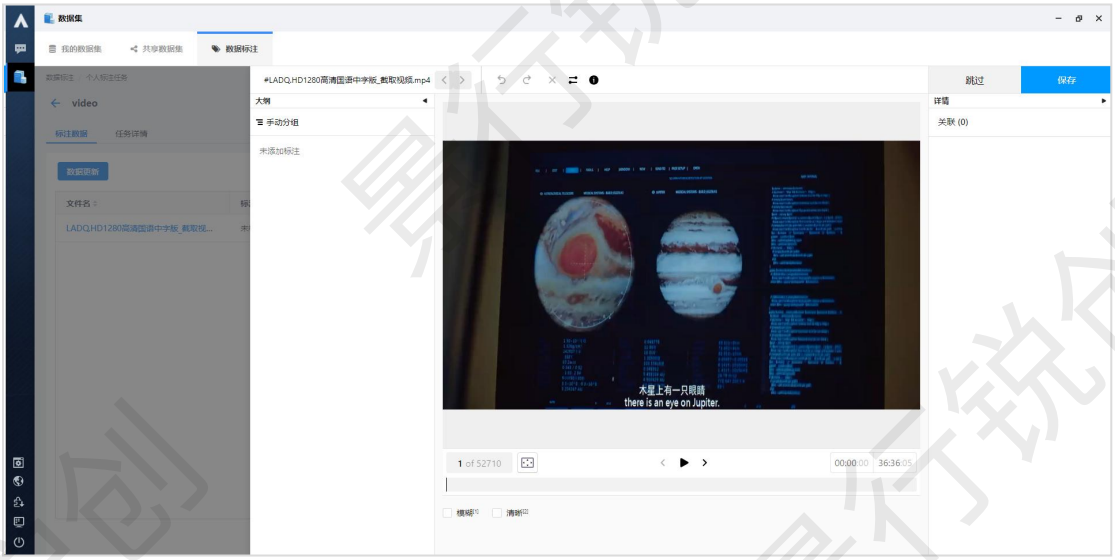
2.4.8.2 标注数据类型

2.4.8.2.1 图像标注



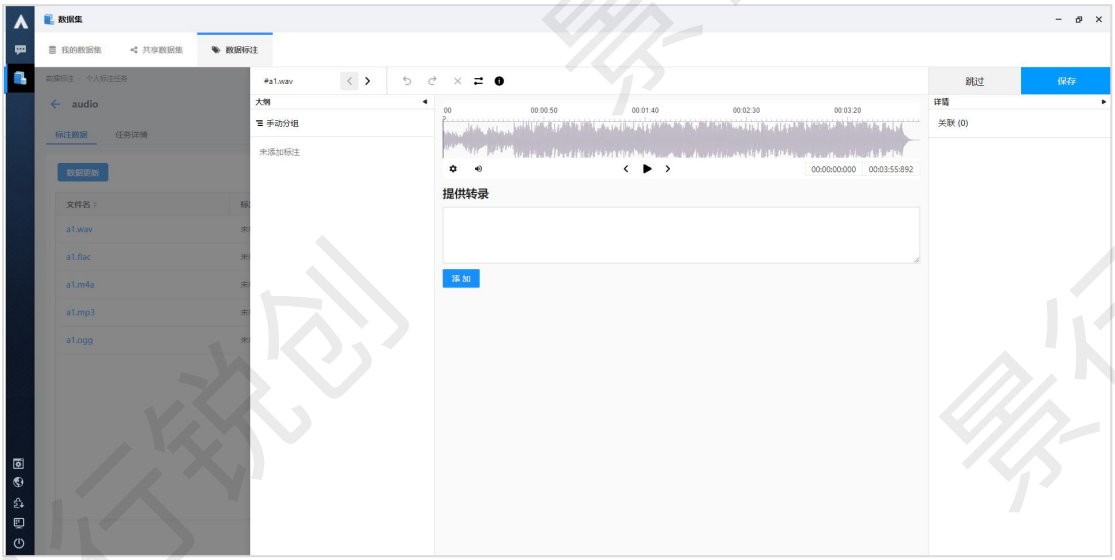
图像标注

2.4.8.2.2 视频标注



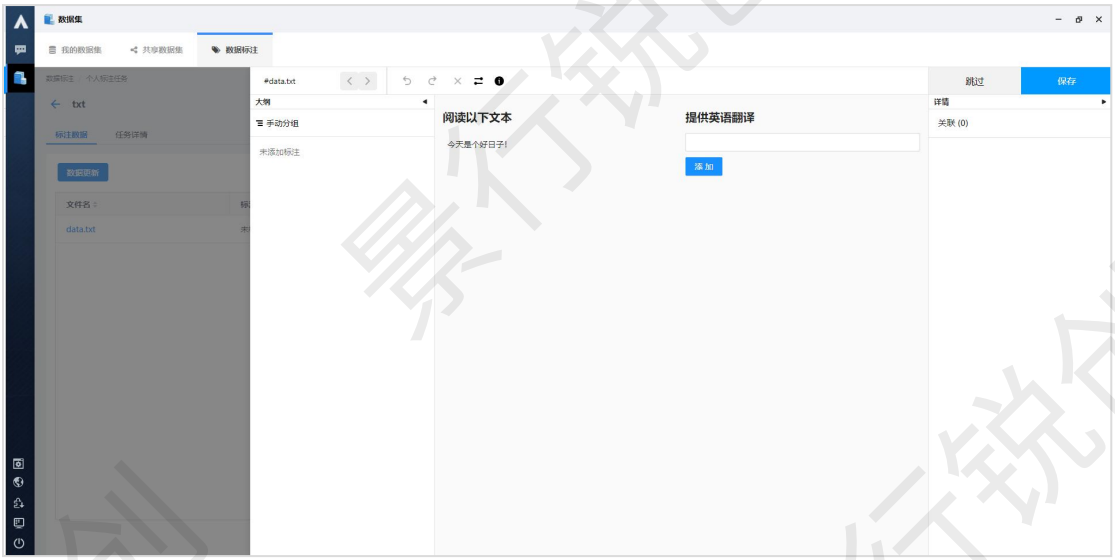
视频标注

2.4.8.2.3 音频标注



音频标注

2.4.8.2.4 文本标注



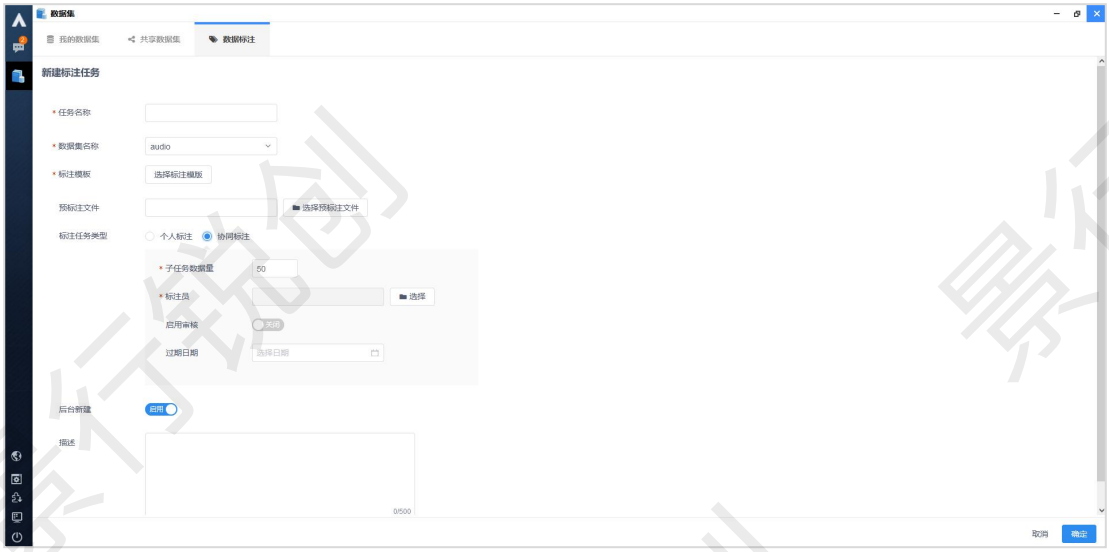
文本标注

2.4.8.3 协同标注

对于一些大的数据集一个人无法完成所有标注，需要寻找其他人协助自己完成整个数据集的标注工作，此时需要创建协同标注任务，同时也可以设置质量审核人员来保证标注结果的质量。

2.4.8.3.1 新建

您可以点击数据标注页面顶部的“新建标注任务”按钮，进入新建标注任务的页面。切换标注任务类型为“协同标注”。



创建协同标注任务

协同标注参数具体含义如下：

子任务数据量：协同任务会按照设置的子任务包数量进行拆分数据集，每个子任务在标注过程中独立操作。

标注员：标注任务可以派发的人员。

启用审核：是否开启标注审核这一流程，开启后标注员提交任务到审核人员，审核人员对标注结果进行检查。

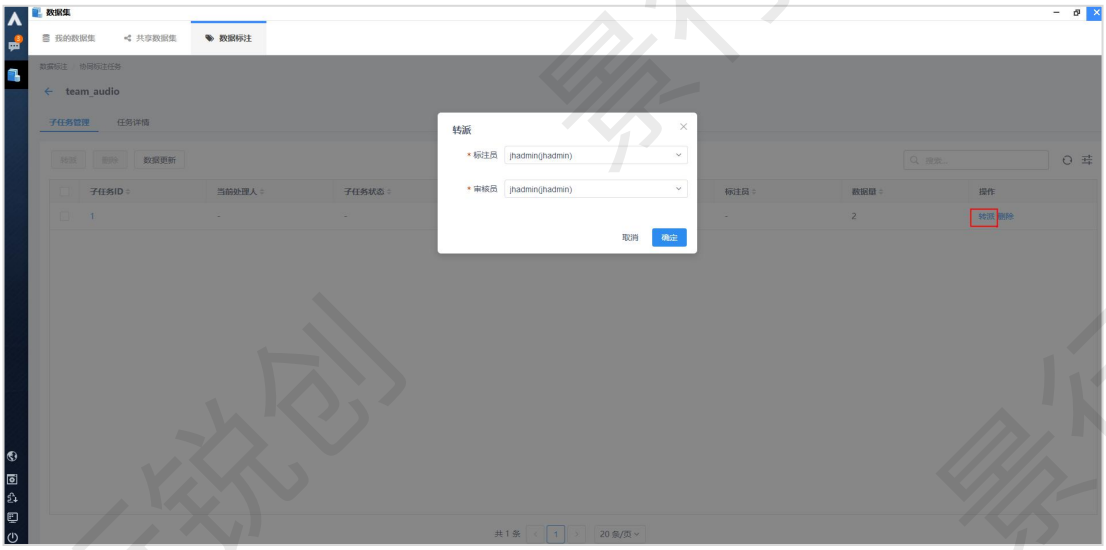
审核员：对标注任务进行审核的人员。

审核策略：支持两种形式，按百分比和按数量，标注员提交子任务后按此规则抽取待检查的数据。

过期日期：整个标注任务的完成期限设置。

2.4.8.3.2 转派任务

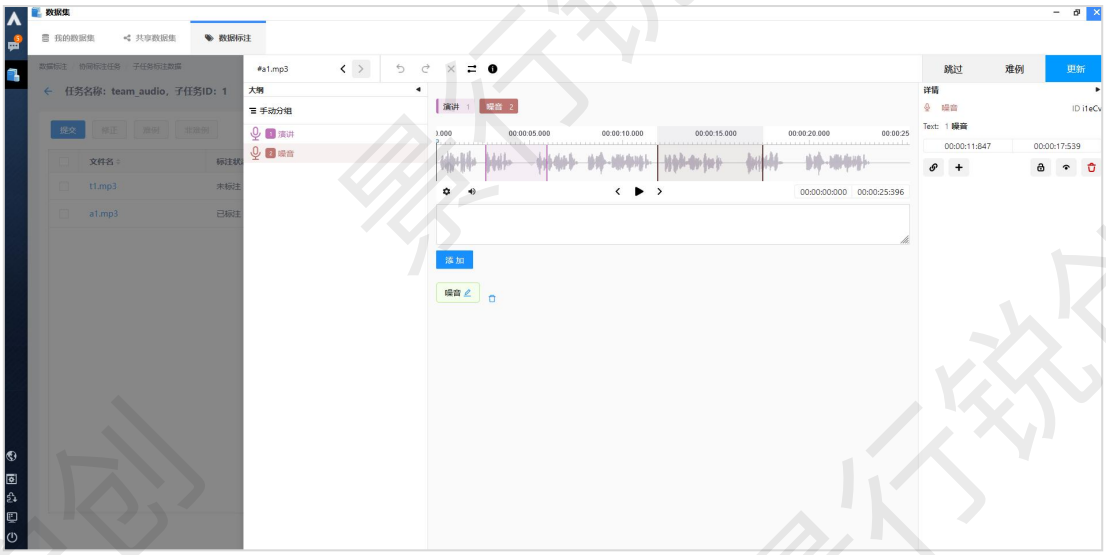
您可以对创建完成的协同标注任务进行转派操作，即将拆分后的若干个子任务分配到具体的标注人员。通过点击任务列表中任务名称，进入子任务列表，点击子任务的转派按钮即可分配到具体标注人员。



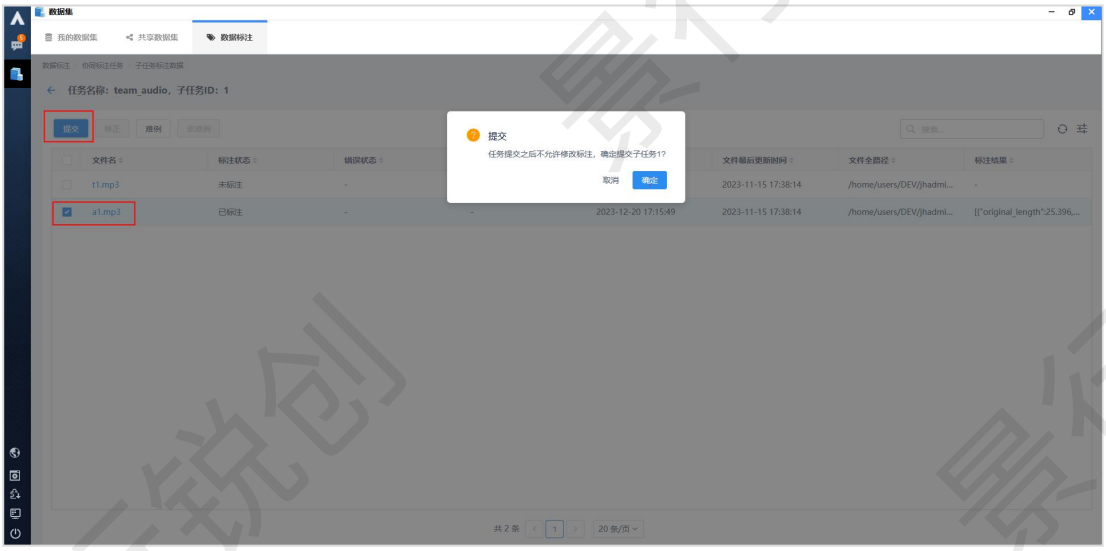
转派任务

2.4.8.3.3 提交任务

当标注子任务被转派给具体的标注人员，标注员就可以开始标注和提交标注任务。



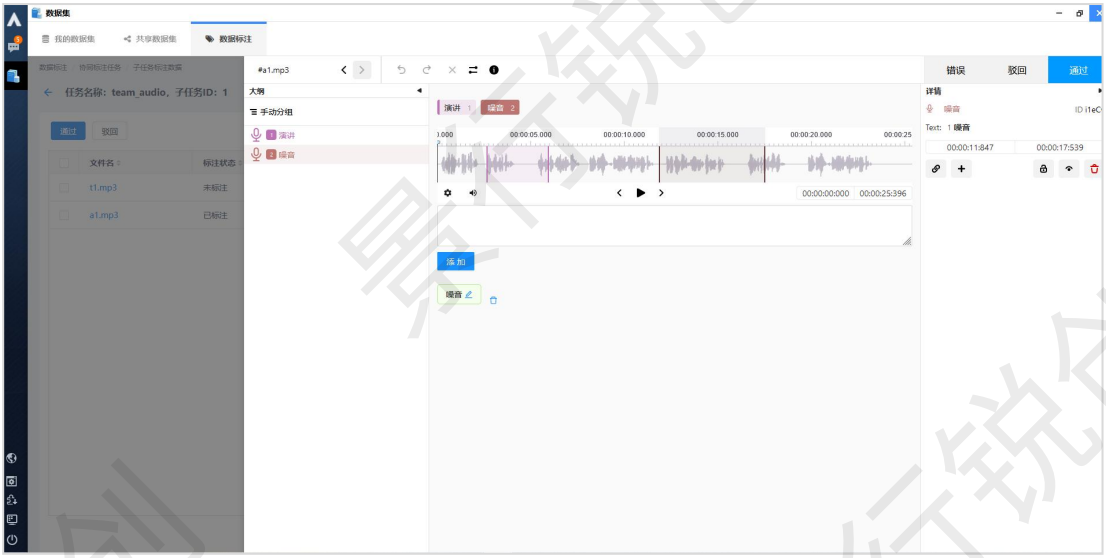
数据标注



提交标注任务

2.4.8.3.4 审核任务

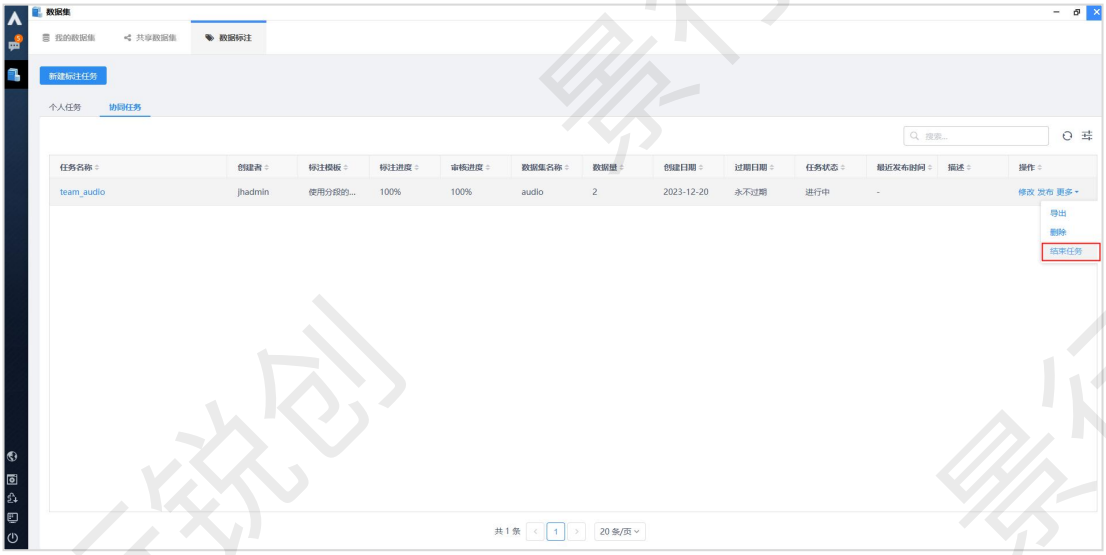
标注员提交任务后审核员进行审查。有错误、驳回、通过三种审核结果。



审核标注任务

2.4.8.3.5 结束任务

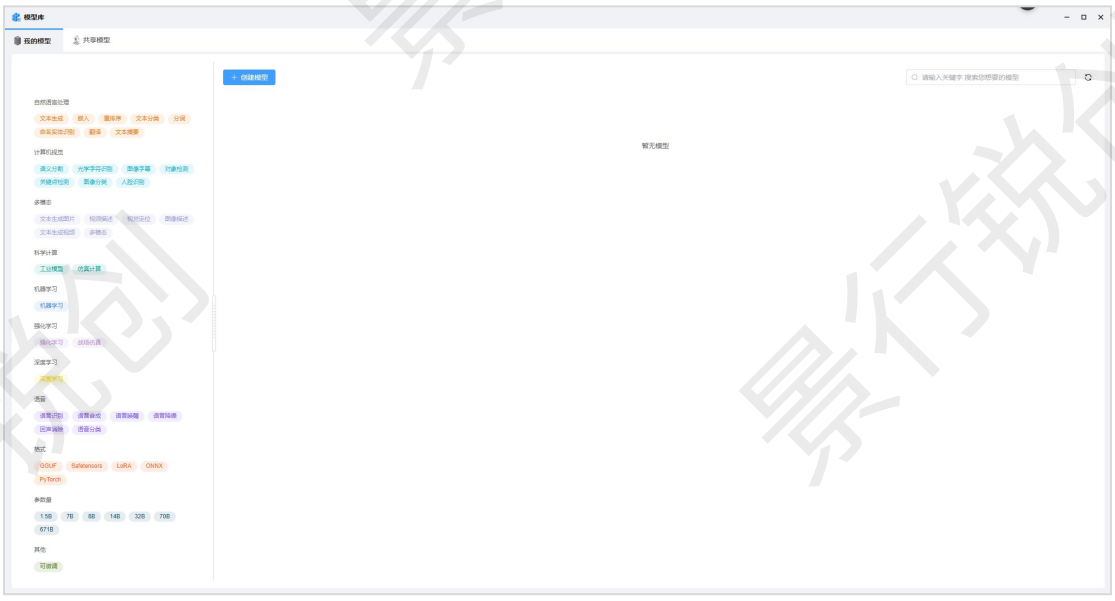
标注任务创建者可以在任意阶段对标注任务进行结束操作，结束操作后标注员和审核员将不能对标注任务进行任何修改。



结束标注任务

2.5 模型库

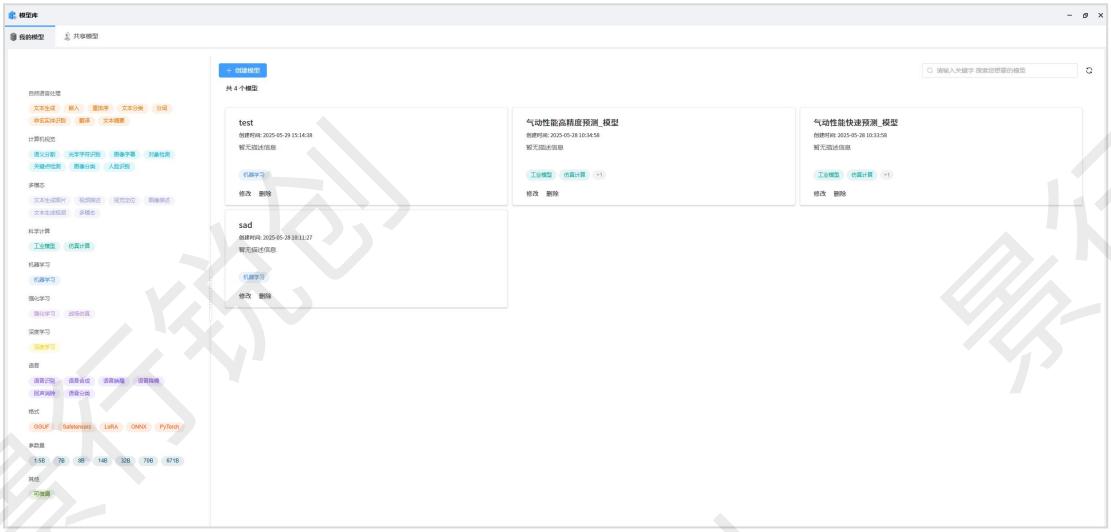
模型库，包含了平台中归档产生和创建模型产生的所有模型。通过模型库模块，可以实现模型版本管理、模型部署、模型转换和模型共享。模型库中包括两个页签：我的模型、共享模型。“我的模型”用于存放用户自己创建的模型，“共享模型”用于存放内置模型、其他用户共享的模型以及用户自己共享的模型。界面如下图所示：



模型库

2.5.1 我的模型

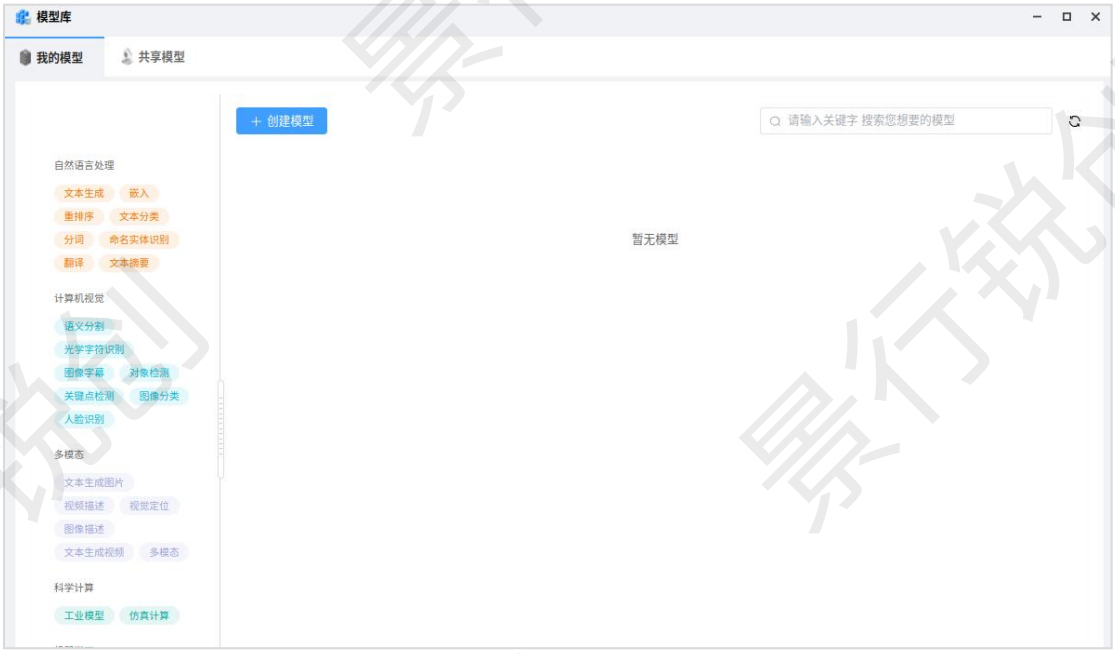
在“模型库”中点击“我的模型”按钮，可以展示我的模型页面，界面如下图所示：



我的模型

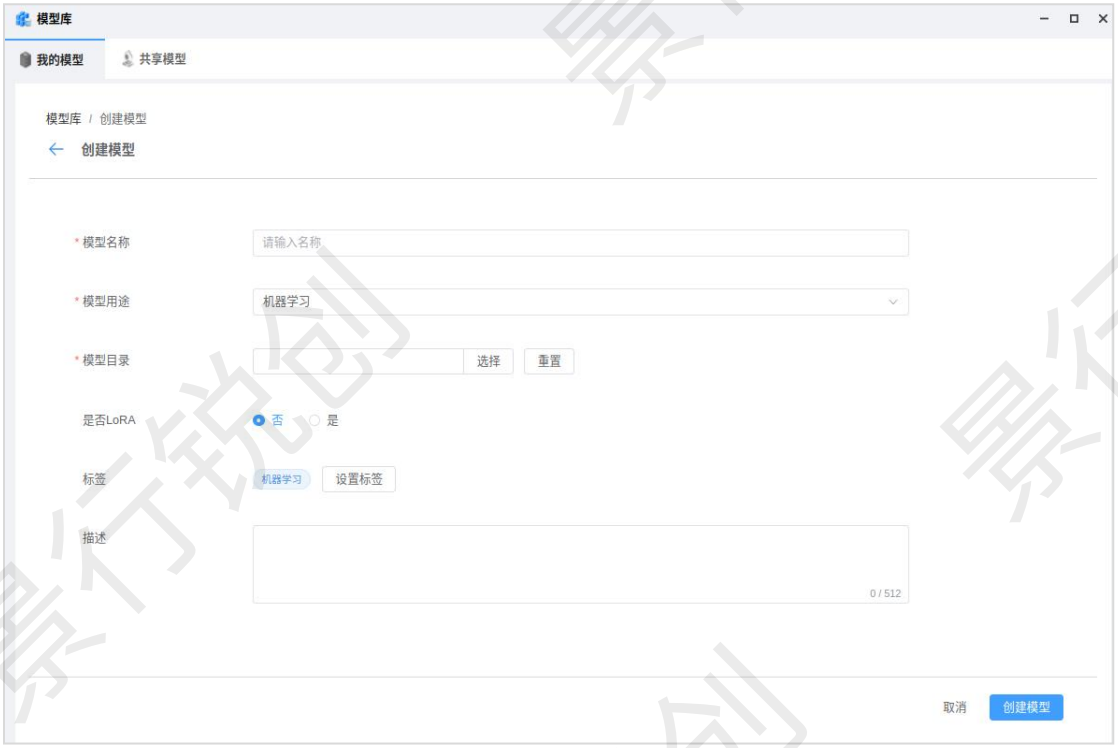
2.5.2 创建模型

在我的模型左侧展示了所有可以设置的标签，右侧为我的模型列表区域。点击“创建模型”按钮，开始创建模型。如下图所示：



在模型库创建模型

点击“创建模型”后，跳转到“创建模型”页面，如下图所示：



新建模型页面

创建模型页面所有参数解释如下：

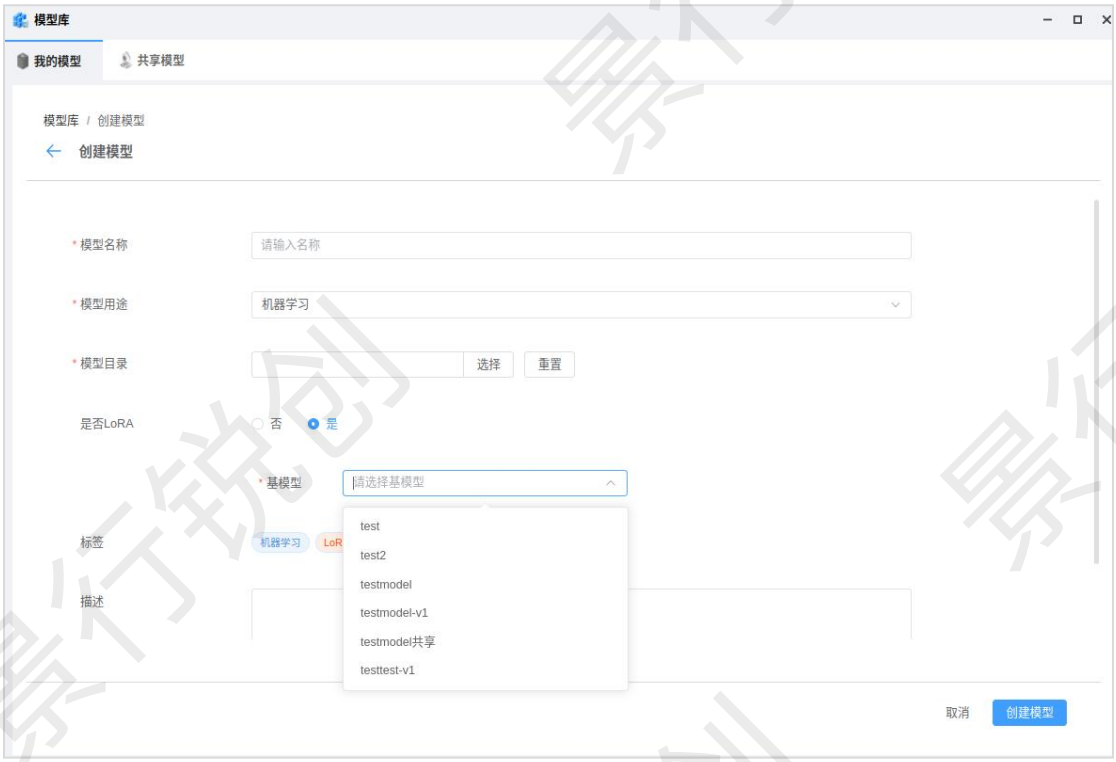
模型名称：用户自定义，但是不能与现有模型名称一致，必填项。

模型用途：选择模型用途用于为部署模型推荐合适的镜像。模型用途可选有：文本生成、文本生成图片、嵌入、重排序、多模态、机器学习、强化学习、深度学习和其他。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

模型目录：选择模型文件所在的目录，必填项。选中“重置”后清空选择内容。

是否 LoRA：是否为 LoRA 模型，如果选择“是”，默认会展示“基模型”参数，其选择框中会列出平台中所有可用的 LoRA 模型供选择。此时，“是否 LoRA”上方的“模型目录”需要选择 LoRA 文件目录。

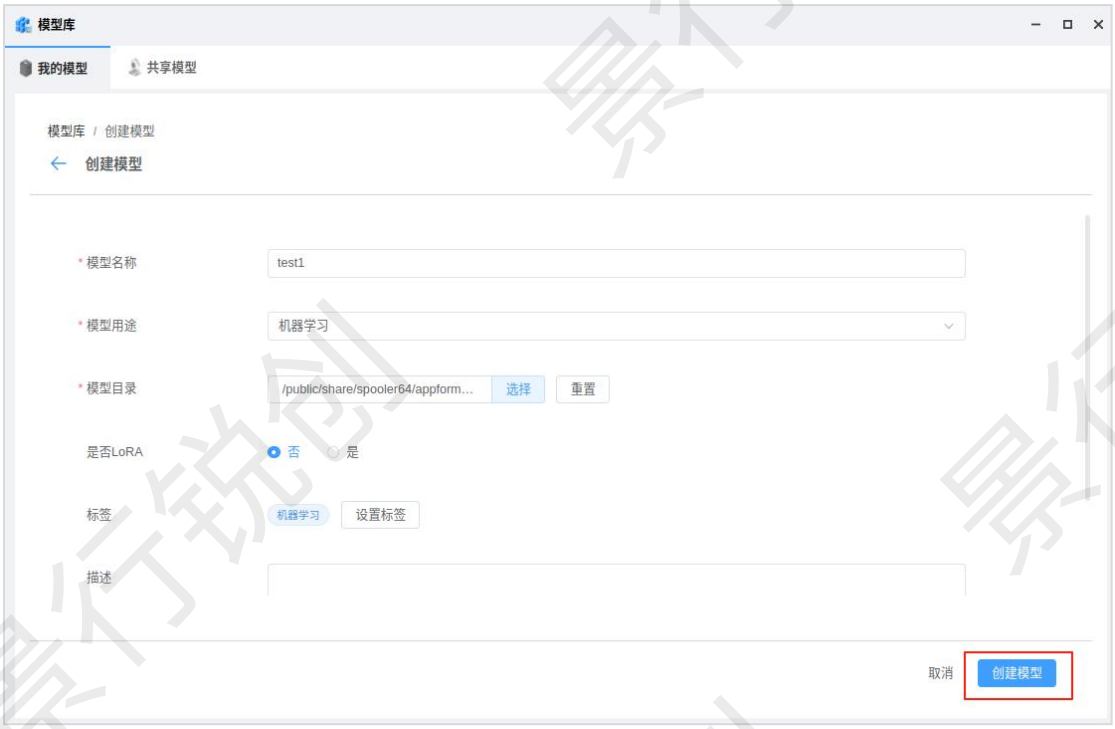


LoRA 选择为“是”的页面

标签：为模型设置标签。选择模型用途的过程中，会默认回填用途标签，也可以选择内置的所有标签。

描述：用于描述模型结果的相关内容，选填项。

点击“创建模型”按钮后，成功创建模型，并会提示“新建模型成功”，如下图所示：



创建模型按钮

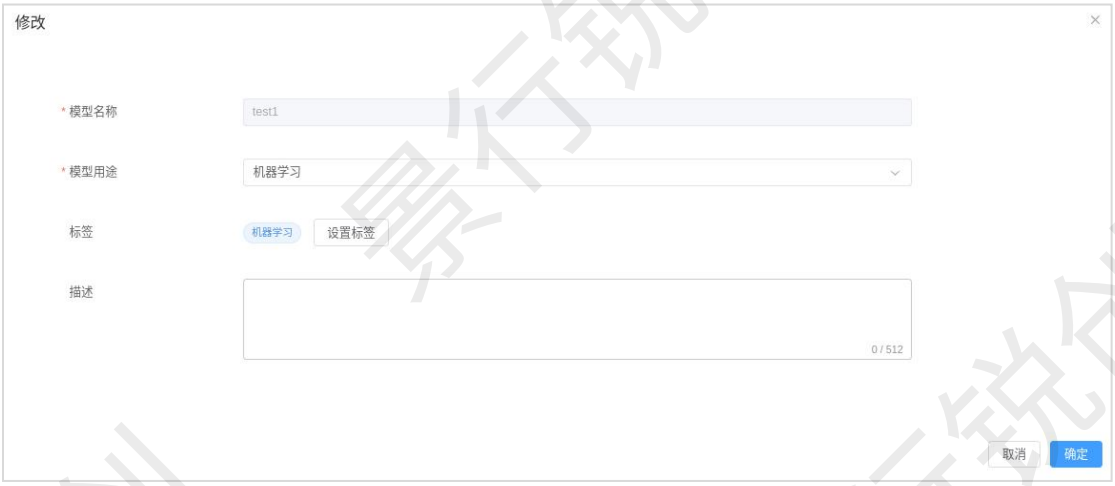
创建模型成功

2.5.3 模型修改

模型卡片上的“修改”按钮，位于模型卡片的下方，如下图所示：

模型修改

该功能是对整个模型的修改。点击“修改”后，弹出修改模型页面，修改参数如页面所示：



模型修改页面

模型名称：创建模型时设置的名称，该参数不可修改；

模型用途：该参数与创建模型时可选参数一致，可修改；

标签：创建模型时设置的标签，可增加或者删除；

描述：与创建模型时设置的描述信息一致，可修改。

2.5.4 模型删除

模型卡片上的“删除”按钮，位于模型卡片的下方，如下图所示：

模型删除按钮

当不勾选“删除模型文件”，点击“确定”按钮后，仅删除此模型卡片；当勾选“删除模型文件”，点击“确定”按钮后，不但会删除该模型卡片，还会删除模型文件，请谨慎操作。

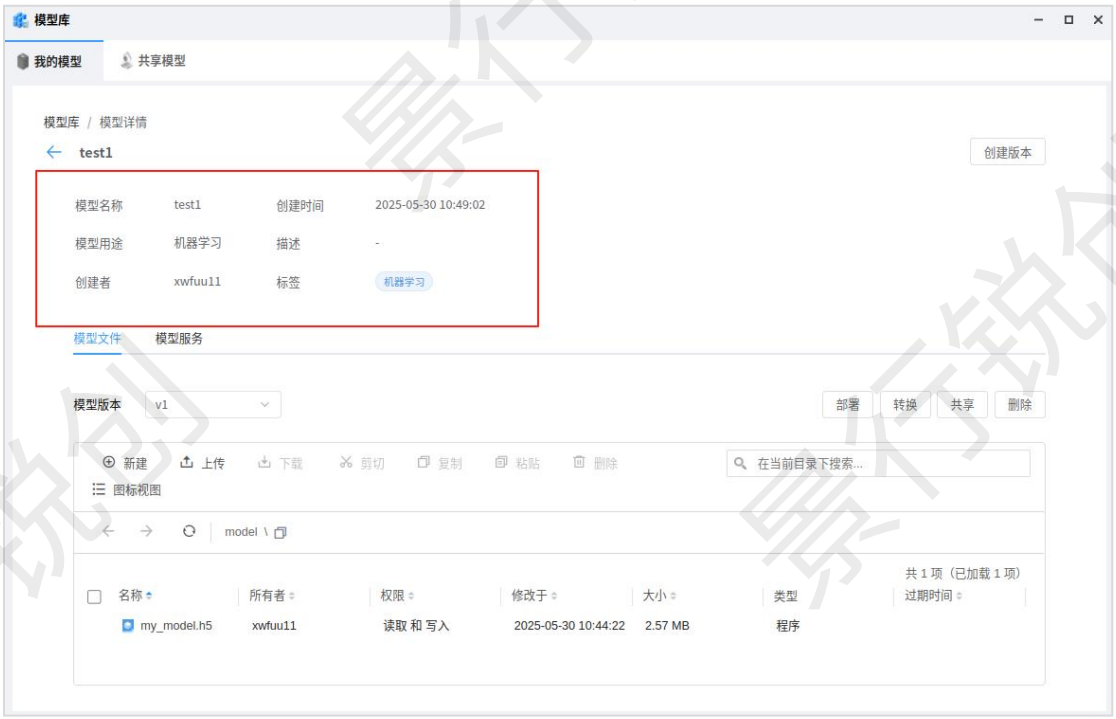
模型删除页面

2.5.5 模型详情

点击具体的模型卡片，如点击“test1”卡片，进入模型详情页面，如下图所示：

模型详情页面

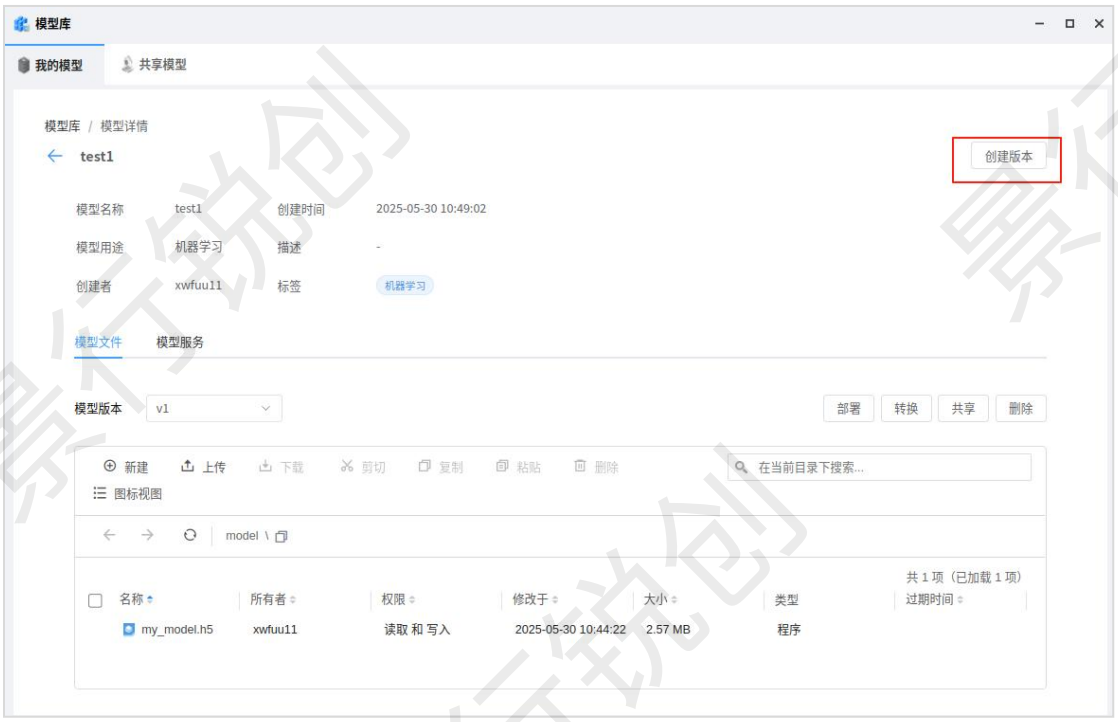
模型详情页面包含了创建模型时设置的所有参数，以及创建版本、模型文件和模型服务模块。



模型详情页面模型创建时设置的参数

2.5.6 创建版本

进入模型详情页面后，点击页面的右上角“创建版本”，实现模型版本的增加。每次创建的版本都会在已有的版本上进行递增，如现有最高版本为 v2，新建后的模型版本将是 v3。另外，创建新的版本会自动继承模型的标签。



模型详情页面创建版本按钮

点击“创建版本”后，弹出创建版本页面，如下图所示：

创建版本页面

创建版本页面参数如下：

模型目录：选择模型文件存储的路径。点击目录右侧的“重置”按钮，将置空模型目录；

创建模型版本重置按钮

描述：添加对模型的描述，会在模型版本下展示该信息。如下图所示：

版本描述信息

创建版本完成后，会在模型版本处增加一个版本，版本号会自动叠加。查看所有版本号，点击模型版本后的版本按钮即可，如下图所示：

模型版本展示

2.5.7 模型文件

在模型详情页面的下侧有“模型文件”模块，如下图所示：

模型文件页面

模型文件页面模型文件默认展示最新版本模型的，内容随着选中的模型版本的改变而改变。模型文件目录支持的操作有新建、上传、下载、剪切、复制、粘贴、删除、图标视图、搜索等操作，如下图所示：

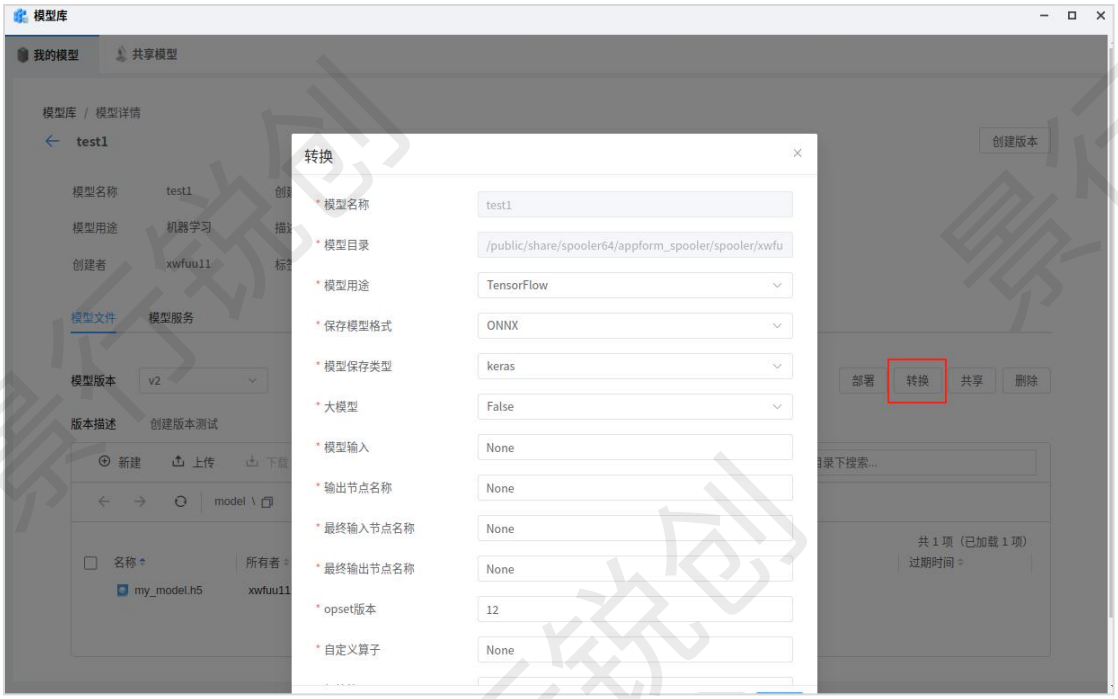
模型文件目录可操作内容

模型版本右侧有“部署”“转换”“共享”“删除”按钮，可以对模型进行部署、转换、共享、删除操作，如下图所示：

模型操作

2.5.7.1 转换模型

转换模型将训练好的模型统一转换为 onnx 模型，以便在模型部署时选择 Triton Inference Server 类型的服务供外部程序调用实现推理功能。点击“转换”按钮，弹出“转换”页面，如下图所示：

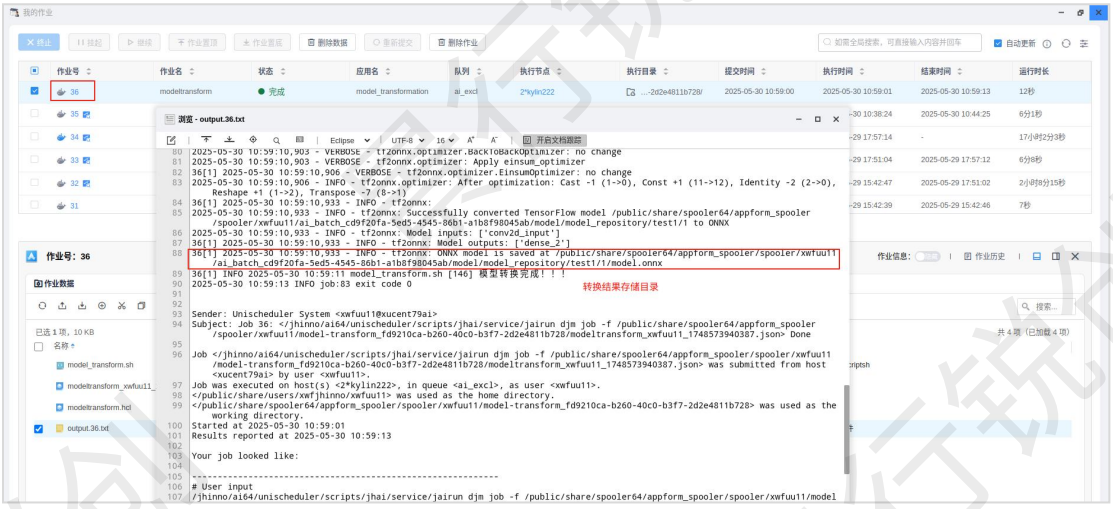


转换模型

选择模型类型、根据模型填写相关参数。点击确定按钮，提交转换模型的作业。

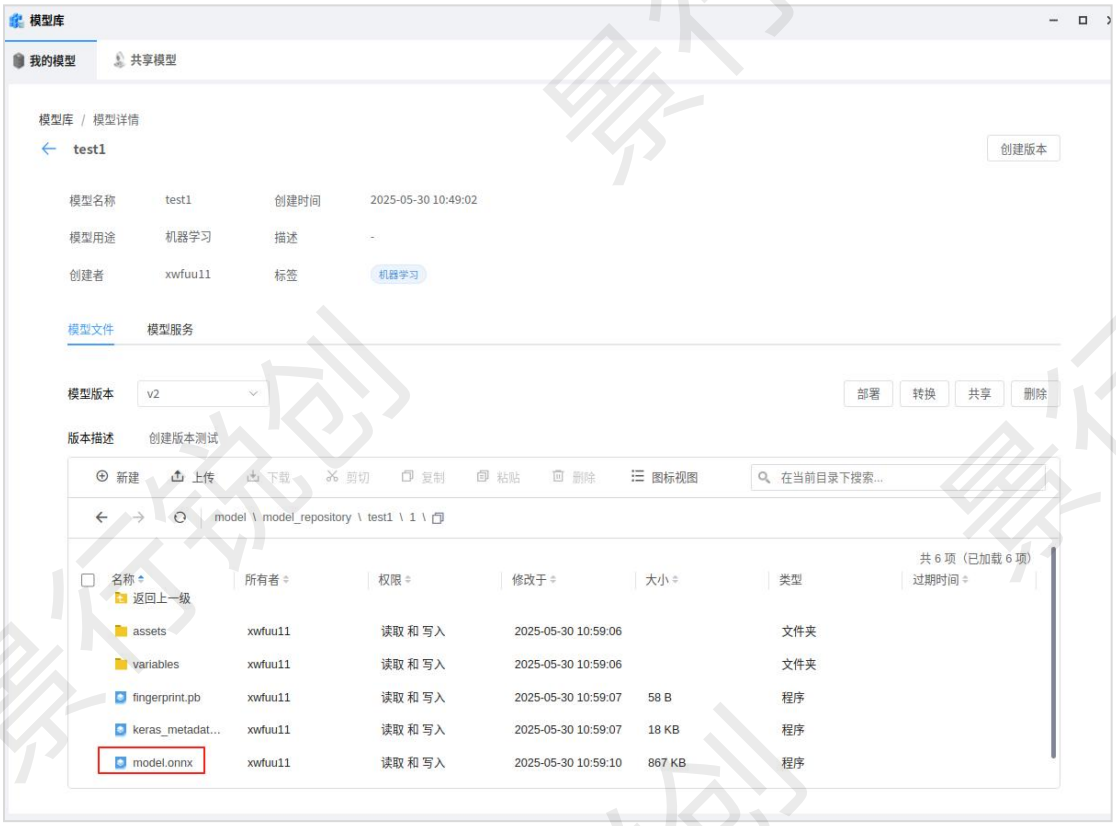
转换模型作业提交

在我的作业中可以看到提交的转换模型作业，如下图：



转换模型作业查看

模型转换成功后，在模型结果中能够看到转换后的模型，如下图：



转换模型结果

2.5.7.2 部署模型

部署模型是将训练好的模型部署成推理服务，可以供外部程序调用实现推理功能。

从模型文件页面进行模型部署

点击“部署”按钮后，跳转至模型部署页面，系统会根据模型用途自动推荐“服务类型”。模型部署页面如下所示：

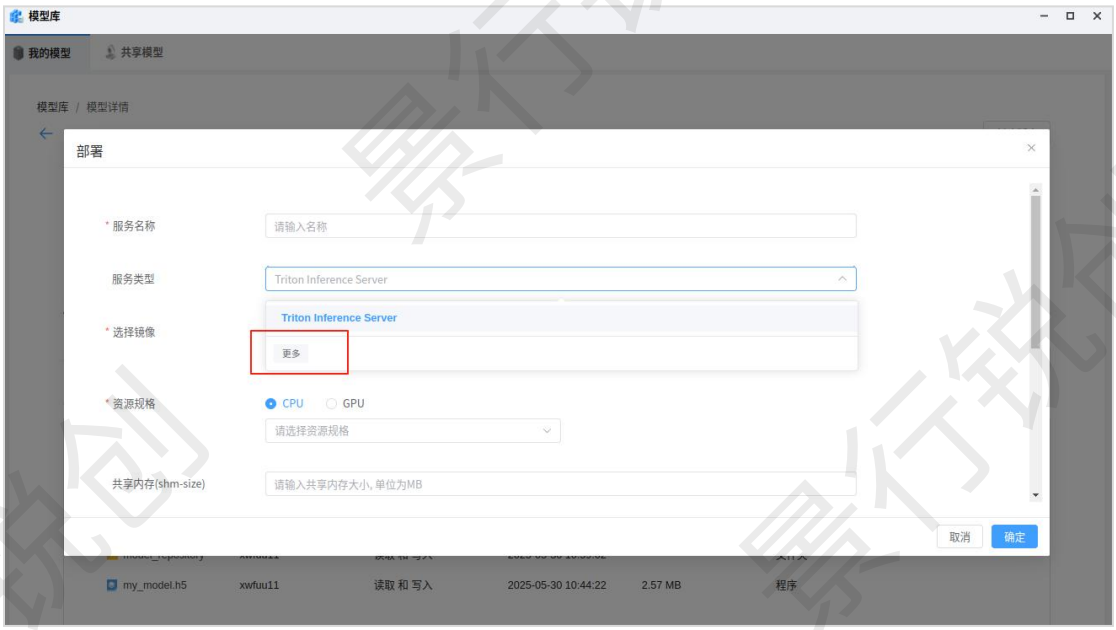
模型部署页面

模型用途和推荐的服务类型对应关系如下：

模型用途	推荐的部署服务框架
机器学习	Triton Inference Server
深度学习	TensorFlow Serving PyTorch Serving Triton Inference Server
强化学习	所有框架

模型用途与服务类型对应表

自动推荐服务除上述规则外，还会结合模型文件共同决定推荐的服务，例如：模型文件下是.h5 后缀的文件，此时点击“部署”按钮，服务类型默认推荐的是“TensorFlow Serving”。推荐服务仅作为参考建议，旨在简化用户操作流程。若系统推荐的服务不符合您的使用场景，您可以点击“更多”按钮选择其他服务选项。



服务类型更多按钮

当确定服务类型后，页面服务参数会随着服务类型的改变而改变，“模型目录”会自动回填，也可以点击“选择”按钮重新选择目录。页面的其他参数详见

“我的服务”模块。

模型部署完成后，部署的服务可以在模型服务找到。如下图所示：

模型服务列表

2.5.7.3 删除

点击“删除”按钮，弹出删除对话框，当不勾选“删除模型文件”，点击“确定”按钮后，仅删除此模型卡片；当勾选“删除模型文件”，点击“确定”按钮后，不但会删除该模型卡片，还会删除模型文件，请谨慎操作。如下图所示：

删除模型

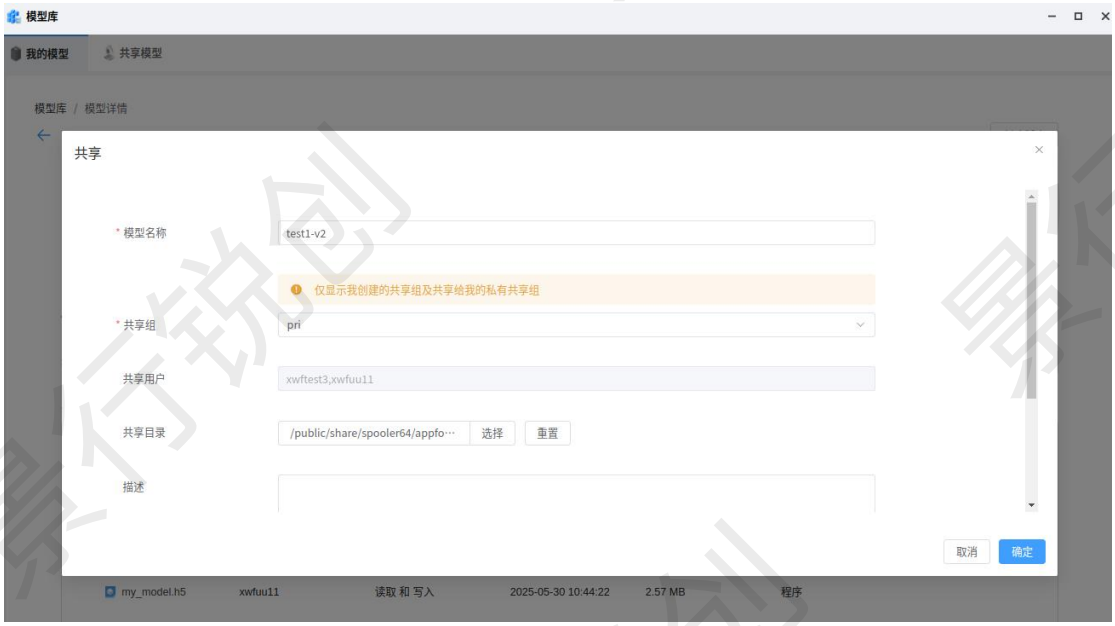
2.5.7.4 共享模型

共享模型是将模型共享给指定共享组，共享组成员可以在自己的共享模型内

查看到的模型。

注意：添加共享模型之前需要先创建共享组。

点击“共享”按钮，弹出共享模型页面，如下图所示：



共享模型页面

图中每个参数的具体含义如下：

模型名称：共享后的模型名称，输入任意符合命名规则的名称即可，可修改。

共享组：选择可见的共享组，选项从我的数据→共享数据区获取。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

共享目录：选择共享组之后显示，只能在所选共享组的共享数据区选择文件夹，点击“选择”按钮，然后在弹出的“预览”窗口中选择文件夹即可，用作存放源数据文件目录，只允许选择一个文件夹。

描述：共享后的数据集描述信息。

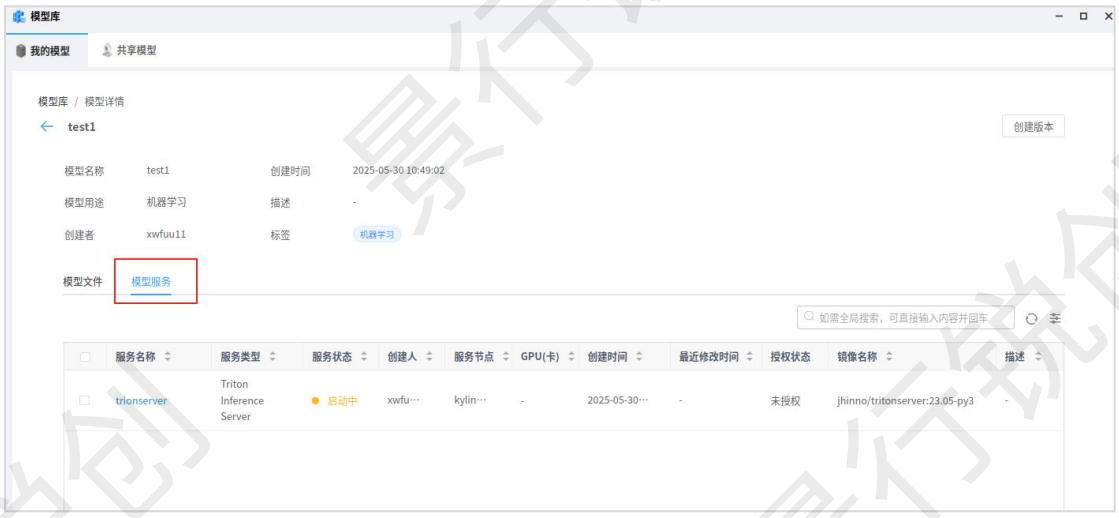
模型后台拷贝：勾选后采用后台形式拷贝源模型文件至共享目录，当前共享页面退出；不勾选则采用前台拷贝方式，当前共享页面不退出。

在共享窗口中填写完成后，点击“确定”按钮，可以保存此次共享的模型。并且在共享模型页面看到，如下图所示：

共享模型列表

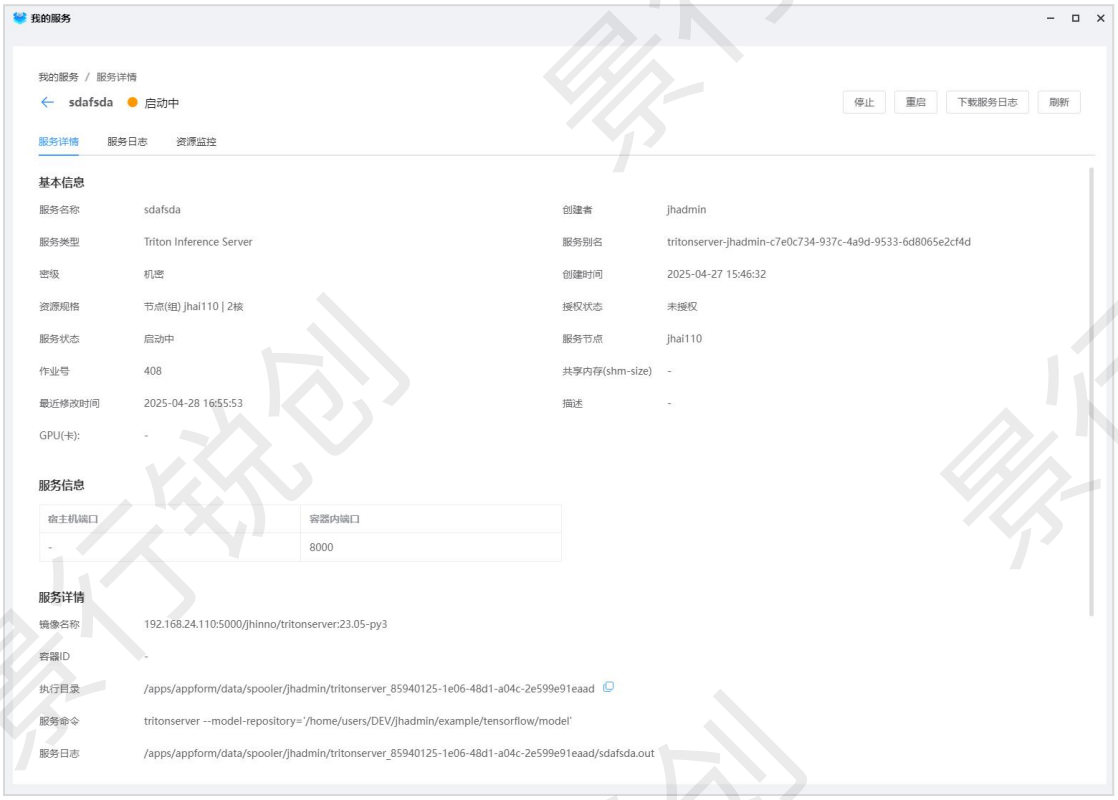
2.5.8 模型服务

模型服务位于模型文件的右侧，包含了在模型库部署该模型的所有服务，如下图所示：



模型服务列表

点击具体服务名称，会弹出该服务的详情页面。



服务详情页面

2.5.9 共享模型

在“模型库”中点击“共享模型”按钮，可以展示共享模型页面，界面如下图所示：

共享模型

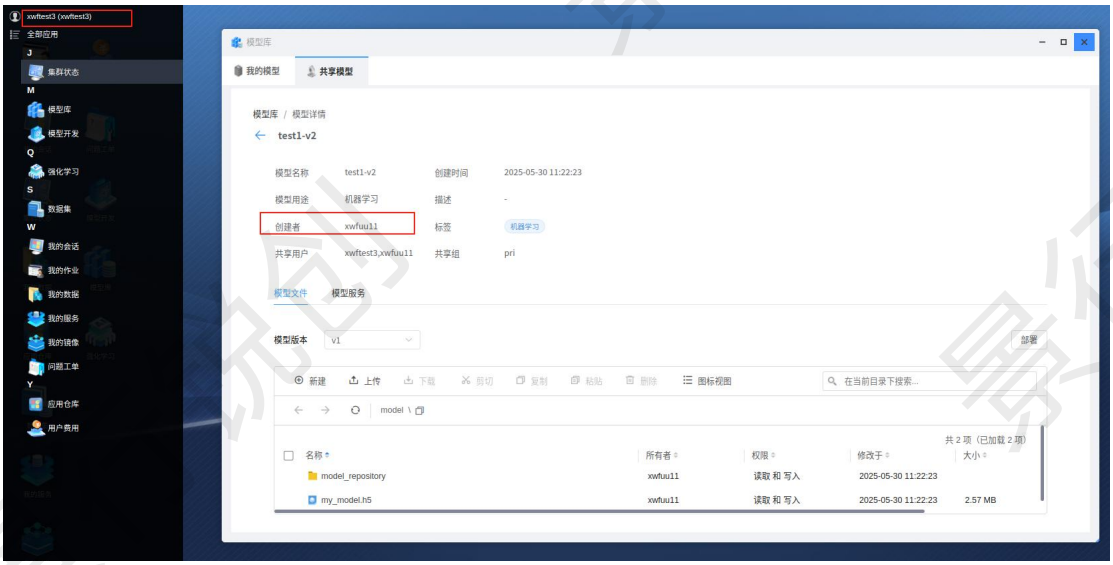
共享模型下存放用户自己共享的模型以及其他用户共享的模型，卡片信息默认展示模型名称、创建时间、描述、标签信息，部分卡片还会展示“修改”“删除”按钮。

用户自己共享的模型：模型卡片上显示修改、删除按钮，用户可以继续修改和删除模型。

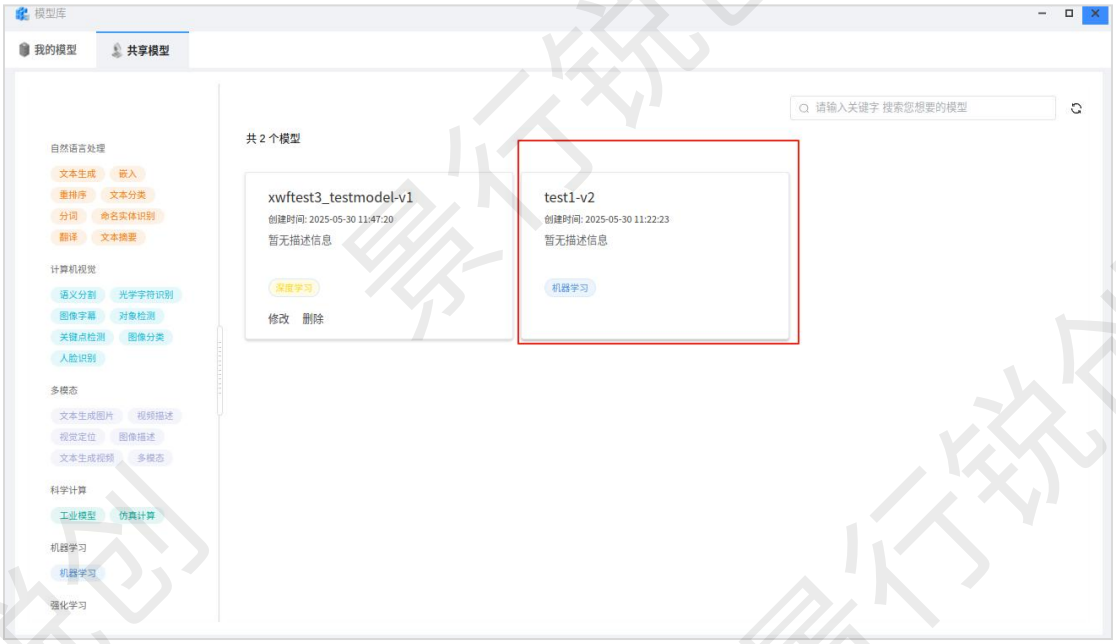
用户自己共享的模型

用户自己共享的模型显示修改、删除按钮

其他用户共享的模型：其他用户共享给我的模型，模型卡片上不显示修改、删除按钮。但管理员可以删除其他用户共享给自己的模型。



其他用户共享的模型

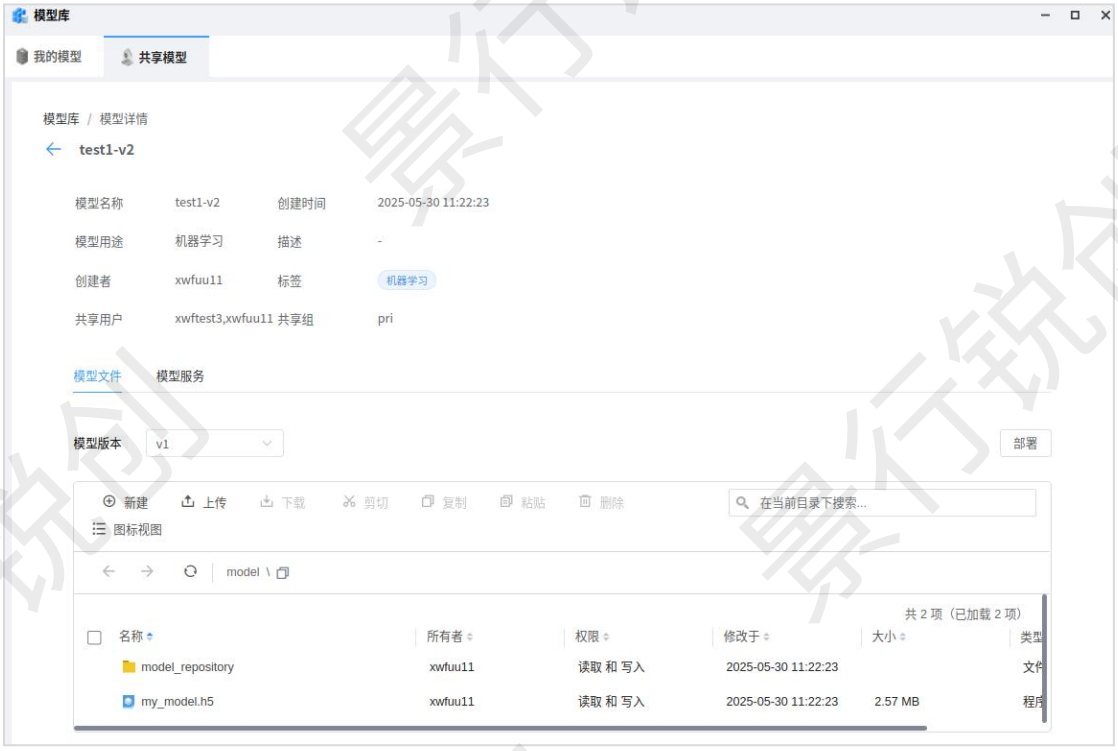


普通用户查看其他用户共享的模型无修改、删除按钮

管理员查看其他用户共享给自己的模型有删除按钮

模型卡片详情

点击具体的模型卡片，如点击“test1-v2”卡片，进入模型详情页面，模型详情页面包含了共享模型的所有信息参数，以及创建版本、模型文件和模型服务模块，如下图所示：



模型详情页面

部署模型：

共享模型部署行为和“我的模型”中一致，详见“我的模型”-“模型文件”-“部署模型”章节。

2.5.10 标签管理

模型标签用于标记模型的类型、用途、语言、格式、参数量等信息，方便模型的管理；如果标记类型不够，用户可以通过“标签管理”模块自行维护标签分类以及标签信息。

在管理门户中，选择“系统管理”，在子菜单中选择“标签管理”，点击“模型标签”，即可维护模型标签分类和模型标签，如下图所示：

模型标签

2.6 我的服务

我的服务包含创建服务和服务管理功能，支持“Tensorboard”“Tensorflow Serving”“Pytorch Serving”“Triton Inference Server”“VisualDL”“MindInsight”“Docker Compose”“自定义服务”类型的服务。如下所示：

- **Tensorboard**：用于可视化查看模型数据。
- **Tensorflow Serving**：将训练的 Tensorflow 模型发布为 Web 服务。
- **Pytorch Serving**：将训练的 Pytorch 模型发布为 Web 服务。
- **Triton Inference Server**：将 onnx 模型发布为 Web 服务。

- **VisualDL**：可视化展示 PaddlePaddle 训练过程数据。
- **MindInsight**：可视化展示 MindSpore 训练过程数据。
- **Docker Compose**：用于定义和运行多容器应用程序。
- **自定义服务**：通过可配置参数自定义服务。

各个角色对服务实例的操作权限：

- **添加者**：对服务具有新建、启动、停止、重启、修改、授权、删除、使用的权限；
- **管理员**：对其他人添加的服务实例仅有启动、停止、重启的权限。

我的服务主页面，如下图所示：

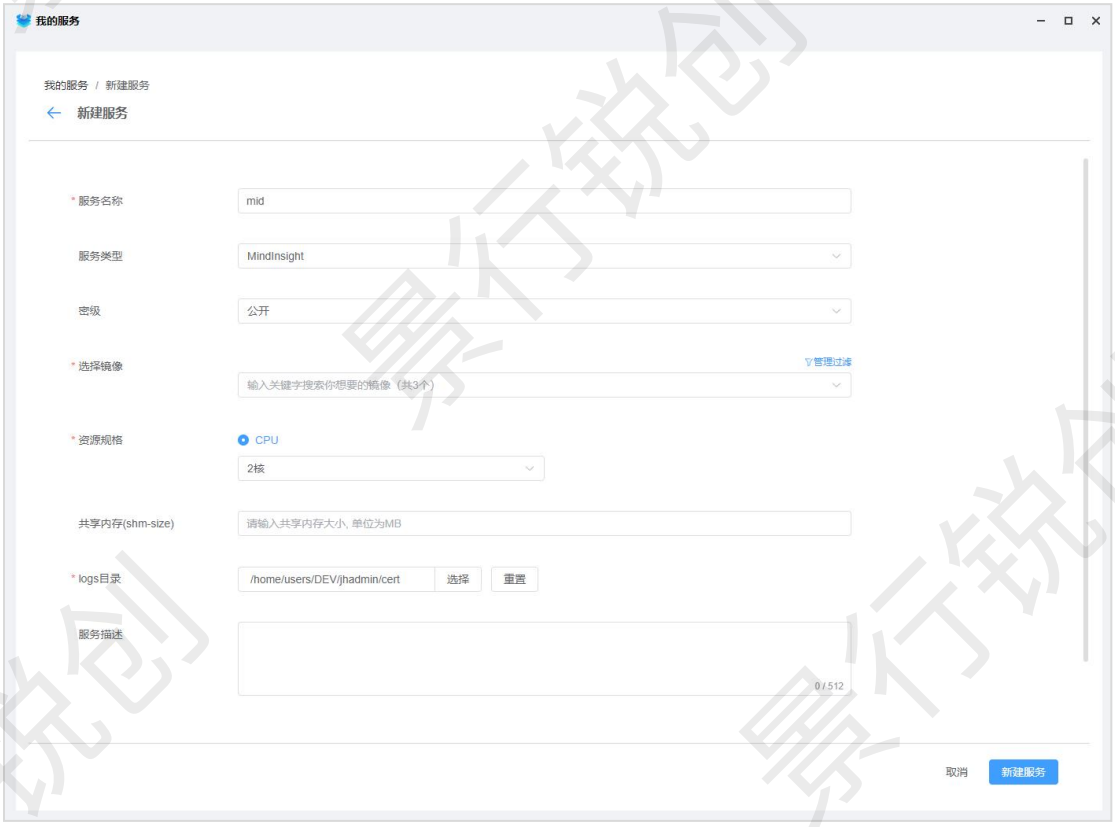
我的服务主界面

2.6.1 新建服务

2.6.1.1 “MindInsight”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 MindInsight 服务类型并填写相关参数，如下图所示：

新建服务功能入口



添加“MindInsight”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

服务类型：选择“MindInsight”。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

镜像名称：选择 mindinsight 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 CPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点/etc/docker/daemon.json 中配置 default-shm-size 参数大小。

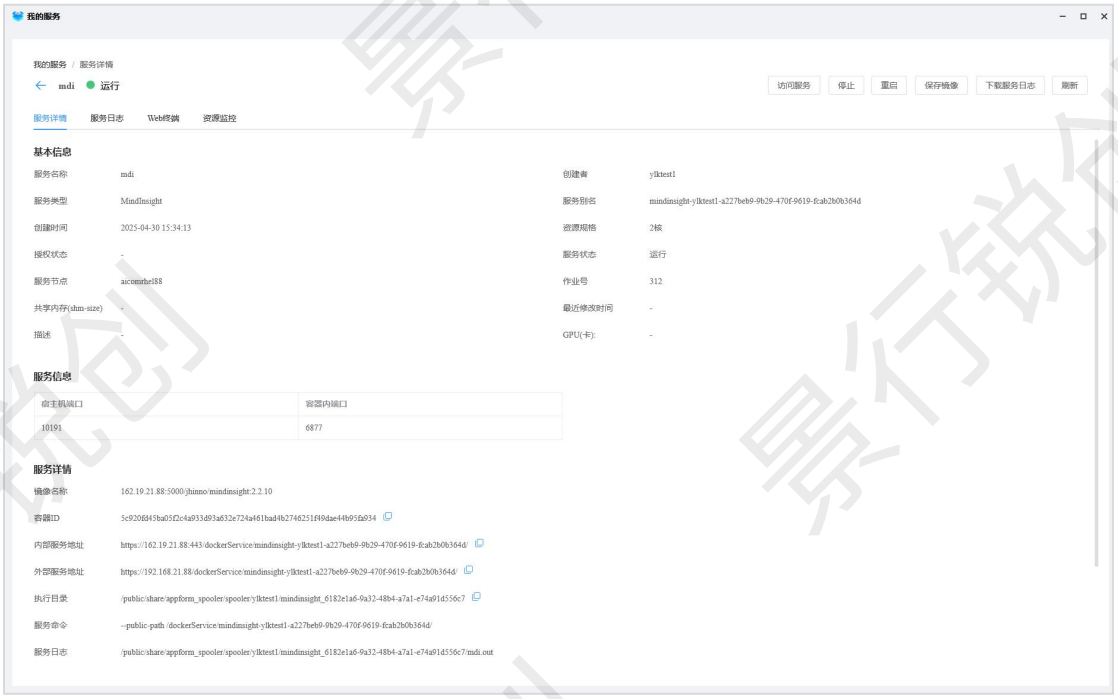
Logs 地址：保存的 Log 文件路径。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Mindinsight 服务

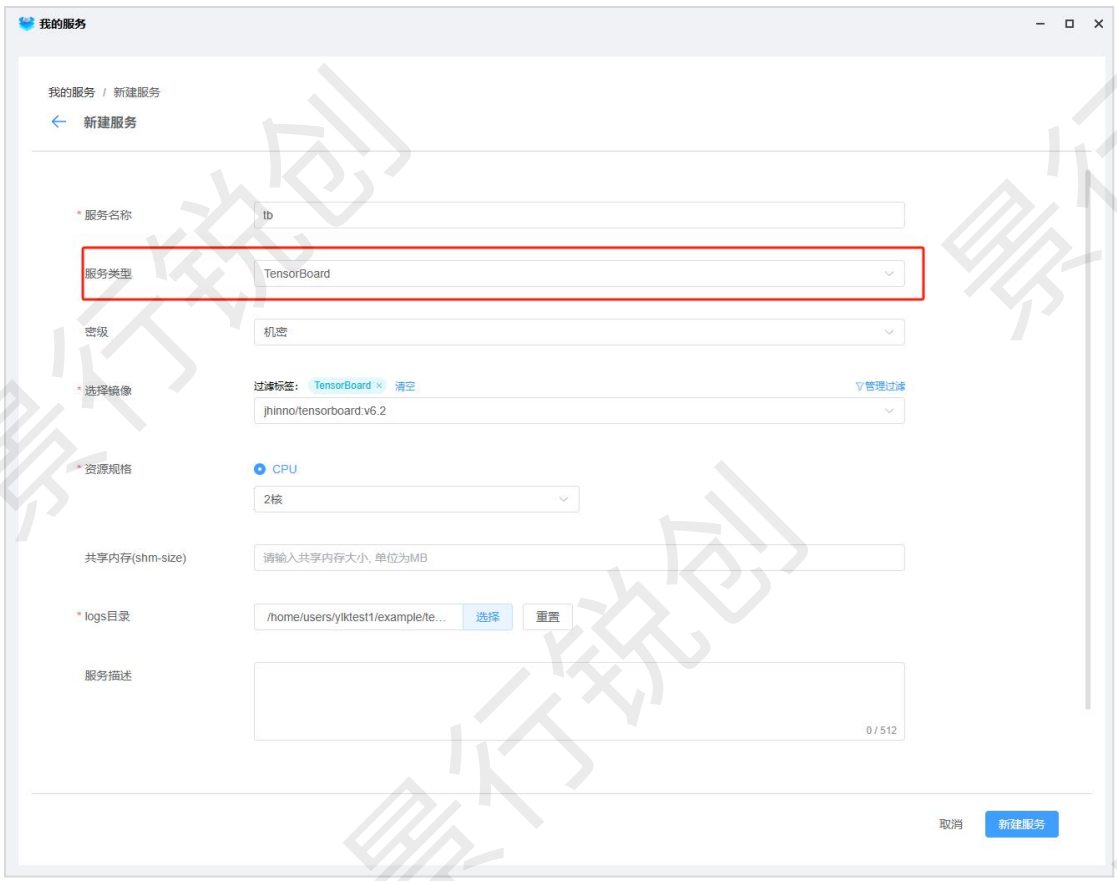
点击创建好的服务名称，如创建的“mdi”，会跳转到服务详情页面，如下图所示：



MindInsight 服务详情

2.6.1.2 “Tensorboard”类型的服务

点击“新建服务”按钮，弹出“创建服务”窗口，选择 Tensorboard 服务类型，并填写相关参数，如下图所示：



添加“Tensorboard”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Tensorboard”。

镜像名称：选择 Tensorboard 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 CPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

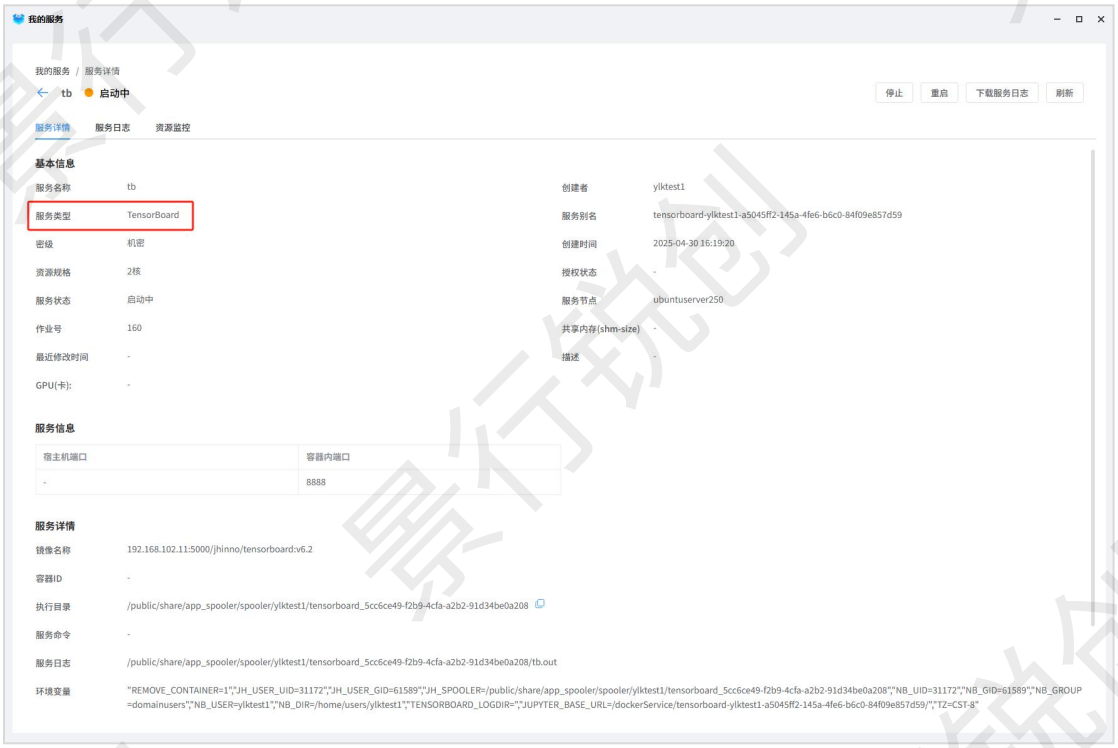
Logs 地址：保存 Log 日志的路径。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Tensorboard 服务

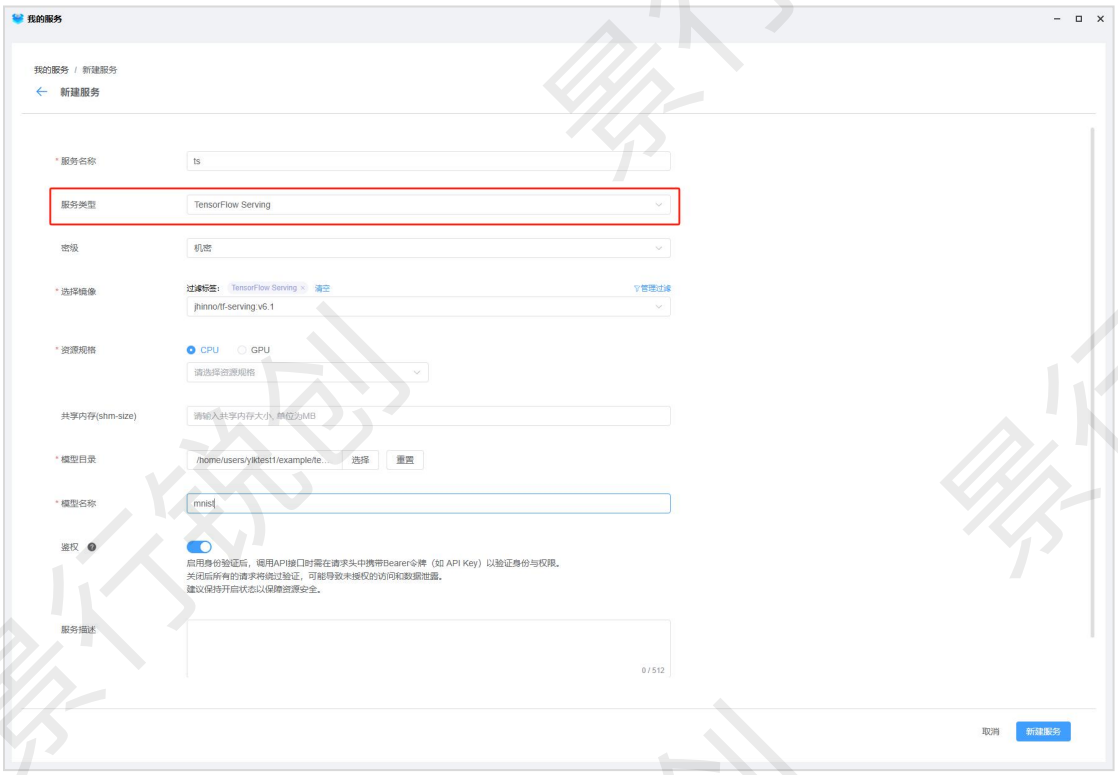
点击创建好的服务名称，如创建的“tb”，会跳转到服务详情页面，如下图所示：



Tensorboard 服务详情页面

2.6.1.3 “Tensorflow Serving”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Tensorflow Serving 服务类型和填写相关参数，如下图所示：



“Tensorflow Serving”服务

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Tensorflow Serving”。

镜像名称：选择 tf serving 镜像。

资源规格：选择启动服务所需要的资源配置。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

模型名称：设置启动服务时，使用的模型名称。

模型目录：选择模型文件所在的目录。

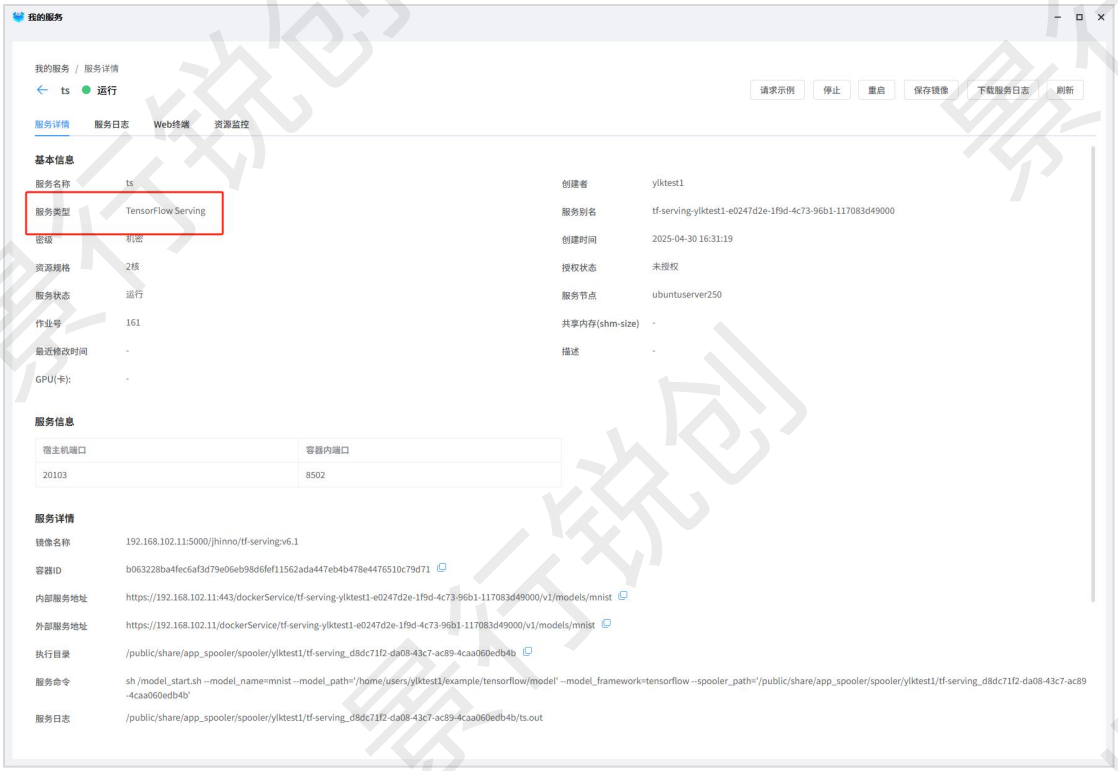
鉴权：选择是否启动 Bearer 令牌（如 API Key）确认身份与权限，开启鉴权后，在调用服务的时候，需要添加 API Key。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Tensorflow Serving 服务

点击创建好的服务名称，如创建的“ts”，会跳转到服务详情页面，如下图所示：



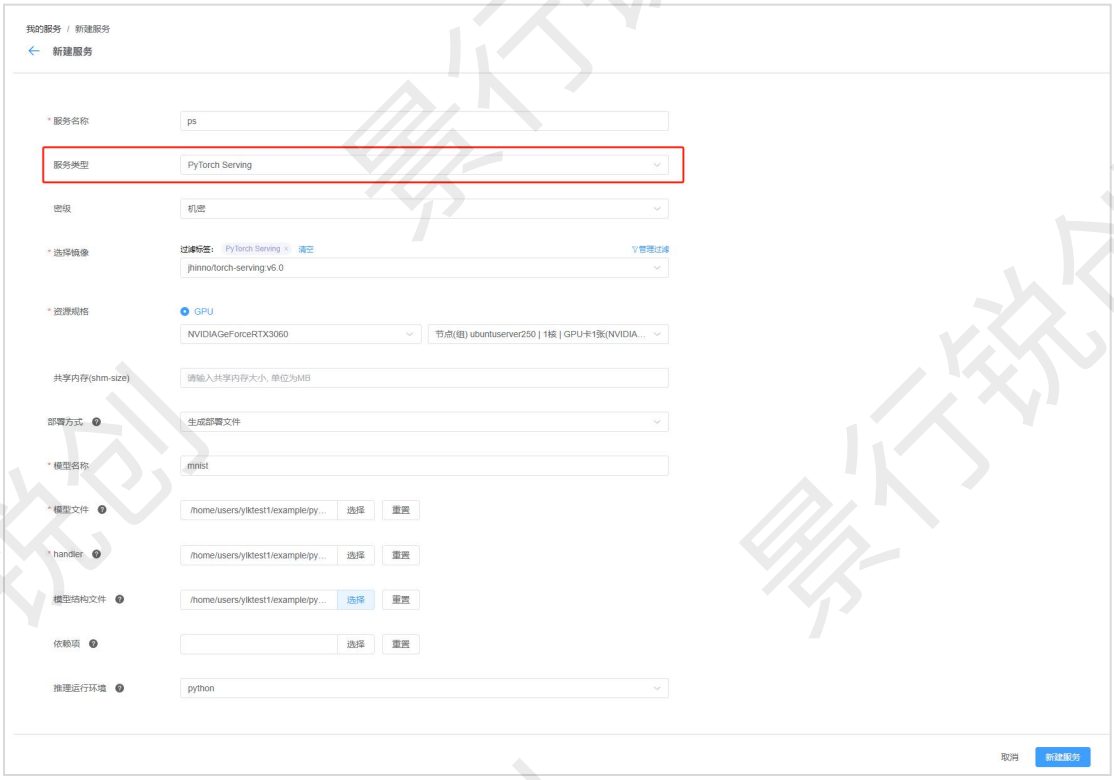
Tensorflow Serving 服务详情页面

在服务详情页面，与普通的服务不同的是，包含了请求示例，也就是如何调用服务的 url。由于开启了鉴权，在调用示例中包含了${YOUR_API_KEY}，使用示例的时候，使用 API Key 替换${YOUR_API_KEY}就可以正常调用服务。Tensorflow Serving 的请求示例内容如下：

Tensorflow Serving 请求示例

2.6.1.4 “Pytorch Serving”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Pytorch Serving 服务类型和填写相关参数，如下图所示：



添加“Pytorch Serving”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Pytorch Serving”。

镜像名称：选择 torch serving 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 GPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点/etc/docker/daemon.json 中配置 default-shm-size 参数大小。

部署方式：可以选择模型文件部署，也可以使用直接可以部署的 mar 文件进行部署。当选择“可部署文件”时，参数仅有“mar 文件”和配置文件，选择“生成部署文件”时，参数除“mar 文件”之外。

模型名称：用户自定义。

模型文件：选择*.pt 模型文件。

handler：选择*.py 文件。处理器文件，选择输入输出处理脚本，一般为 Python 脚本，例如 mnist_handler.py。

模型结构文件：选择*.py 模型结构文件，模型结构文件，选择包含模型结构的 Python 脚本，例如 model.py。

依赖项：选择依赖项。选择加载模型所需的标签等文件，文件类型不限，可以多选，例如 index.json。

推理运行环境：使用哪种语言进行推理，选择 python 或者 python3。

yaml 文件：包含模型配置的 yaml 文件。

配置文件：config.properties 文件中除端口之外的参数输入，多个参数使用。

mar 文件：mar 文件所在路径。

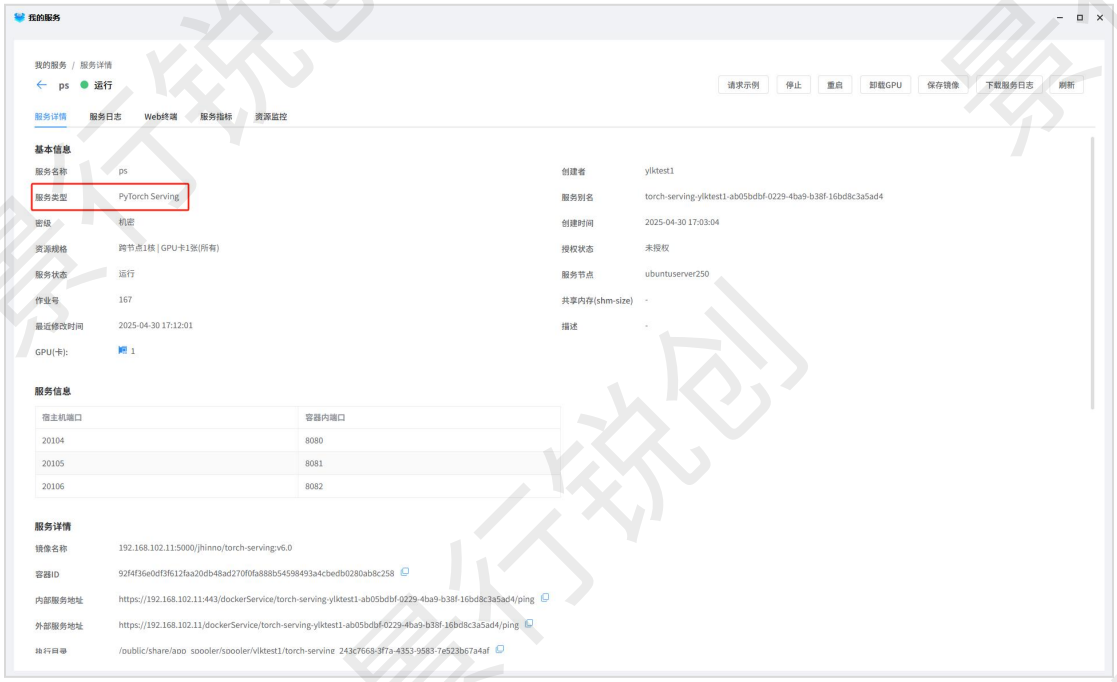
鉴权：选择是否启动 Bearer 令牌（如 API Key）确认身份与权限，开启鉴权后，在调用服务的时候，需要添加 API Key。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Pytorch Serving 服务

点击创建好的服务名称，如创建的“ps”，会跳转到服务详情页面，如下图所示：



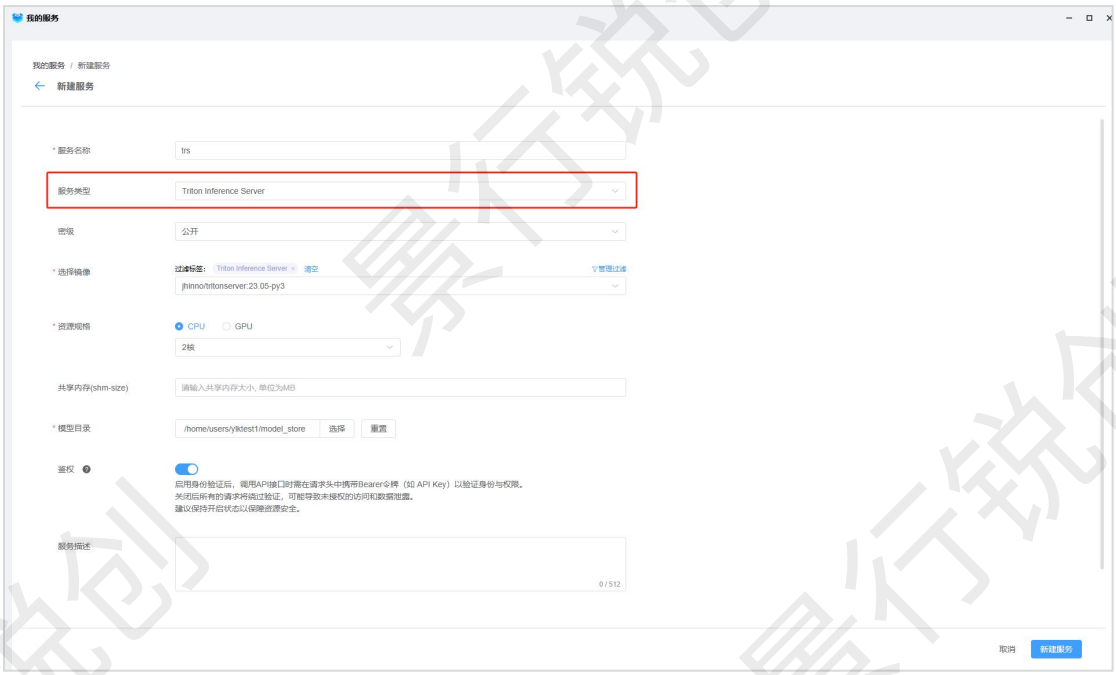
Pytorch Serving 服务详情页面

在服务详情页面，与普通的服务不同的是，包含了请求示例，也就是如何调用服务的 url。由于开启了鉴权，在调用示例中包含了${YOUR_API_KEY}，使用示例的时候，使用 API Key 替换${YOUR_API_KEY}就可以正常调用服务。Pytorch Serving 的请求示例内容如下：

Pytorch Serving 请求示例

2.6.1.5 “Triton Inference Server”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Triton Inference Server 服务类型和填写相关参数，如下图所示：



添加“Triton Inference Server”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安

全、保密。

服务类型：选择“Triton Inference Server”。

镜像名称：选择 triton server 镜像。

资源规格：选择启动服务所需要的资源配置。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点/etc/docker/daemon.json 中配置 default-shm-size 参数大小。

鉴权：选择是否启动 Bearer 令牌（如 API Key）确认身份与权限，开启鉴权后，在调用服务的时候，需要添加 API Key。

模型目录：用户自定义。

注意：

每个模型目录下都至少包含一个模型版本目录和一个模型配置文件。

模型版本目录：包含模型文件，且必须以数字命名，作为模型版本号，数字越大版本越新。

模型配置文件：用于提供模型的基础信息，通常命名为 config.pbtxt。

假设模型存储目录在/examplebucket/models/triton/路径下，模型存储目录的格式如下：

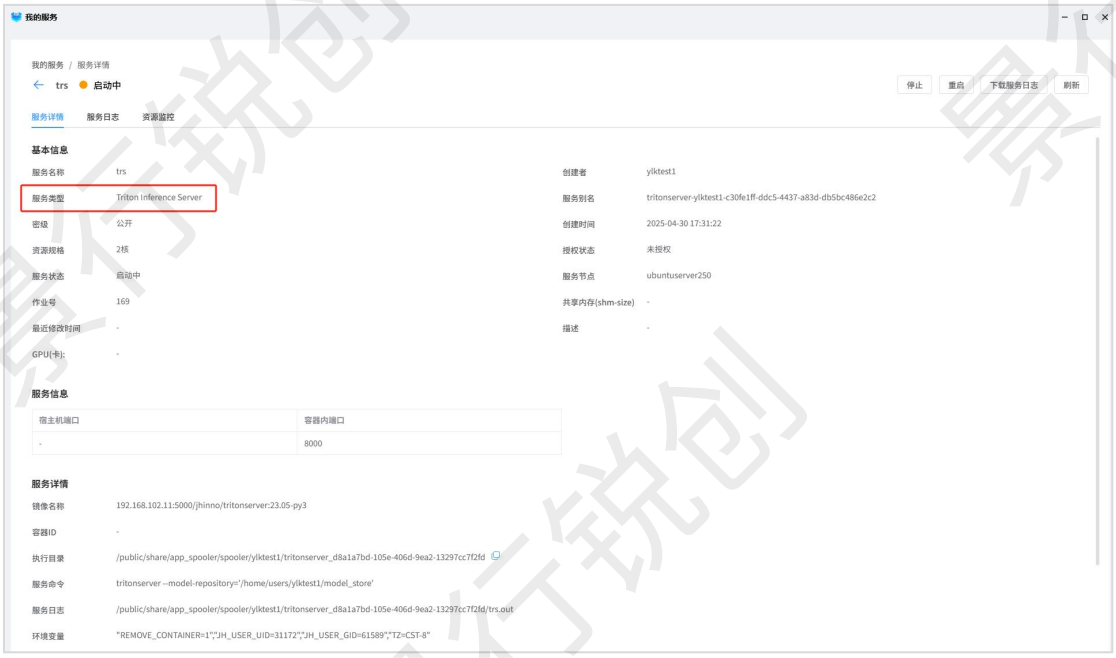
```
triton
└──resnet50_pt
    ├── 1
    │   └── model.pt
    ├── 2
    │   └── model.pt
    ├── 3
    │   └── model.pt
    └── config.pbtxt
```

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Triton Inference Server 服务

点击创建好的服务名称，如创建的“trs”，会跳转到服务详情页面，如下图所示：



Triton Inference Server 服务详情页面

在服务详情页面，与普通的服务不同的是，包含了请求示例，也就是如何调用服务的 url。由于开启了鉴权，在调用示例中包含了${YOUR_API_KEY}，使用示例的时候，使用 API Key 替换${YOUR_API_KEY}就可以正常调用服务。Pytorch Serving 的请求示例内容如下：

Triton Inference Server 请求示例

注意：

（1）在部署模型时，模型算法名称应和模型文件所在目录保持一致，否则在加载模型时会出错。如模型算法名称为 maskrcnn，那么目录结构应该如下：

```
triton
└———maskrcnn
    ├——— 1
    |    └——— model.onnx
    └——— config.pbtxt
```

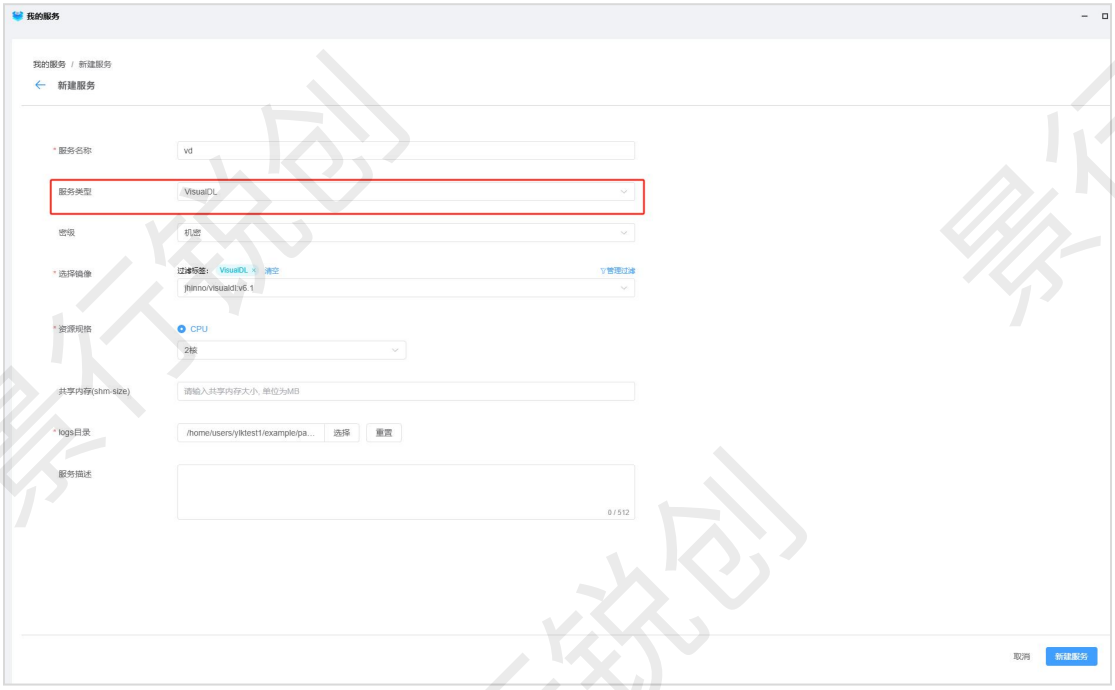
（2）部署模型时，会出现算子不支持情况，请使用 tensorflow serving 或者 torch serving 部署。

（3）如果需要部署多个版本的模型，需要在 config.pbtxt 文件中添加 version_policy 参数如：

```
version_policy: { all: {}} #所有版本
version_policy: { latest: { num_versions: 2}} #最近的两个版本
version_policy: { specific: { versions: [1,3]}} #指定版本 1 和 3
```

2.6.1.6 “VisualDL”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 VisualDL 服务类型和填写相关参数，如下图所示：



添加“VisualDL”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“VisualDL”。

镜像名称：选择 visualdl 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 CPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

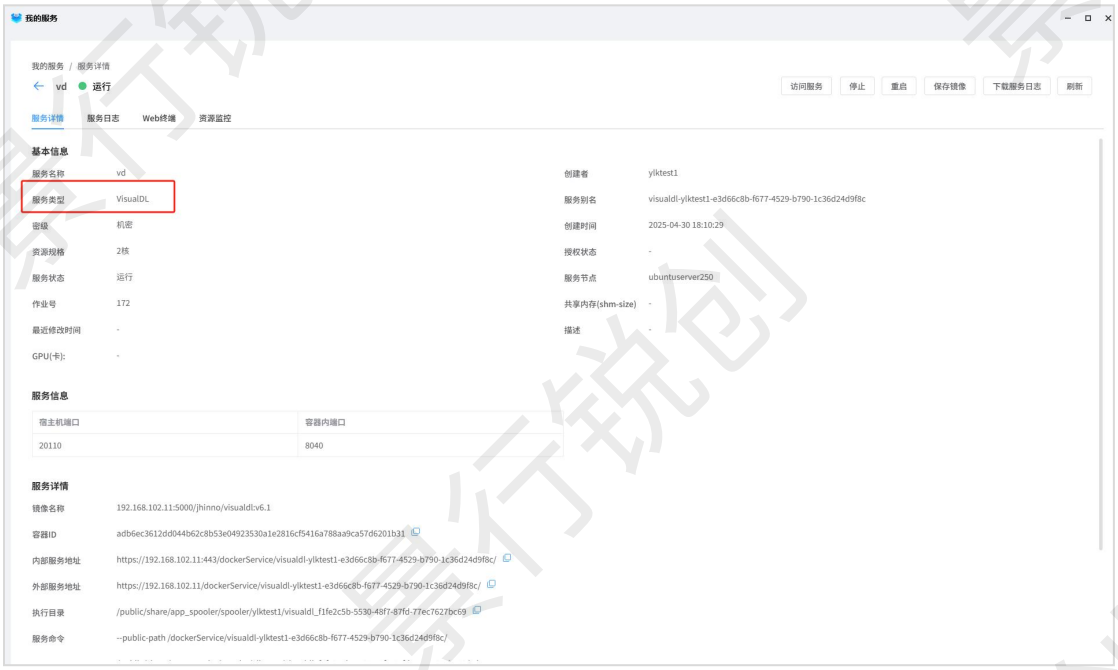
logs 地址：保存的模型 Log 路径。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 VisualDL 服务

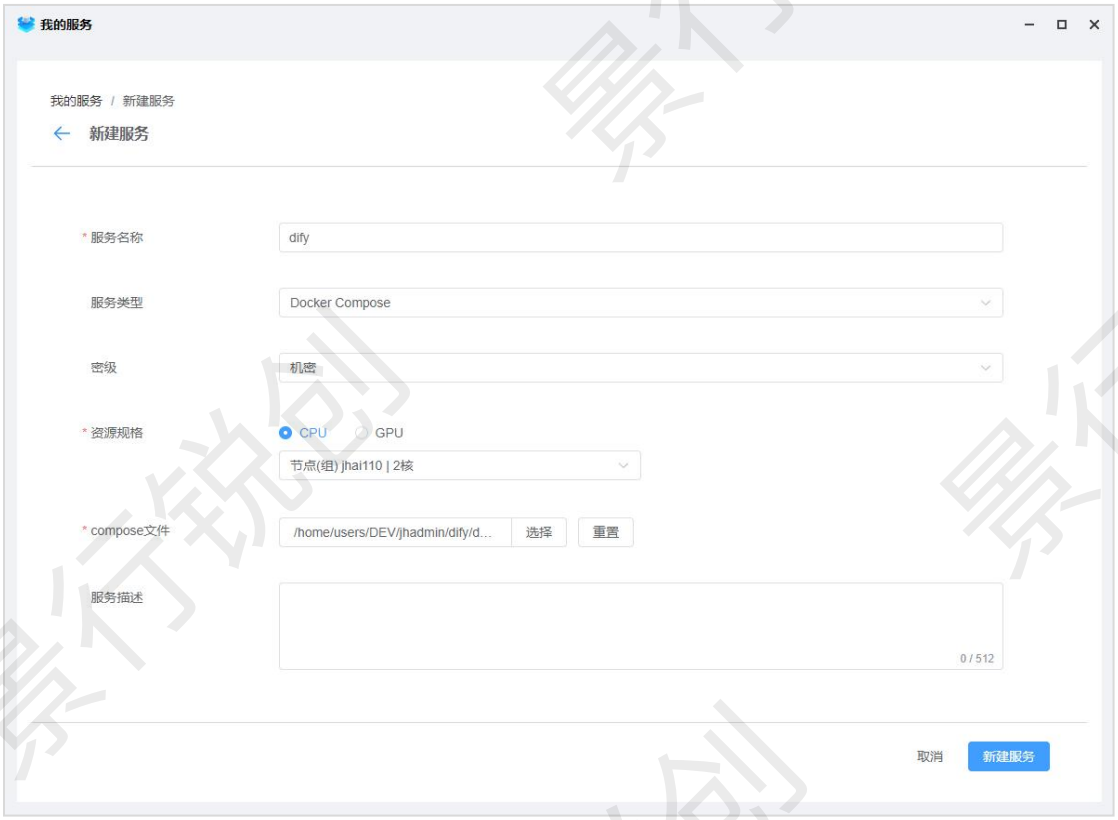
点击创建好的服务名称，如创建的“vd”，会跳转到服务详情页面，如下图所示：



VisualDL 服务详情页面

2.6.1.7 “Docker Compose”类型服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Docker Compose 服务类型和填写相关参数，如下图所示：



添加“Docker Compose”类型的服务配置

服务参数的含义：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Docker Compose”。

资源规格：选择启动服务所需要的资源配置，默认只能选择 GPU 资源。

compose 文件：选择启动 docker compose 服务需要的 yaml 配置文件。

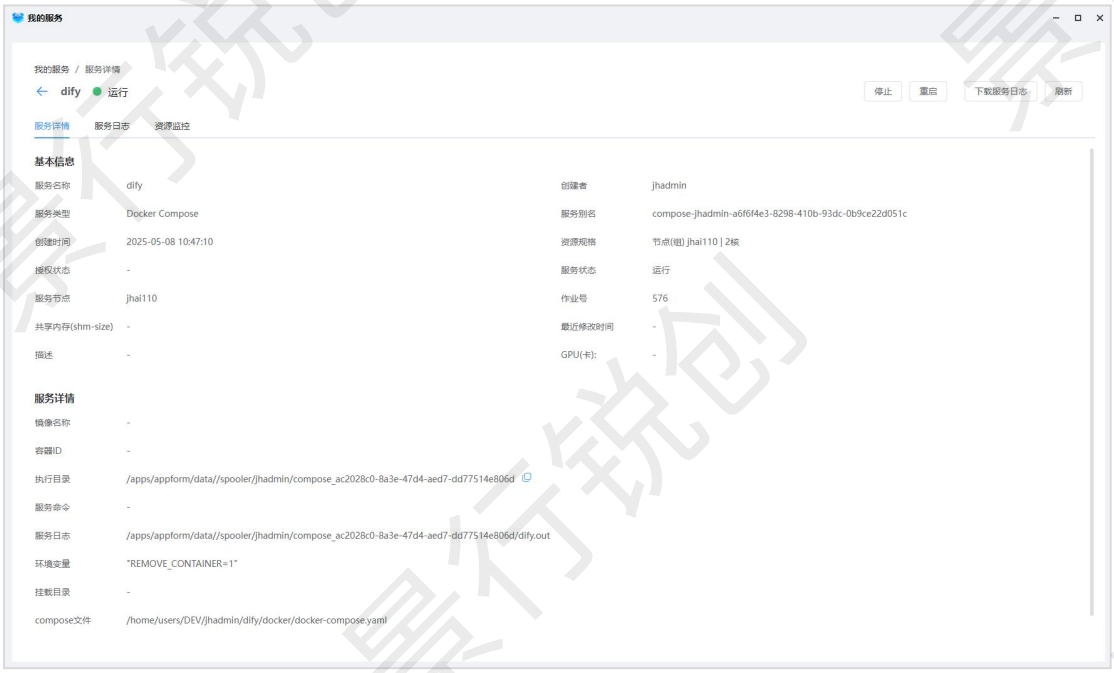
自定义参数：手动添加 LLaMa Box 中的更多参数，异常参数或重复参数可能会导致服务启动失败。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务实例，在我的服务列表可以看到新建的 Docker Compose 服务，如下图所示：

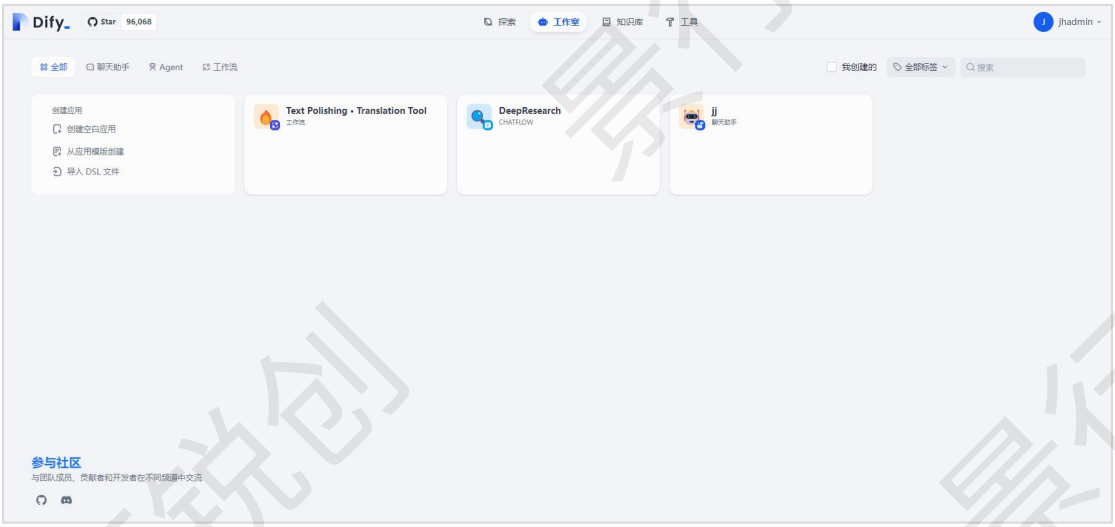
Docker Compose 服务

点击我的服务列表中的 Docker Compose 服务名称，跳转至 Docker Compose 服务详情页面，如下图所示：



Docker Compose 服务详情页面

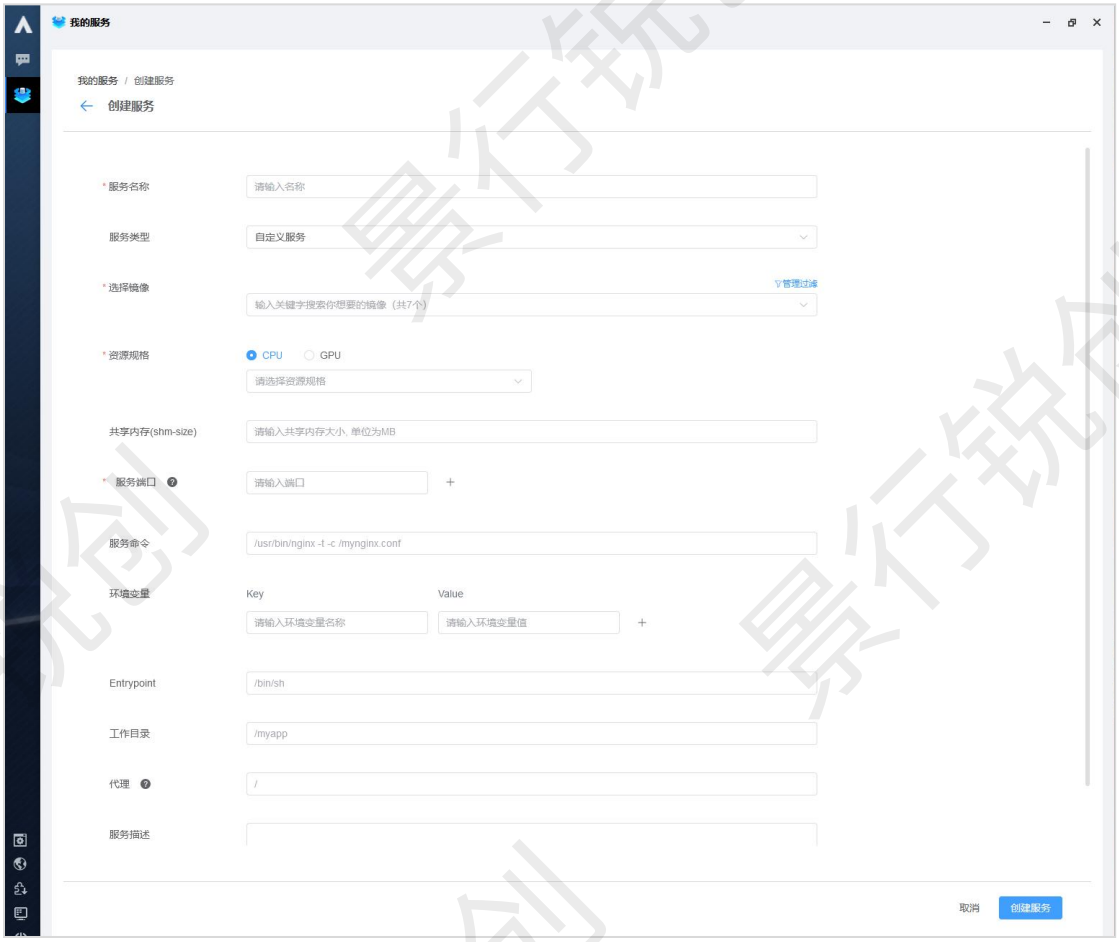
示例中启动“Dify”，按照 Compose 文件中的设置，打开 Dify 界面，如下图所示：



Dify 界面

2.6.1.8 “自定义服务”类型的服务

点击“新建服务”按钮，弹出“创建服务”窗口，选择自定义服务服务类型和填写相关参数，如下图所示：



添加“自定义服务”类型的服务配置

服务参数的含义：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“自定义服务”。

镜像名称：选择自定义服务镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 GPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

服务端口：容器内服务使用的端口。

服务命令：容器内服务执行的命令。

Entrypoint：设置容器启动时执行的主程序，并允许向该程序传递参数。

工作目录：容器内部的一个目录。

代理：代理用于设置容器内 Web 服务的访问路径，仅支持 HTTP 服务。例如，若容器内 Web 服务路径为 http://ip:port/demo，可以将代理设置为 /demo，启动后通过服务地址访问该服务。

服务描述：输入服务描述信息，可选。

2.6.2 访问服务

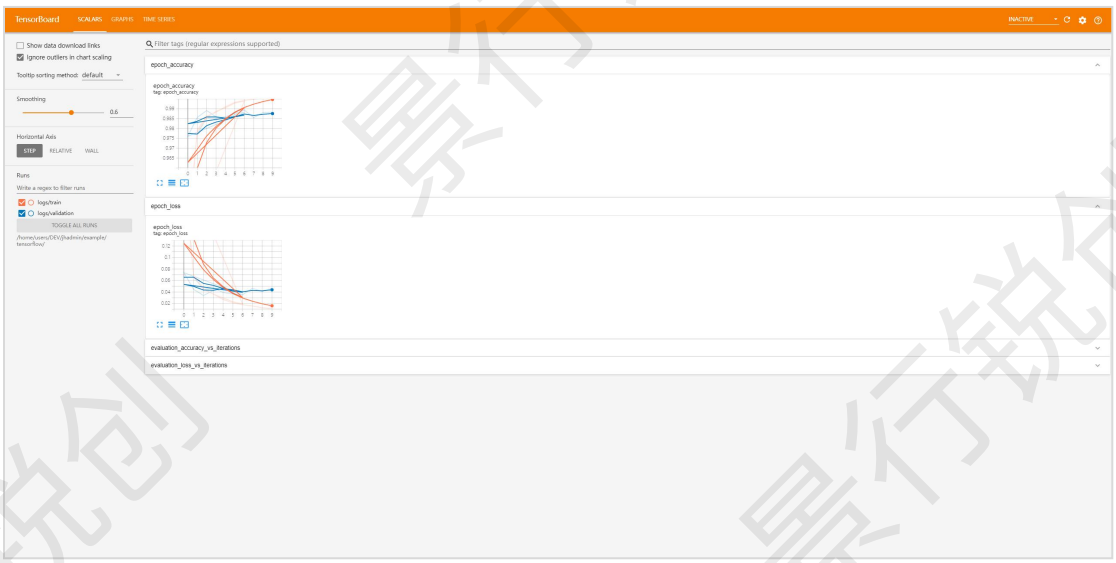
点击某些类型服务（“MindInsight”“Tensorboard”“VisualDL”和“自定义服务”）名称，弹出“服务详情”页面，右上角点击“访问服务”按钮，会打开服务相关页面，如下图所示：

访问服务入口

注意：

(1) 访问服务功能仅在服务实例的状态为“运行”状态时可用。

(2) 访问服务功能仅限用户自己通过服务管理发布的服务或训练模型打开的服务，开发中心创建的服务不存在该功能。



服务内容

2.6.3 保存镜像

保存镜像可以将服务当前容器保存为新的容器镜像。点击“保存镜像”按钮，弹出“保存镜像”窗口，如下图所示：

保存镜像

图中每个参数的具体含义如下：

目标命名空间：选择镜像保存的命名空间。

镜像名称：输入保存后的镜像名称。

Tag：输入镜像版本，默认为 latest。

点击“确定”按钮，将运行服务的容器保存为镜像，如下图所示：

保存镜像

2.6.4 下载服务日志

下载服务日志会将运行服务过程中的日志保存到本地。点击“下载服务日志”按钮，自动下载日志到本地，如下图所示：

下载服务日志

保存的日志命名规则为 output.+作业号+.txt，日志保存在浏览器默认下载目录。

下载日志命名详情

2.6.5 刷新

刷新服务会刷新整个服务内容，服务状态不会自动更新，只有在点击刷新的

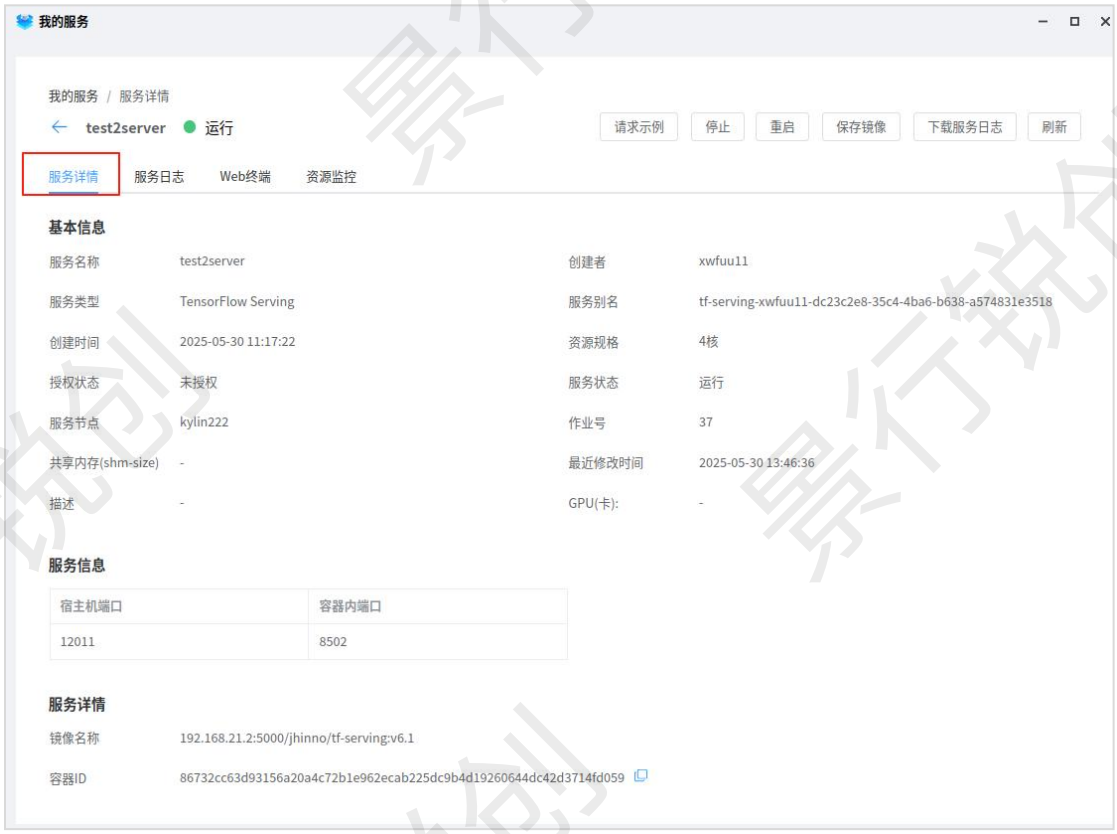
情况下更新，如下图所示：

刷新服务

2.6.6 服务详情

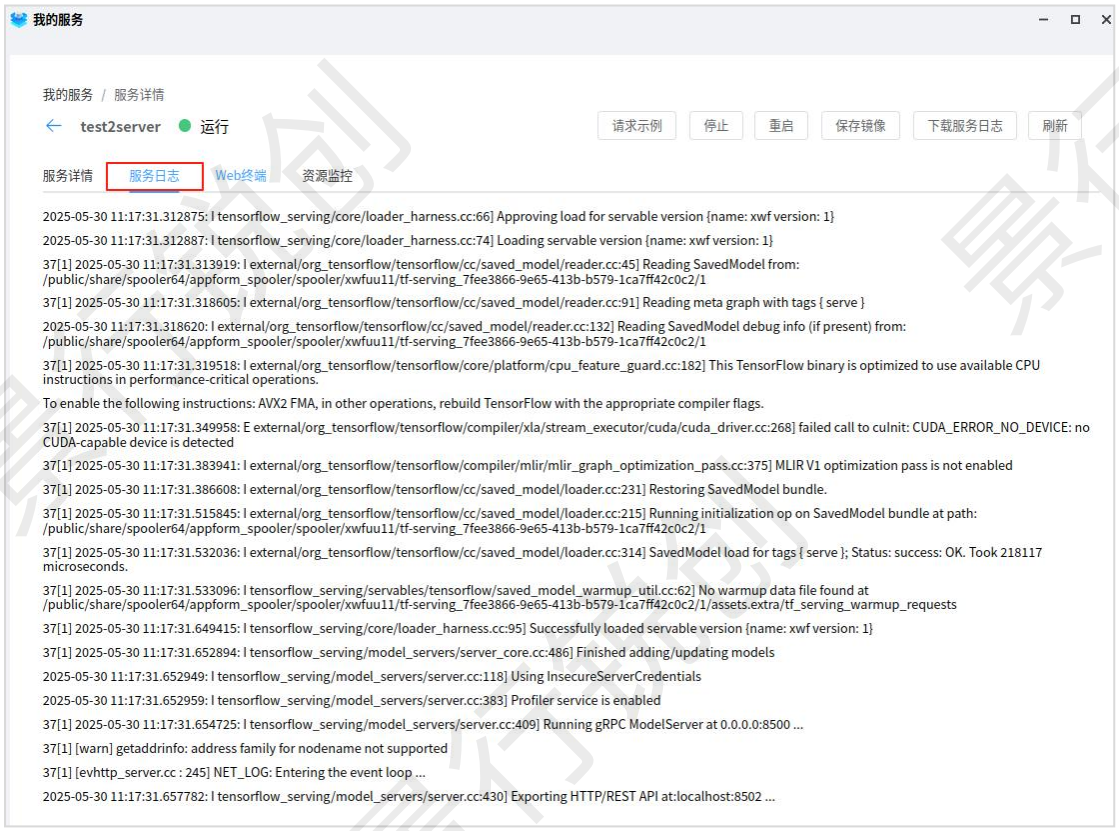
点击服务列表中的服务实例的服务名称，会打开服务详情页面。

服务详情：展示服务的基本信息，服务信息和服务详情，如下图所示：



服务详情

服务日志：点击“服务日志”，切换至服务日志页面，查看服务日志。具体详情如下图所示：

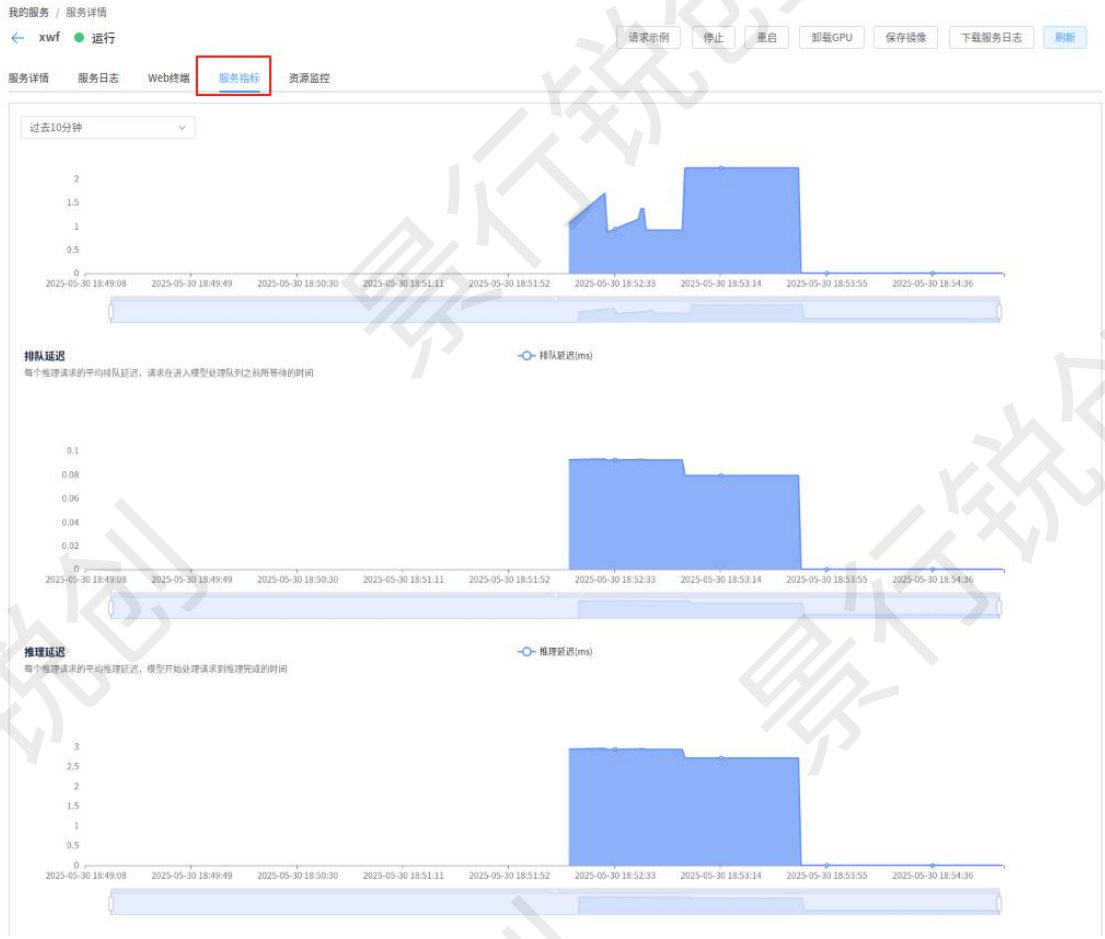


服务日志

Web 终端：点击“Web 终端”，切换至 Web 终端页面，可以在页面模拟终端操作服务容器，如下图所示：

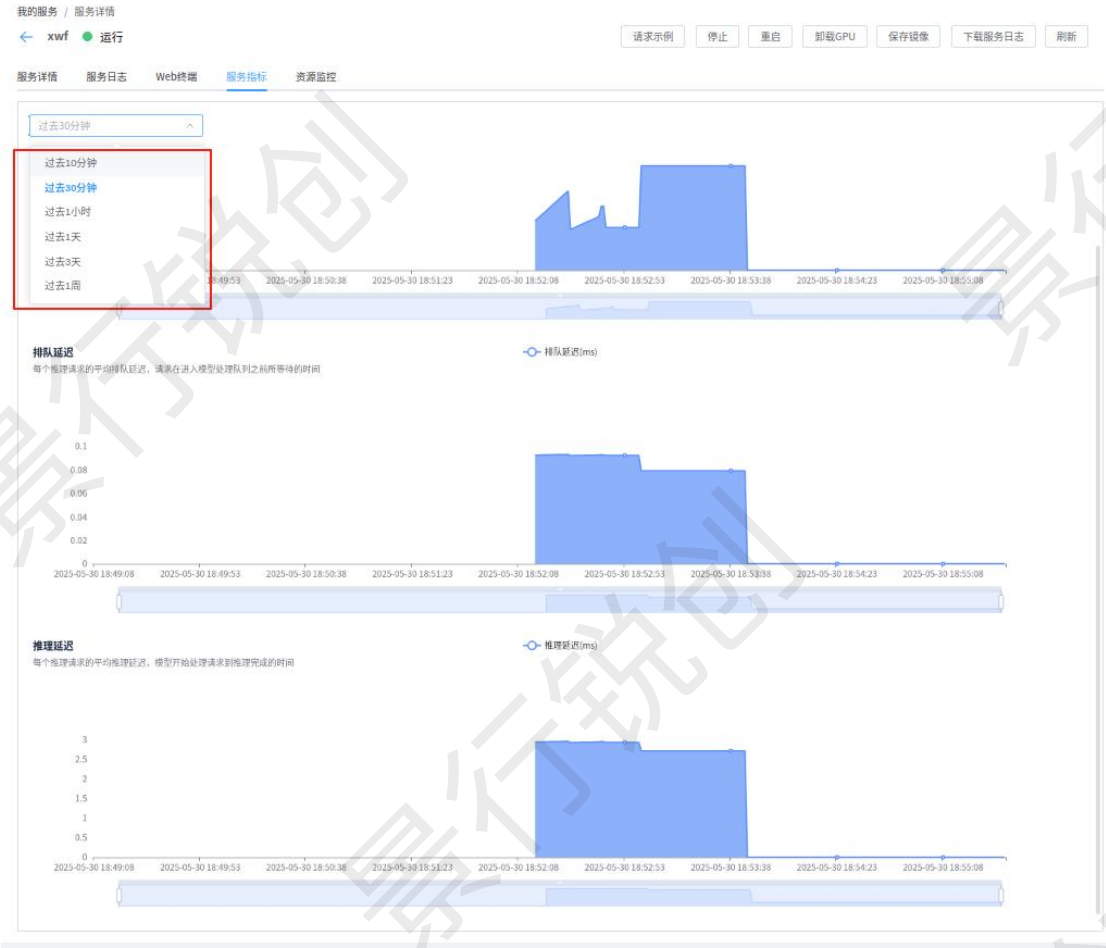
Web 终端

服务指标：点击“服务指标”，切换到服务指标页面，查看推理服务的服务指标的变化。如下图所示：



服务指标内容

这些指标可以按照时间段筛选，时间段包含过去 10 分钟、过去 30 分钟、过去 1 小时、过去 1 天、过去 3 天和过去 1 周，如下图所示：



按照时间段查看指标

注意：只有运行状态的 Pytorch Serving 服务有“服务指标”按钮。

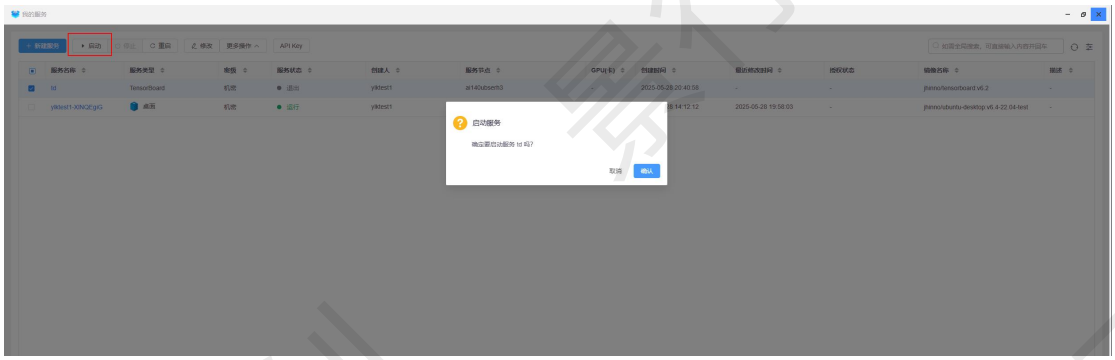
资源监控：点击“资源监控”，切换到资源监控页面，看到服务使用的资源情况。启动服务只选择 cpu 资源，资源监控将显示 cpu 和内存的使用情况；如果启动服务选择 gpu 资源，则在资源监控页面会多出 gpu 的监控信息。如下图所示：

未选择 GPU 的资源监控

选择 GPU 的资源监控

2.6.7 启动服务

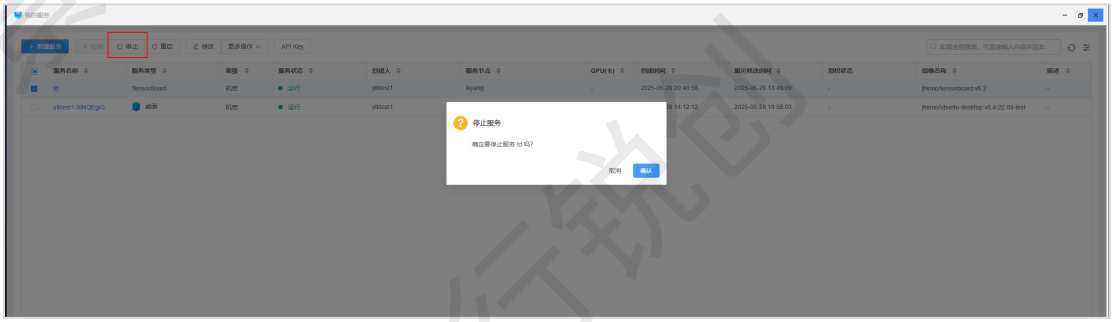
选中服务列表中某一条在服务管理启动后处于非启动状态的服务实例后，点击“启动”按钮，弹出“是否启动此服务”提示窗口，点击“确定”按钮，即可启动该条服务实例。



启动服务

2.6.8 停止服务

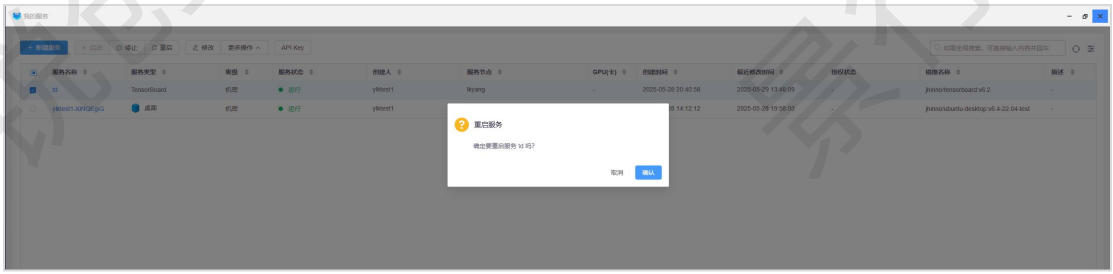
选中“运行”中的服务，点击“停止”按钮，弹出“是否停止此服务”提示窗口，点击“确定”按钮，即可停止该条服务实例。如下图所示：



停止服务

2.6.9 重启服务

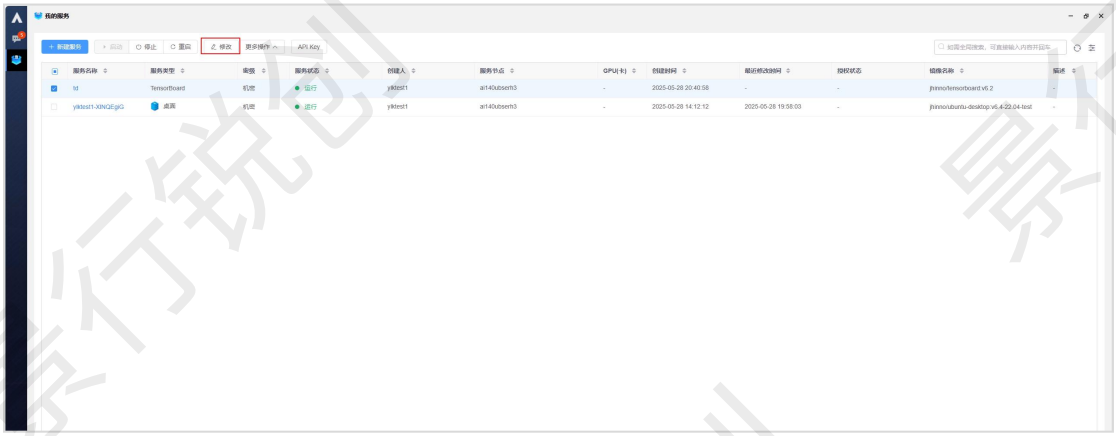
选中“运行”中且“非开发环境”启动的服务，点击“重启”按钮，弹出“是否重启此服务”提示窗口，点击“确定”按钮，即可重启该条服务实例。如下图所示：



重启服务

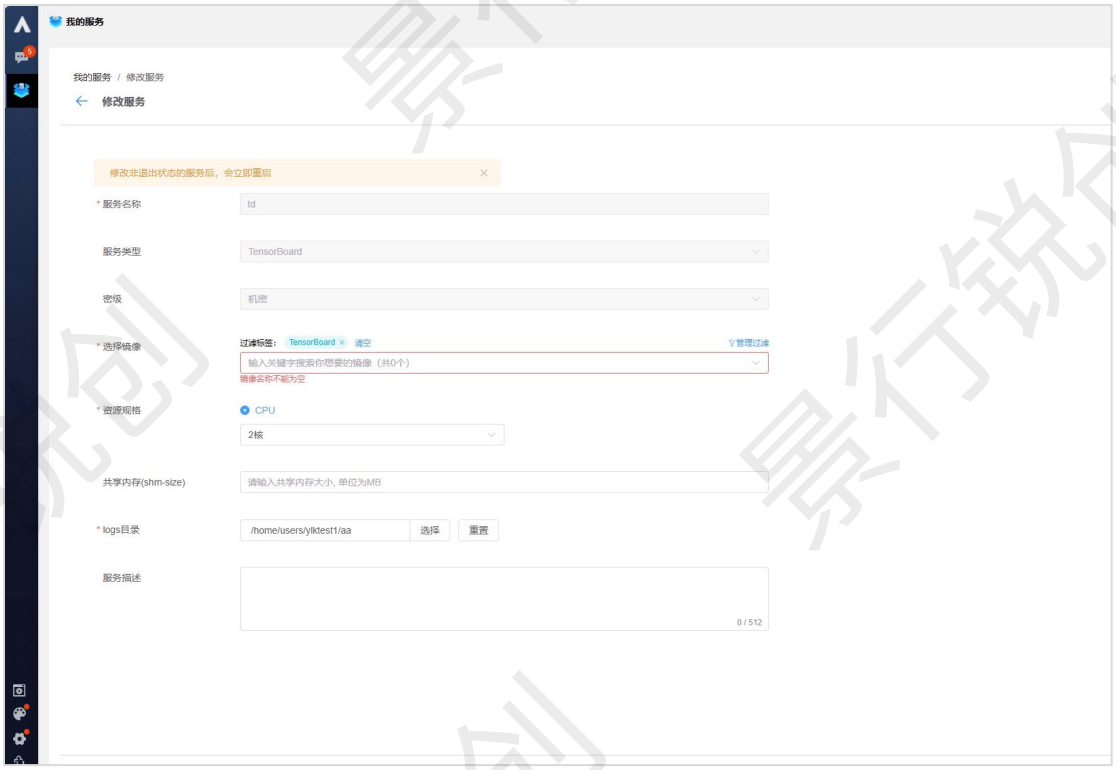
2.6.10 修改服务

选中在服务管理页面创建的服务（非开发环境启动的服务），点击“修改”按钮，修改服务窗口，重新设置服务参数，点击“确定”按钮，即可修改完成该条服务实例。当服务处于运行状态，修改后服务会按照最新参数重启；其他状态仅会修改服务参数，服务再次启动时，会按照最新参数启动。如下图所示：



修改服务入口

修改服务，除了服务名称、服务类型和密级之外的参数都可以修改，如下图所示：

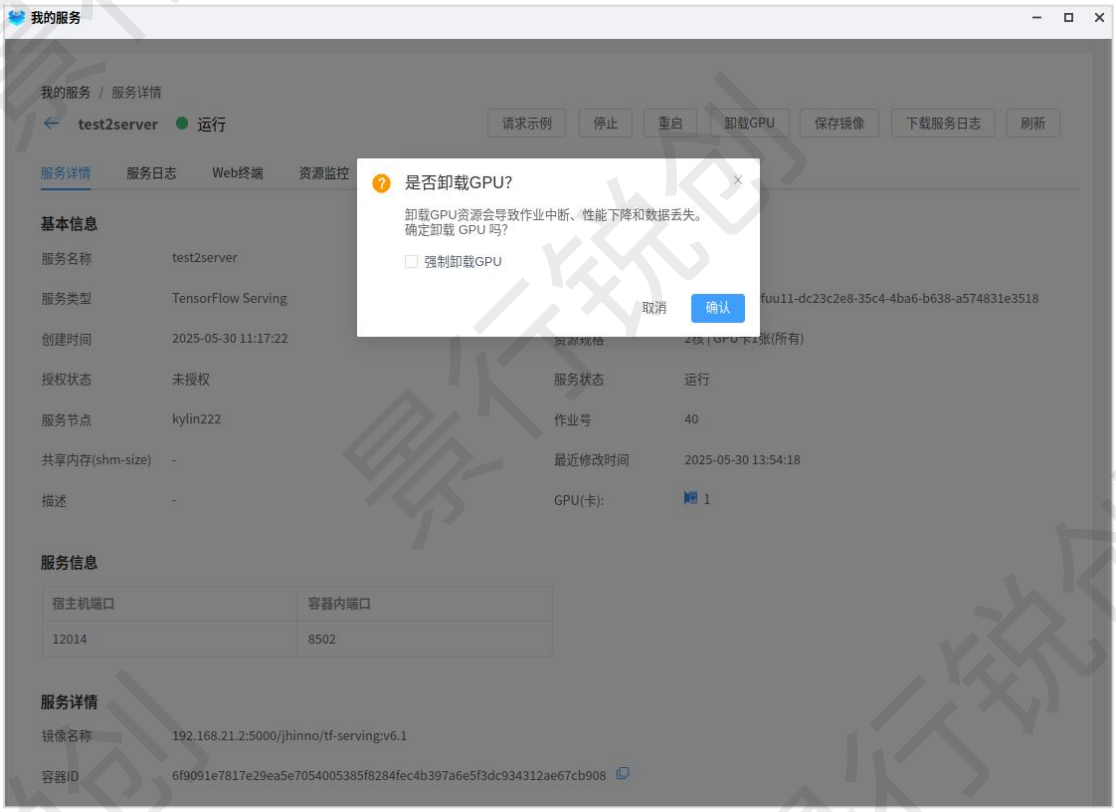


修改服务页面

参数修改完成后，点击“修改服务”按钮，即可完成服务修改。

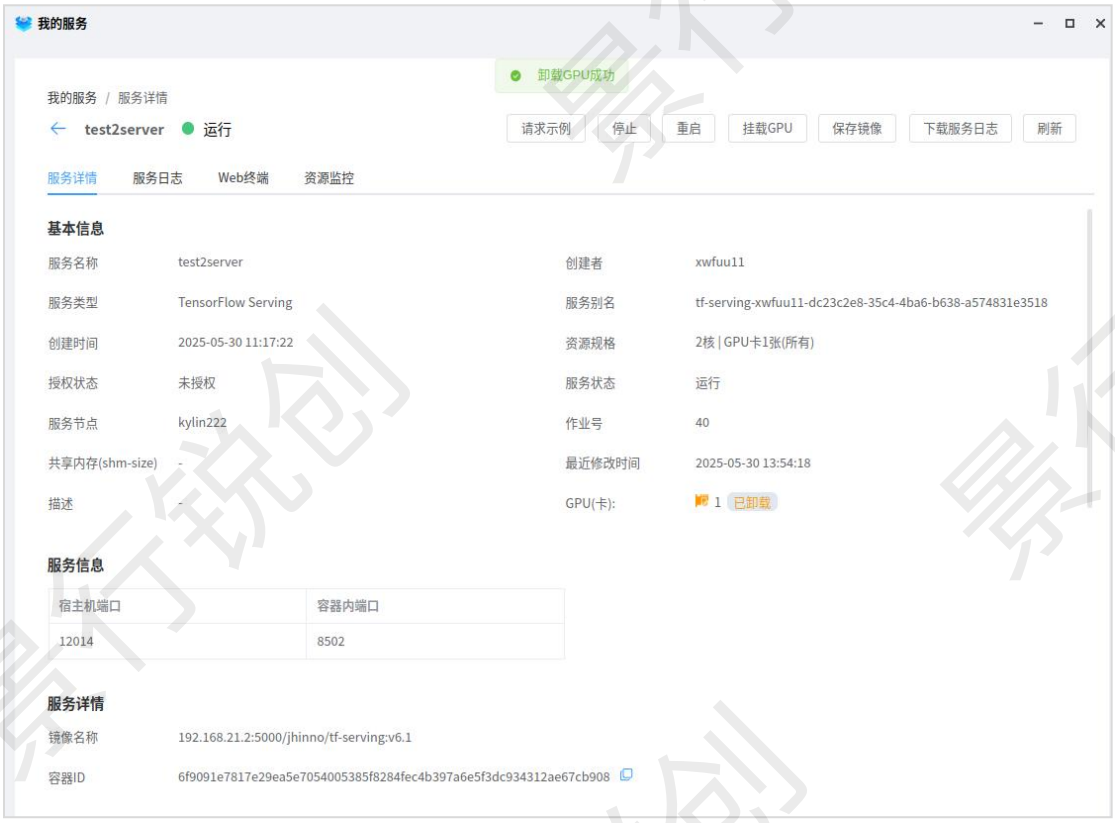
2.6.11 卸载 GPU

创建服务的过程中，选择 GPU 资源后，在服务列表中会看到 GPU 信息。点击自己创建选择 GPU 卡且运行中的服务，进入服务详情页面，在服务详情页面的右上角有“卸载 GPU”按钮。点击“卸载 GPU”按钮后，弹出“是否卸载 GPU”弹框，点击“确认”后，GPU 卸载成功，在服务列表和服务详情页面可以看到 GPU 处于卸载状态，如下图所示：



是否卸载 GPU

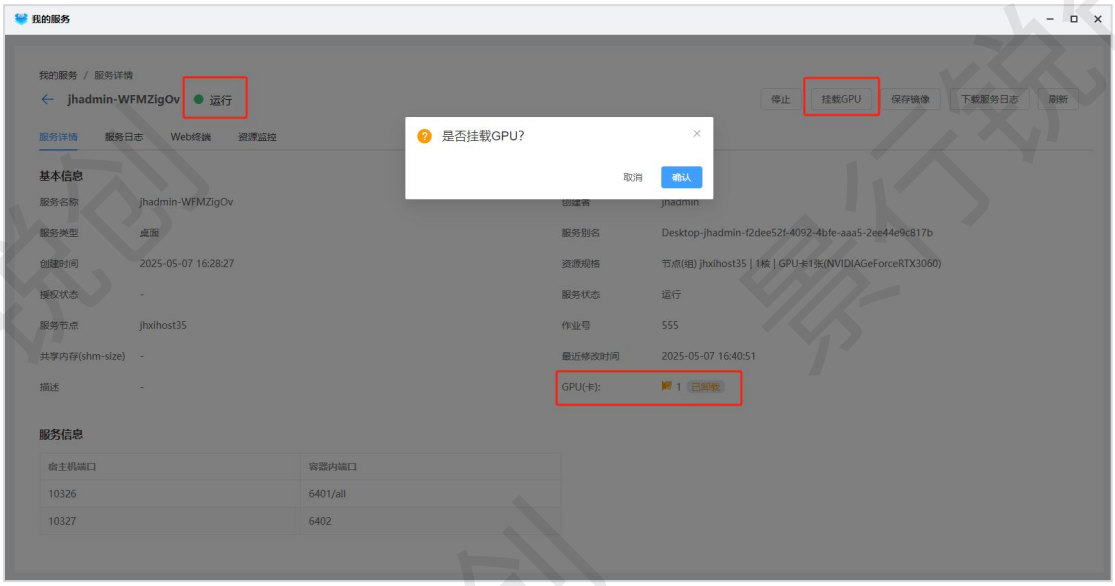
卸载完成后，会提示“卸载 GPU 成功”，并且 GPU（卡）属性区域提示“已卸载”，如下图所示：



卸载 GPU 成功的服务详情页面

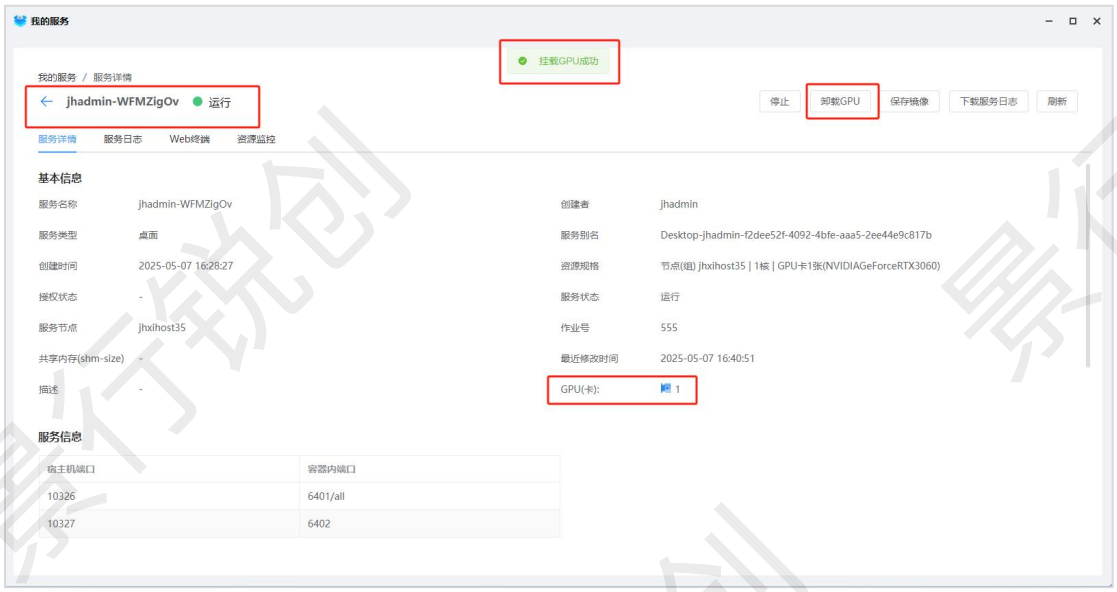
2.6.12 挂载 GPU

卸载 GPU 后运行的服务，可以选择“挂载 GPU”，为服务添加 GPU 资源。点击“挂载 GPU”，弹出对话框，确认是否挂载 GPU，点击“确认”后，容器服务重新加载到 GPU。



挂载 GPU 页面

挂载 GPU 成功的服务，服务状态为“运行”，“挂载 GPU”按钮变为“卸载 GPU”，GPU（卡）属性不再有“已卸载”关键字，如下图所示：

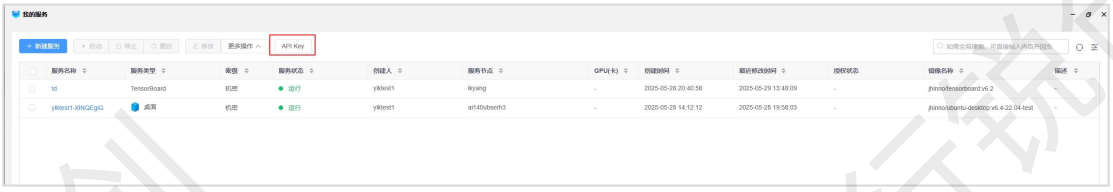


挂载 GPU 成功的服务详情页面

2.6.13 API Key

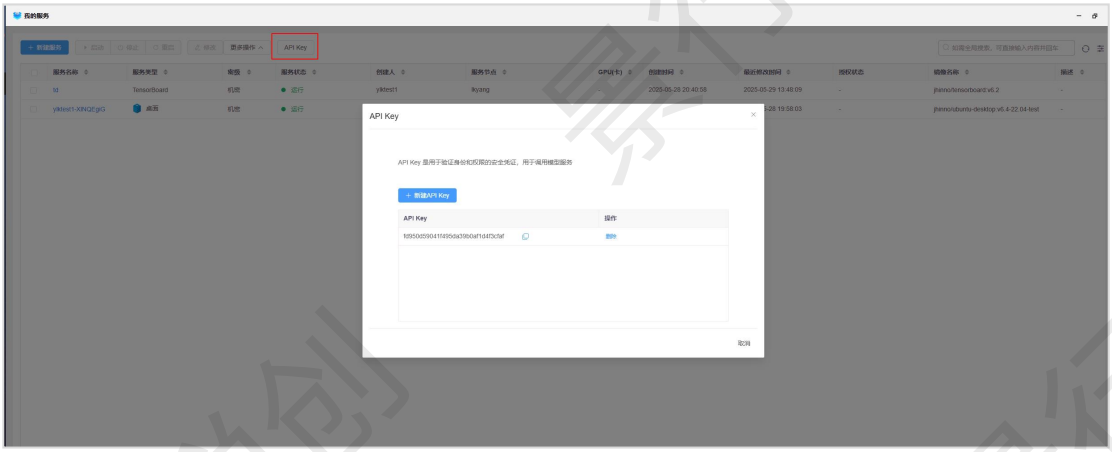
API Key 是用于验证身份和权限的安全凭证，用于调用模型服务。启动指定服务时，如果开启鉴权，则需要在调用服务时，添加此处生成的 API Key。

在我的服务页面，API Key 位于服务列表上方，如下图所示：



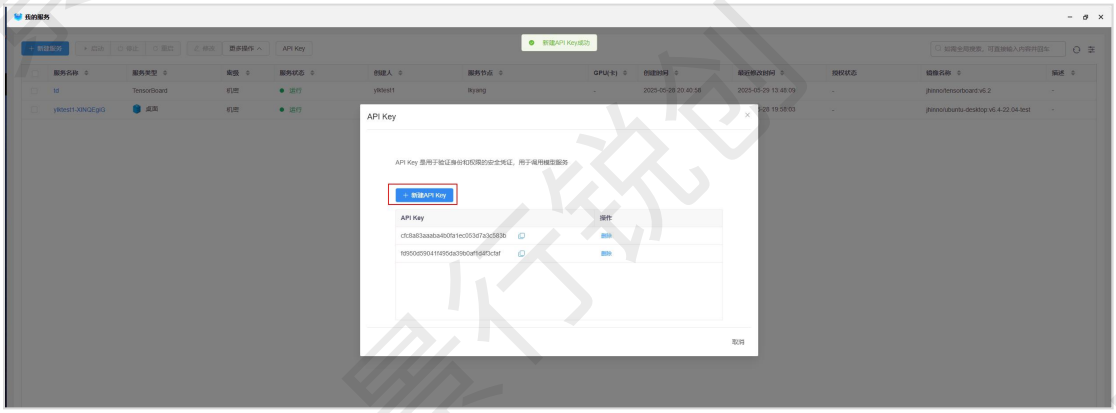
“API Key”入口

点击“API Key”进入 API Key 设置页面，新增或者删除 API Key，如下图所示：



API Key 设置页面

每个用户可以创建 5 个 API Key，在调用服务时，使用任意一个 Key 都可以。新增 API Key 只需点击“+新建 API Key”即可，如下图所示：



新建 API Key

不需要某个 Key 时，可以点击对应 Key 后的“删除”按钮，如下图所示：

删除 API Key

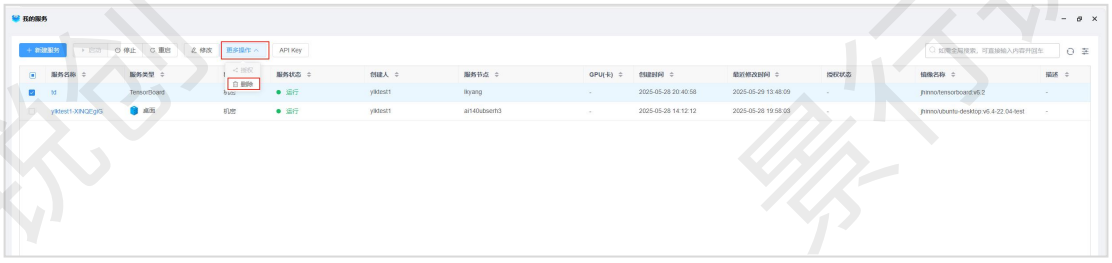
复制 key 可以通过点击复制按钮实现，如下图所示：

复制 API Key

2.6.14 更多操作

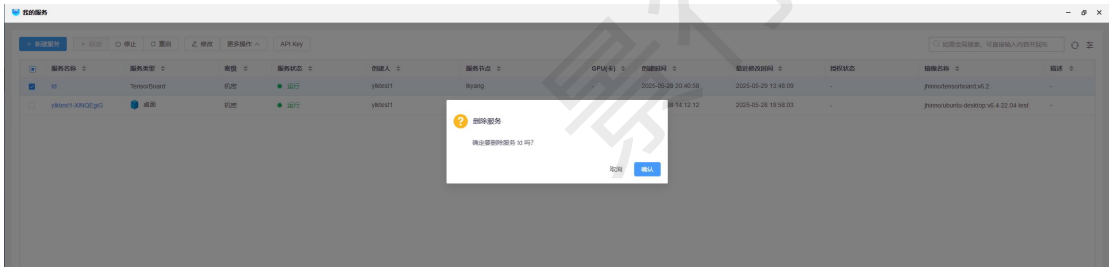
选中在服务管理页面服务，点击“更多操作”，弹出“授权”和“删除”按钮，所有服务均可被删除。

用户自己创建且运行中的“Pytorch Serving”“Tensorflow Serving”“Triton Inference Server”类型的服务实例可以“授权”。



更多操作

- 删除服务



删除服务

- 授权服务


我的服务
新建服务
停止
重启
修改
更多操作
API Key
如需全局搜索，可直接输入内容并回车
服务名称
服务类型
服务状态
创建人
服务节点
GPU(卡)
创建时间
最近修改时间
授权状态
授权
授权
部分人使用
所有人使用
选择用户
请输入
本单位(defaultDep)
test001(test001)
xwftest1(xwftest1)
xwftest3(xwftest3)
请输入
已选择：0
暂无数据

3. 附录一：方案设计组件属性说明

3.1 数据处理组件

3.1.1 数据提取

- Fluent 数据提取
  - Fluent 数据提取：Fluent 数据提取。
  - 选择规则文件：选择 rul 格式的数据文件。
  - 待提取数据：选择待提取数据。

- CFDpp 数据提取
  - 选择规则文件：选择 rul 格式的数据文件。
  - 待提取数据：选择待提取数据。

- 导入到 csv 数据集
  - 导入到 csv 数据集：导入到 csv 数据集。
  - 生成数据集的名称：生成数据集名称。

3.1.2 数据加载

- 数值数据
  - 数值数据：csvImport，csv 格式的数据导入。
  - 选择数据文件：选择 csv 或者 tsv 格式的数据文件。

- 图像数据
  - 图片数据：图像数据导入。
  - 图像文件：选择图像目录，图像需保存在同一个文件夹中。

- 数值数据数据集
  - 数值数据数据集：数值数据数据集。
  - 选择数据集：选择数值数据数据集。

- 图像数据集
  - 图像数据集：图像数据集。
  - 选择数据集：选择图像数据集。

- Tfrecord 数据集
  - Tfrecord 数据集：Tfrecord 数据集。
  - 选择数据集：选择 Tfrecord 数据集。
  - 自动获取样本数：布尔值，指定是否遍历 tfrecord 文件来获取样本个数，并写入 meta.json 文件中，如果 meta.json 中已经存在样本数记录则忽略此项。

- 自定义读取数据
  - 自定义读取数据：自定义读取数据。
  - 选择数据文件：选择数据文件。
  - 自定义函数名称：填写自定义读取数据的函数名称。
  - train_size：训练集大小，当该组件直连模型训练时，参数生效，默认为 100。
  - val_size：验证集大小，当该组件直连模型训练时，参数生效，默认为 100。
  - 自定义读取数据脚本：按照模板添加自定义读取数据脚本。

3.1.3 数据预处理

- 选择特征
  - 选择特征：选择样本特征和标签特征。
  - 样本特征：选择样本特征作为模型输入。
  - 样本标签：选择样本标签作为模型输出。

- 输入特征
  - 输入特征：输入样本特征和标签特征。

- 样本特征：样本特征列作为模型输入，如[1,2,3]选择第二列，第三列和第四列或者[1-4]选择第二列到第四列或者[1]表示选择第二列，当参数为 all 时，表示选择所有列，默认 all。
- 标签特征：样本标签列作为模型标签输入，如[1,2,3]选择第二列，第三列和第四列或者[1:4]选择第二列到第四列或者[1]表示选择第二列，当参数为 all 时，表示选择所有列，None 表示不选，默认 None。

- 数据分割
  - 数据分割：实现训练集、测试集和验证集的随机分割。
  - 训练/验证集/测试：数据集划分，按所选比例划分为训练集、测试集和验证集。

- 保存数据
  - 保存数据：保存生成的数据。
  - 生成数据集的名称：生成数据集名称。
  - 数据集类型：保存数据集的类型。

- 自定义处理数据
  - 自定义数据处理：自定义处理数据。
  - 自定义处理数据脚本：按照模板添加自定义处理数据脚本。
  - 自定义函数名称：填写自定义处理数据的函数名称。

3.1.4 数值数据集预处理

- 丢弃缺失值
  - 丢弃缺失值：丢弃缺失值所在的行或者列。
  - column index：输入行列表，如[0,1]，默认为 None，当 axis 设置为 1 时该参数生效，否则无效，默认为 None。
  - axis：默认 axis=0。0 为按行删除，1 为按列删除。
  - how：选择删除规则，默认 any.'any'： 只要含有 NA，就舍去该行/

列;'all'： 只有该行/列均为NA时才舍去。

➢ thresh：指定行/列具有非NA的数目，即至少具有thresh个非NA值时才进行保留。

- 填充缺失值
  - ➢ 填充缺失值：填充缺失值。
  - ➢ value：填充值。
  - ➢ method：填充缺失值的方法，当为'ffill'，表示用前面的值填充，当'bfill'表示用后面的值填充。
  - ➢ axis：非负整数，连接轴，不包括批处理轴，默认0.0 表示'index'，1表示'columns'。
  - ➢ limit：如果指定了方法，则这是连续的NaN值的前向/后向填充的最大数量。换句话说，如果连续NaN数量超过这个数字，它将只被部分填充。如果未指定方法，则这是沿着整个轴的最大数量，其中NaN将被填充。如果不是无，则必须大于0。
  - ➢ downcast：尝试向下转换为适当的相等类型的字符串。

- 删除重复数据
  - ➢ 删除重复数据：删除重复的数据。
  - ➢ keep：删除重复数据的规则。first：保留第一次出现的重复行，删除后面的重复行。last： 删除重复项，除了最后一次出现。False：删除所有重复项。

- 数据标准化
  - ➢ 数据标准化：数据标准化。
  - ➢ method：选择数据标准化方法。

- 删除数据
  - ➢ 删除数据：删除指定行或列数据。

- labels：输入删除的行列的名字，输入列表或者 None，列表可以是[1,2]或者['a','b']。
- axis：指定删除行或者列，0 指删除行，1 表示删除列，默认 0。
- index:输入行列表或者行号,如 0 或者[0,1]或者 None,默认为 None，当 axis 设置为 1 时该参数生效，否则无效。
- level：针对多重索引时的优先级，默认为 None。

- 数据排序
  - 数据排序：根据指定列数据也可根据指定行的数据排序。
  - column index：输入行列表或者行号，如 0 或者[0,1]，默认为 0，当 axis 设置为 1 时该参数生效，否则无效。
  - axis：按照行或者列排序，0 指按照指定列排序，1 指按照指定行排序，默认 0。
  - ascending：是否按指定列的数组升序排列，默认为 True，即升序排列。
  - na_position：设定缺失值的显示位置。

- 数据合并
  - 数据合并：根据列索引或者索引列表对数据合并。
  - how：数据的拼接方式，inner。
  - left_on：左侧 DataFrame 中的列或索引级别用作键。可以是列名，索引级名称，也可以是长度等于 DataFrame 长度的数组。
  - right_on：右侧 DataFrame 中的列或索引级别用作键。可以是列名，索引级名称，也可以是长度等于 DataFrame 长度的数组。
  - sort：按字典顺序通过连接键对结果 DataFrame 进行排序，默认为 True。
  - suffixes：用于重叠列的字符串后缀元组。 默认为('x','y')。
  - indicator：将一列添加到名为_merge 的输出 DataFrame，其中包含有关每行源的信息。

- 特征编码
  - 特征编码：对选择的数据进行编码，分为哑编码和独热编码。
  - how：OrdinalEncoder 为哑编码，能够将分类特征转换为分类数值。OneHotEncoder 为独热编码，将每一个分类特征变量的 m 个可能的取值转变成 m 个二值特征，对于每一条数据这 m 个值中仅有一个特征值为 1，其他的都为 0；LabelEncoder，可以将数据(类别型或者数值型都可以)转换为一个整数值。

- 标签二值化
  - 标签二值化：对选择的标签数据进行二值化，分单标签和多标签。
  - how：LabelBinarizer，多类别单标签数据二值化；MultiLabelBinarizer，多类别多标签数值二值化。

- 通过复制原始数据并添加高斯分布的噪音来进行数据增强
  - 数值数据增强：自定义组件别名。
  - 是否包含原始数据：数据增强后是否包含原始数据，默认为 True。
  - 增强倍数：数据增强的倍数。
  - 标准差：高斯噪音的标准差。
  - 是否根据每列范围调整：是否根据当列的数据范围调整当列的噪音的标准差，默认为 False。

3.1.5 图像数据集预处理

3.1.5.1 图像预处理

- 图像缩放
  - 图像缩放：图像缩放。
  - dsize：非负整数组成的元组，例如，shape=(32，32)表示图像缩放到 32*32，默认(32,32)。
  - fx：输入非负浮点数，沿水平轴缩放的比例因子，默认 0。

- fy：输入非负浮点数，沿垂直轴缩放的比例因子，默认 0。
- interpolation：设置为 resize 的插值方式，默认为双线性插值。

- 均值滤波
  - 均值滤波：均值滤波。
  - ksize：正整数组成的元组，例如，shape=(3，3)表示滤波器核大小为 3*3，默认(3,3)。
  - anchor：锚点，被平滑的那个点，如果这个点坐标是负值的话，就表示取核的中心为锚点，所以默认值 Point(-1,-1)表示这个锚点在核的中心，默认值 Point(-1,-1)。
  - borderType：用于推断图像外部像素的某种边界模式，默认值 BORDER_DEFAULT。

- 方框滤波
  - 方框滤波：方框滤波。
  - ddepth：输入整数，输出图像深度，默认-1。
  - ksize：正整数组成的元组，例如，shape=(3，3)表示滤波器核大小为 3*3，默认(3,3)。
  - anchor：锚点，被平滑的那个点，如果这个点坐标是负值的话，就表示取核的中心为锚点，所以默认值 Point(-1,-1)表示这个锚点在核的中心，默认值 Point(-1,-1)。
  - normalize：内核是否按其面积进行标准化。
  - borderType：用于推断图像外部像素的某种边界模式，默认值 BORDER_DEFAULT。

- 高斯滤波
  - 高斯滤波：高斯滤波。
  - ksize：正整数组成的元组，例如，shape=(3，3)表示滤波器核大小为 3*3，默认(3,3)。

- sigmaX：X 方向上的高斯核标准偏差，默认值 0。
- sigmaY：Y 方向上的高斯核标准偏差，默认值 0。
- borderType：用于推断图像外部像素的某种边界模式，默认值 BORDER_DEFAULT。

- 中值滤波
  - 中值滤波：中值滤波。
  - ksize：线性直径大小，例如：3, 5, 7，默认 3。

- 双边滤波
  - 双边滤波：双边滤波。
  - d：过滤过程中每个像素领域的直径范围。
  - sigmaColor：颜色空间过滤器的 sigma 值，浮点数。
  - sigmaSpace：坐标空间中滤波器的 sigma 值，浮点数。
  - borderType：用于推断图像外部像素的某种边界模式，默认值 BORDER_DEFAULT。

- 2D 卷积
  - 2D 卷积：2D 卷积。
  - ddepth：输入整数，输出图像深度，默认-1。
  - kernel：正整数组成的元组，例如，shape=(3，3)表示滤波器核大小为 3*3，默认(3,3)。
  - anchor：锚点，被平滑的那个点，如果这个点坐标是负值的话，就表示取核的中心为锚点，所以默认值 Point(-1,-1)表示这个锚点在核的中心，默认值 Point(-1,-1)。
  - delta：输入浮点数，选填，滤波后的像素添加固定参数，默认 0。
  - borderType：用于推断图像外部像素的某种边界模式，默认值 BORDER_DEFAULT。

- 图像裁剪
  - 图像裁剪：图像裁剪。
  - rect：输入字符串，字符串由','分开，分开后的字符串格式应为 a-b 型，a 与 b 均为非负整数，例如，'0-32,0-32'表示裁剪区域为[0-32,0-32]也就是[y0-y1,x0-x1]。

- 全局二值化
  - 全局二值化：全局二值化。
  - thresh：输入浮点数，指用来对像素值进行分类的阈值，默认 0。
  - maxval：输入浮点数，指当像素值高于（有时是小于）阈值时应该被赋予的新的像素值，默认 0。
  - thresholdType：设置为阈值方法，默认为 cv.INTER_NEAREST(线性插值)。

- 局部二值化
  - 局部二值化：局部二值化。
  - maxValue：输入浮点数，阈值的最大值，默认 0。
  - adaptiveMethod：在一个邻域内计算阈值所采用的算法，有两个取值，分别为 ADAPTIVE_THRESH_MEAN_C 和 ADAPTIVE_THRESH_GAUSSIAN_C。
  - thresholdType：设置为阈值方法，默认为 cv.INTER_NEAREST(线性插值)。
  - blockSize：adaptiveThreshold 的计算单位是像素的邻域块，这是局部邻域大小，3、5、7 等，非负整数，默认 3。
  - C：一个偏移值调整量，用均值和高斯计算阈值后，再减或加这个值就是最终阈值，浮点数，默认 0。

- 颜色空间转换
  - 颜色空间转换：图像颜色空间转换。
  - code：指定颜色空间转换类型，默认为 cv2.COLOR_BGR2RGB。

- 图像颜色空间分离
  - 图像颜色空间分离：图像颜色空间分离。

3.1.5.2 分类图像数据增强

- 文件名标注数据集
  - 文件名标注数据集：文件名标注数据集。
  - featurewise_center：如果为 False，将输入数据的均值设置为 0，逐特征进行，默认'False'。
  - samplewise_center：如果为 False，将每个样本的均值设置为 0，逐特征进行，默认'False'。
  - featurewise_std_normalization：如果为 False，将输入除以数据标准差，逐特征进行，默认'False'。
  - samplewise_std_normalization:如果为 False，将每个输入除以其标准差，逐特征进行，默认'False'。
  - zca_epsilon：ZCA 白化的 epsilon 值，默认为 1e-6。
  - zca_whitening：是否应用 ZCA 白化，默认'False'。
  - rotation_range：整数，随机旋转的度数范围，默认 0。
  - width_shift_range：整数，随机旋转的度数范围，默认 0;float：如果 <1，则是除以总宽度的值，或者如果 >=1，则为像素值;1-D 数组：数组中的随机元素;int：来自间隔 (-width_shift_range, +width_shift_range) 之间的整数个像素;width_shift_range=2 时，可能值是整数 [-1, 0, +1]，与 width_shift_range=[-1, 0, +1] 相同；而 width_shift_range=1.0 时，可能值是 [-1.0, +1.0) 之间的浮点数。
  - height_shift_range：整数，随机旋转的度数范围，默认 0;float：如果 <1，则是除以总宽度的值，或者如果 >=1，则为像素值；1-D 数组：数组中的随机元素；int：来自间隔 (-width_shift_range, +width_shift_range) 之间的整数个像素；width_shift_range=2 时，可能值是整数 [-1, 0, +1]，与 width_shift_range=[-1, 0, +1] 相同；而

width_shift_range=1.0 时，可能值是 [-1.0, +1.0) 之间的浮点数。

- shear_range：浮点数。剪切强度（以弧度逆时针方向剪切角度），默认 0.0。
- zoom_range：浮点数 或 [lower, upper]。随机缩放范围。如果是浮点数，[lower, upper] = [1-zoom_range, 1+zoom_range]，默认 0.0。
- channel_shift_range：浮点数。随机通道转换的范围，默认 0.0。
- fill_mode：'constant', 'nearest', 'reflect' or 'wrap' 之一。默认为 'nearest'。输入边界以外的点根据给定的模式填充。
- cval：浮点数或整数。用于边界之外的点的值，当 fill_mode ='constant' 时，默认 0.0。
- horizontal_flip：是否随机水平翻转，默认'False'。
- vertical_flip：是否随机垂直翻转，默认'False'。
- rescale：如果是 None 或 0，不进行缩放，否则将数据乘以所提供的值（在应用任何其他转换之前），默认 None。
- preprocessing_function：应用于每个输入的函数。这个函数会在任何其他改变之前运行。这个函数需要一个参数：一张图像（秩为 3 的 Numpy 张量），并且应该输出一个同尺寸的 Numpy 张量，默认 None。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, height, weight, ..., channels），channels_first 对应输入形状为(batch, channels, height, weight, ...)，默认 channels_last。
- validation_split:浮点数。Float. 保留用于验证的图像的比例(严格在 0 和 1 之间)，默认 0.0。
- dtype：生成数组使用的数据类型，默认 None。
- batch_size：正整数，默认 32。
- shuffle：是否随机打乱数据，默认为 True。
- seed：整数或者 None，默认 None。
- save_to_dir：None 或 字符串（默认 None）。这使你可以最佳地指定正在生成的增强图片要保存的目录，默认 None。

- save_prefix：字符串。 保存图片的文件名前缀（仅当 save_to_dir 设置时可用）。
- save_format：'png'或'jpeg'之一，指定保存图片的数据格式，默认'jpeg'。

- 文件夹名标注数据集
  - 文件夹名标注数据集：文件夹名标注数据集，每个类应该包含一个子目录。任何在子目录树下的 PNG, JPG, BMP, PPM 或 TIF 图像，都将被包含在生成器中。
  - featurewise_center：如果为 False，将输入数据的均值设置为 0，逐特征进行，默认'False'。
  - samplewise_center：如果为 False，将每个样本的均值设置为 0，逐特征进行，默认'False'。
  - featurewise_std_normalization：如果为 False，将输入除以数据标准差，逐特征进行，默认'False'。
  - samplewise_std_normalization:如果为 False，将每个输入除以其标准差，逐特征进行，默认'False'。
  - zca_epsilon：ZCA 白化的 epsilon 值，默认为 1e-6。
  - zca_whitening：是否应用 ZCA 白化，默认'False'。
  - rotation_range：整数，随机旋转的度数范围，默认 0。
  - width_shift_range： 整数，随机旋转的度数范围，默认 0;float：如果 <1，则是除以总宽度的值，或者如果 >=1，则为像素值；1-D 数组： 数组中的随机元素；int： 来自间隔 (-width_shift_range, +width_shift_range) 之间的整数个像素；width_shift_range=2 时，可能值是整数 [-1, 0, +1]，与 width_shift_range=[-1, 0, +1] 相同；而 width_shift_range=1.0 时，可能值是 [-1.0, +1.0) 之间的浮点数。
  - height_shift_range： 整数，随机旋转的度数范围，默认 0;float：如果 <1，则是除以总宽度的值，或者如果 >=1，则为像素值；1-D 数组： 数组中的随机元素；int： 来自间隔 (-width_shift_range,

+width_shift_range）之间的整数个像素；width_shift_range=2 时，可能值是整数 [-1, 0, +1]，与 width_shift_range=[-1, 0, +1] 相同；而 width_shift_range=1.0 时，可能值是 [-1.0, +1.0）之间的浮点数。

- shear_range：浮点数。剪切强度（以弧度逆时针方向剪切角度），默认 0.0。
- zoom_range：浮点数 或 [lower, upper]。随机缩放范围。如果是浮点数，[lower, upper] = [1-zoom_range, 1+zoom_range]，默认 0.0。
- channel_shift_range：浮点数。随机通道转换的范围，默认 0.0。
- fill_mode：'constant', 'nearest', 'reflect' or 'wrap' 之一。默认为 'nearest'。输入边界以外的点根据给定的模式填充。
- cval：浮点数或整数。用于边界之外的点的值，当 fill_mode ='constant' 时，默认 0.0。
- horizontal_flip：是否随机水平翻转，默认'False'。
- vertical_flip：是否随机垂直翻转，默认'False'。
- rescale：如果是 None 或 0，不进行缩放，否则将数据乘以所提供的值（在应用任何其他转换之前），默认 None。
- preprocessing_function：应用于每个输入的函数。这个函数会在任何其他改变之前运行。这个函数需要一个参数：一张图像（秩为 3 的 Numpy 张量），并且应该输出一个同尺寸的 Numpy 张量，默认 None。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, height, weight, ..., channels），channels_first 对应输入形状为(batch, channels, height, weight, ...)，默认 channels_last。
- validation_split:浮点数。Float. 保留用于验证的图像的比例(严格在 0 和 1 之间），默认 0.0。
- dtype：生成数组使用的数据类型，默认 None。
- target_size：整数元组 (height, width)，所有找到的图都会调整到这个维度，默认为 (256, 256)。
- color_mode：'grayscale', 'rbg' 之一，图像是否转换为 1 个或 3

个颜色通道，默认：'rgb'。

- classes：可选的类别列表（例如，['dogs','cats']）。默认：None。如未提供，类比列表将自动从 y_col 中推理出来，y_col 将会被映射为类别索引）。包含从类名到类索引的映射的字典可以通过属性 class_indices 获得。
- class_mode： 'categorical', 'binary', 'sparse', 'input', 'other' or None 之一。 默认：'categorical'。决定返回标签数组的类型：'categorical' 将是 2D one-hot 编码标签；'binary'将是 1D 二进制标签；'sparse'将是 1D 整数标签；'input' 将是与输入图像相同的图像（主要用于与自动编码器一起使用）;'other'将是 y_col 数据的 numpy 数组；None，不返回任何标签。
- batch_size：非负整数，一批数据的大小，默认 32。
- shuffle：是否随机打乱数据，默认为 True。
- seed：整数或者 None，可选的混洗和转换的随机种子，默认 None;
- save_to_dir：None 或 字符串（默认 None）。这使你可以最佳地指定正在生成的增强图片要保存的目录，默认 None。
- save_prefix：字符串。 保存图片的文件名前缀（仅当 save_to_dir 设置时可用）。
- save_format：'png'或'jpeg'之一，指定保存图片的数据格式，默认'png'。
- subset：如果在 ImageDataGenerator 中设置了 validation_split，则为 training 或者 validation 的子集，默认 None。
- interpolation：在目标大小与加载图像的大小不同时，用于重新采样图像的插值方法，支持的方法有'nearest','bilinear', and 'bicubic'，默认情况下，使用 'nearest'。

3.1.5.3 目标检测数据集增强

- 随机反转图像
  - 随机反转图像：在水平方向上随机反转图像，使用概率 p 对图像数

据进行反转。

- p：输入浮点数，图像翻转概率，默认 0.5。

- 随机缩放图像

  - 随机缩放图像：在保持纵横比不变的情况下，缩放图像。丢弃变换图像中剩余面积小于 25% 的边界框。 分辨率保持不变，用黑色填充剩余区域。

  - scale：图像缩放因子，从范围（1-scale，1+scale）中随机抽取图像的缩放因子，默认 0.2。

- 随机平移图像

  - 随机平移图像：随机平移图像，删除变换图像中剩余区域中面积小于 25%的边界框，保持分辨率不变，如果存在剩余区域则用黑色填充。

  - translate：输入 0 到 1 之间的整数或者浮点数，图像从范围（1-translate，1+translate）中随机抽取的因子进行平移，默认 0.2。

- 随机旋转图像

  - 随机旋转图像：随机旋转图像。

  - angle：输入正整数，图像旋转范围从（-angle，angle）中随机抽取的因子进行旋转，默认 10。

- 随机裁剪图像

  - 随机裁剪图像：在水平方向上，随机裁剪图像。

  - shear_factor：输入 0 到 1 之间的整数或者浮点数，定义随机水平裁剪图像因子，默认 0.2。

3.2 模型设计组件

- 自定义模型

  - 自定义模型：自定义模型结构。

- 自定义模型脚本：按照模板添加自定义模型脚本。

- 模型训练

- 模型训练：开始训练模型，包括深度学习组件拖拽生成的模型，自定义模型和深度学习模型增量训练。
- 模型名称：保存模型的名称。
- 迭代次数：训练模型的迭代次数，默认为100。
- 批次大小：一次训练所选取的样本数。
- 损失函数：模型训练的损失函数，暂时不支持Reduction函数。
- metrics：metrics函数，暂时不支持MeanRelativeError，MeanIoU,PrecisionAtRecall，RecallAtPrecision，SensitivityAtSpecificity和SpecificityAtSensitivity。
- 优化器：设置模型优化器。
- 学习率调整策略：设置学习率调整策略：None表示不设置任何学习策略，学习率为固定值；CosineDecay，余弦衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),alpha(最小学习率值作为initial_learning_rate的一部分);CosineDecayRestarts，包含重新启动的余弦衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),first_decay_steps(要衰减的步数),t_mul(用于导出i-th周期内的迭代次数),m_mul(用于导出i-th周期的初始学习率),alpha(最小学习率值作为initial_learning_rate的一部分);ExponentialDecay，指数衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),decay_rate(衰减率),staircase(如果True以离散间隔衰减学习率);InverseTimeDecay，反时间衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),decay_rate(衰减率),staircase(是否在离散梯度中使用衰减，而不是在连续梯度);PiecewiseConstantDecay，分段常数衰减策略，与该策略相关的参数有：boundaries(Tensors或ints或floats的列表具有严格增加的

条目,并且所有元素具有与优化器步骤相同的类型),values(Tensor 或 float 或 int 列表，指定由 boundaries 定义的间隔的值，它应该比 boundaries 多一个元素，并且所有元素都应该具有相同的类型);PolynomialDecay，多项式学习能力衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),end_learning_rate(最小的最终学习率),power(多项式的幂),cycle(是否应该循环超过 decay_steps)。

- learning_rate:设置固定学习率,0 到 1 之间的浮点数,默认 0.001。
- initial_learning_rate：初始学习率，0 到 1 之间的浮点数，默认 0.1。
- decay_steps：衰减步数，正整数，默认 1。
- alpha：最小学习率值作为 initial_learning_rate 的一部分，0 到 1 之间的浮点数，默认 0.0。
- first_decay_steps：衰减步数，正整数，默认 1。
- t_mul：用于导出 i-th 周期内的迭代次数，非负浮点数，默认 2.0。
- m_mul：用于导出 i-th 周期的初始学习率，非负浮点数，默认 1.0。
- decay_rate：衰减率，非负浮点数，默认 0.5。
- staircase：是否在离散梯度中使用衰减，而不是在连续梯度，默认'False'。
- boundaries：严格递增的由浮点型或者整型组成的列表，默认[100000,110000]。
- values：指定由 boundaries 定义的间隔值，它应该比 boundaries 多一个元素，并且所有元素都应该具有相同的类型，默认[1.0,0.5,0.1]。
- end_learning_rate：最小的最终学习率，非负浮点数，默认 0.5。
- power：多项式的幂，非负浮点数，默认 1.0。
- cycle：是否应该循环超过 decay_steps，默认'False'。
- rho:Adadelta 梯度平方移动均值的衰减率,非负浮点数,默认 0.95。
- epsilon：epsilon，防止除 0 错误，0 到 1 之间的浮点数或者科学计数法，默认 1e-07。
- initial_accumulator_value：初始梯度累加和，非负浮点数，默认

0.1。

- beta_1：0 到 1 之间，非常接近 1 的浮点数，默认 0.9。
- beta_2：0 到 1 之间，非常接近 1 的浮点数，默认 0.999。
- amsgrad：是否应用此算法的 AMSGrad 变种，默认'False'。
- centered：如果 True，则通过梯度的估计方差对梯度进行归一化；如果为 False，则通过非居中的第二时刻。将此设置为 True 可能有助于训练，但在计算和内存方面稍微贵一些，默认'False'。
- momentum：用于加速 SGD 在相关方向上前进，并抑制震荡，非负浮点数，默认 0.0。
- nesterov：是否使用 Nesterov 动量，默认'False'。
- beta_2：浮点值，必须小于或等于零，控制在训练期间学习率如何降低，使用零表示固定的学习率，默认-0.5。
- l1_regularization_strength：浮点值，必须大于或等于零。默认为 0.0。
- l2_regularization_strength：浮点值，必须大于或等于零。默认为 0.0。
- l2_shrinkage_regularization_strength：浮点值，必须大于或等于零。这与上面的 L2 不同，因为上面的 L2 是一个稳定惩罚，而这个 L2 收缩是一个幅度惩罚。当输入稀疏时，收缩只会发生在活动权重上。默认为 0.0。
- beta：浮点值，beta 值。默认为 0.0。

3.2.1 深度学习

- Sequential
  - Sequential：指定模型框架，只能创建顺序结构的模型。

- Input
  - Input：指定模型框架，利用函数 API 创建模型，可以创建复杂模型，Input 用于实例化 keras 张量。
  - shape：正整数组成的元组或 None，例如，shape=(32，)表示预期的

输入将是一批 32 维向量；“None”表示形状未知的维度，默认 None。

- batch_size：正整数，可选的静态批处理大小，默认 None。
- name：字符串，层的可选名称。在模型中应该是唯一的(不要重复使用相同的名称)。如果没有提供，它将自动生成，默认 None。
- dtype：输入所期望的数据类型，默认 None。
- sparse：布尔值，指定要创建的占位符是否稀疏，默认 False。
- tensor：可选地将现有张量封装到输入层中。如果设置，该层将不会创建占位张量。默认 None。
- ragged：布尔值，指定要创建的占位符是否不规则，默认 False。

- Dense
  - Dense：Dense 层，又称全连接层，作为第一层时需要指定输入形状，以对接输入数据的形状。
  - Units：正整数，定义该层神经元数，即输出维度，如 32。
  - Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
  - Input Shape：正整数组成的元组或 None。当 Dense 作为模型第一层时，需要指定输入维度，如(12,)表示输入维度为 12，非第一层时，输入 None，默认为 None。
  - Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
  - Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
  - Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
  - Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

- Activation

  - Activation：Activation 层，指定激活函数，并将激活函数应用于输出。
  - Activation：选择激活函数，默认 relu。

- Dropout

  - Dropout：Dropout 层，作用于输入，在训练期间每次更新时将输入单元的分数率随机设置为 0，有助于防止过拟合。
  - Dropout Rate：输入 0 到 1 之间的整数或者浮点数，定义输入数据需要丢弃的比例，默认 0.0。
  - noise_shape：1D 整数张量， 表示将与输入相乘的二进制 dropout 掩层的形状，默认为 None。
  - seed：一个作为随机种子的 Python 整数，默认为 None。

- Reshape

  - Reshape：Reshape 层，将输出重新塑造为特定的形状。
  - Target Shape：输入整数元组，将数据转换为特定的形状，默认(3, 4)。
  - Input Shape：正整数组成的元组或 None。当 Reshape 作为模型第一层时，需要指定输入维度，如（12,），表示输入维度为 12，非第一层时，输入 None，默认为 None。

- LeakyReLU

  - LeakyReLU：LeakyReLU 激活函数层，作用是给所有负值赋予一个非零斜率。
  - alpha：非负的浮点数，定义负斜率系数，默认 0.3。

- Embedding

  - Embedding：Embedding 层，又称嵌入层，将正整数(索引)转换为固定大小的稠密向量。
  - Input Dim：正整数，输入文本词汇的大小，如 1000。
  - Output Dim：正整数，指定嵌入层的维度，如 64。
  - Embeddings Initializer：选择权值初始化方法，默认'glorot_uniform'。
  - Embeddings Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Embeddings Constraint：选择权值矩阵的约束（限制）函数，默认 None，暂时只支持 None。
  - Mask Zero：输入值 0 是否为被屏蔽的特殊“填充”值。
  - Input Length：正整数，输入序列的长度，如果要连接 Flatten 或 Dense 层则需要设置该参数为常数，如 100。

- Masking

  - Masking：Masking 层，对于输入张量的每一个时间步（张量的第一个维度），如果所有时间步中输入张量的值与 mask_value 相等，那么这个时间步将在所有下游层被覆盖(跳过)（只要它们支持覆盖）。
  - Target Shape：输入浮点数，mask value，默认 0.0。
  - Input Shape：正整数组成的元组或 None。当 Reshape 作为模型第一层时，需要指定输入维度，如（12,），表示输入维度为 12，非第一层时，输入 None，默认为 None。

- Lambda

  - Lambda：Lambda 层，将任意表达式封装为 Layer 对象。
  - function：需要封装的函数，将输入张量作为第一个参数。

- output_shape：预期的函数输出尺寸，只在使用 Theano 时有意义，可以是元组或者函数。 如果是元组，它只指定第一个维度；样本维度假设与输入相同：output_shape = (input_shape[0],) + output_shape 或者，输入是 None 且样本维度也是 None： output_shape = (None,) + output_shape 如果是函数，它指定整个尺寸为输入尺寸的一个函数：output_shape = f(input_shape)。
- arguments：可选的，需要传递给函数的关键字参数。
- 函数定义：添加函数定义。

● Conv1D

- Conv1D：Conv1D 层，一维卷积层或时间卷积层，该层创建了一个卷积核，与输入层在单个空间(或时间)维度上进行卷积，产生一个输出张量。
- Filters：正整数，定义卷积核的数目，即输出的维度，默认 32。
- Kernel Size：整数或由单个整数构成的列表或元组，定义卷积核的空域或时域窗口的长度，默认 2。
- Strides：整数或由单个整数构成的列表或元组，定义卷积的步长，默认 1。
- Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；causal 表示将产生因果（膨胀的）卷积，即 output[t]不依赖于 input[t+1：]，当对不能违反时间顺序的时序信号建模时有用；默认 valid。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, channels),channels_first 对应输入形状为（batch, channels, steps），默认 channels_last。
- Dilation Rate：整数或由单个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认 1。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Input Shape：整数元组或 None，定义输入数据的维度，该层作为

Sequential 层下的第一层时，需指定输入维度，例如(10,128)代表一个长为 10 的序列，序列中每个信号为 128 向量，而(None, 128)代表变长的 128 维向量序列，非第一层时输入 None，默认 None。

- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

- Conv2D

  - Conv2D：Conv2D 层，二维卷积层或图像空间卷积层，该层创建了一个卷积核，与输入层进行卷积，产生一个输出张量。
  - Filters：正整数，定义卷积核的数目，即输出的维度，默认 32。
  - Kernel Size：单个整数或由 2 个整数构成的列表或元组，指定二维卷积窗口的高度和宽度，可以是单个整数，以便为所有空间维度指定相同的值，默认(2, 2)。
  - Strides：单个整数或由两个整数构成的列表或元组，指定沿高度和宽度的卷积步长。可以是单个整数，以便为所有空间维度指定相同的值，默认（1,1）。
  - Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
  - Data Format：选择数据格式，定义输入数据的维度顺序，

channels_last 对应输入形状为（batch, height, width, channels），channels_first 对应输入形状为（batch, channels, height, width），默认 channels_last。

➢ Dilation Rate：单个整数或两个整数的元组/列表，指定用于扩展卷积的扩展速率。可以是单个整数，以便为所有空间维度指定相同的值，默认(1, 1)。

➢ Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。

➢ Input Shape：None 或整数组成的元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (128,128,3) 代表 128*128 的彩色 RGB 图像（data_format='channels_last'），非第一层时输入 None，默认 None。

➢ Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。

➢ Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。

➢ Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。

➢ Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

➢ Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

➢ Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

➢ Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

➢ Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

- Conv2DTranspose

➢ Conv2DTranspose：Conv2DTranspose 层，二维转置卷积层或去卷积层，可以使具有某些卷积输出形状的东西到具有其输入形状的东西，同时保持与上述卷积兼容的连接模式。

➢ Filters：正整数，定义卷积核的数目，即输出的维度，默认 32。

- Kernel Size：单个整数或由 2 个整数构成的列表或元组，指定二维卷积窗口的高度和宽度，可以是单个整数，以便为所有空间维度指定相同的值，默认(2,?2)。
- Strides：单个整数或由两个整数构成的列表或元组，指定沿高度和宽度的卷积步长。可以是单个整数，以便为所有空间维度指定相同的值，默认（1,1）。
- Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
- Output Padding：None 或单个整数或 2 个整数的元组/列表，指定沿着输出张量的高度和宽度的填充量。可以是单个整数，以便为所有空间维度指定相同的值。沿给定维度的输出填充量必须小于沿同一维度的步幅。默认 None，推断输出形状。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch,?height,?width,?channels），channels_first 对应输入形状为（batch,?channels,?height,?width），默认 channels_last。
- Dilation Rate：单个整数或两个整数的元组/列表，指定用于扩展卷积的扩展速率。可以是单个整数，以便为所有空间维度指定相同的值，默认(1,?1)。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Input Shape：None 或整数组成的元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (128,128,3) 代表 128*128 的彩色 RGB 图像（data_format='channels_last'），非第一层时输入 None，默认 None。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。

- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

● DepthwiseConv2D

- DepthwiseConv2D：DepthwiseConv2D 层，仅执行深度空间卷积中的第一步，分别作用于每个输入通道。
- Kernel Size：整数或由 2 个整数构成的列表或元组，指定二维卷积窗口的高度和宽度，可以是单个整数，以便为所有空间维度指定相同的值，默认(2, 2)。
- Strides：单个整数或由两个整数构成的列表或元组，指定沿高度和宽度的卷积步长，该 keras 版本仅支持元组数值一致。可以是单个整数，以便为所有空间维度指定相同的值，默认（1,1）。
- Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, height, width, channels），channels_first 对应输入形状为（batch, channels, height, width），默认 channels_last。
- depth_multiplier：每个输入通道的深度方向卷积输出通道的数量，默认 1。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。

- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- depthwise Initializer：选择深度方向核矩阵的初始化方法，默认'glorot_uniform'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- depthwise Regularizer：定义深度方向核矩阵的正则化方法，如l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或None，默认None。
- Bias Regularizer：定义偏置向量的正则化函数，如l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或None，默认None。
- Activity Regularizer：定义该层输出的正则化函数，如l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或None，默认None。
- depthwise Constraint：选择深度方向核矩阵的约束（限制）函数，默认None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认None。

- SeparableConv2D
  - SeparableConv2D：深度方向的可分离 2D 卷积，可分离的卷积的操作包括，首先执行深度方向的空间卷积 （分别作用于每个输入通道），紧接一个将所得输出通道 混合在一起的逐点卷积。
  - Filters：正整数，定义卷积核的数目，即输出的维度，默认32。
  - Kernel Size：单个整数或由2个整数构成的列表或元组，指定二维卷积窗口的高度和宽度，可以是单个整数，以便为所有空间维度指定相同的值，默认(2, 2)。
  - Strides：单个整数或由两个整数构成的列表或元组，指定沿高度和宽度的卷积步长。可以是单个整数，以便为所有空间维度指定相同的值，默认（1,1）。
  - Padding：选择填充的补0策略，valid表示只进行有效的卷积，即对边界数据不处理；same表示保留边界处的卷积结果，使输出形状和输入形状相同；默认valid。
  - Data Format：选择数据格式，定义输入数据的维度顺序，

channels_last 对应输入形状为（batch, height, width, channels），channels_first 对应输入形状为（batch, channels, height, width），默认 channels_last。

- Dilation Rate：单个整数或两个整数的元组/列表，指定用于扩展卷积的扩展速率。可以是单个整数，以便为所有空间维度指定相同的值，默认(1, 1)。
- depth_multiplier：每个输入通道的深度方向卷积输出通道的数量，默认 1。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- depthwise Initializer：运用到深度方向的核矩阵的初始化器，默认'glorot_uniform'。
- Pointwise Initializer：运用到逐点核矩阵的初始化器，默认'glorot_uniform'。
- depthwise Initializer：偏置向量的初始化器，默认'glorot_uniform'。
- Depthwise Regularizer：运用到深度方向的核矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：运用到偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Activity Regularizer：运用到层输出（它的激活值）的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- depthwise constraint：运用到深度方向的核矩阵的约束函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Pointwise Constraint：选择偏置向量的约束（限制）函数，默认 None。

- Conv3D

➢ Conv3D：Conv3D 层，三维卷积层或体积空间卷积层，该层创建了一个卷积核，与输入层进行卷积，产生一个输出张量。

➢ Filters：正整数，定义卷积核的数目，即输出的维度，默认 32。

➢ Kernel Size：一个整数或 3 个整数的元组/列表，指定三维卷积窗口的深度、高度和宽度。可以是单个整数，以便为所有空间维度指定相同的值，默认(2, 2, 2)。

➢ Strides：一个整数或 3 个整数的元组/列表，指定卷积在每个空间维度上的步长。是否可以使用单个整数为所有空间维度指定相同的值，默认(1, 1, 1)。

➢ Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状(batch, spatial_dim1, spatial_dim2, spatial_dim3, channels)，channels_first 对应输入形状(batch, channels, spatial_dim1, spatial_dim2, spatial_dim3)，默认 channels_last。

➢ Dilation Rate：一个整数或 3 个整数的元组/列表，指定用于扩展卷积的扩展速率。可以是单个整数，以便为所有空间维度指定相同的值，默认(1, 1, 1)。

➢ Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。

➢ Input Shape：None 或整数组成的元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如，input_shape=(128, 128, 128, 1)表示只有一个通道的 128x128x128 的图像数据（data_format='channels_last'），非第一层时输入 None，默认 None。

➢ Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。

➢ Kernel Initializer：选择权值初始化方法，默认'glorot_uniform'。

➢ Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。

➢ Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

➢ Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

➢ Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

➢ Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

➢ Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

● Conv3DTranspose

➢ Conv3DTranspose：Conv3DTranspose 层，三维转置卷积层或去卷积层，可以使具有某些卷积输出形状的东西到具有其输入形状的东西，同时保持与上述卷积兼容的连接模式。

➢ Filters：正整数，定义卷积核的数目，即输出的维度，默认 32。

➢ Kernel Size：一个整数或 3 个整数的元组/列表，指定三维卷积窗口的深度、高度和宽度。可以是单个整数，以便为所有空间维度指定相同的值，默认(2, 2, 2)。

➢ Strides：一个整数或 3 个整数的元组/列表，指定卷积在每个空间维度上的步长。是否可以使用单个整数为所有空间维度指定相同的值，默认(1, 1, 1)。

➢ Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。

➢ Output Padding：单个整数或 3 个整数的元组/列表，指定沿着输出张量的深度、高度和宽度的填充量。可以是单个整数，以便为所有空间维度指定相同的值。沿给定维度的输出填充量必须小于沿同一维度的步幅。默认 None，推断输出形状。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状(batch, depth, height, width, channels)，

channels_first 对应输入形状(batch, channels, depth, height, width)，默认 channels_last。

- Dilation Rate：一个整数或 3 个整数的元组/列表，指定用于扩展卷积的扩展速率。可以是单个整数，以便为所有空间维度指定相同的值，默认(1, 1, 1)。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Input Shape:整数元组,定义输入数据的维度,该层作为 Sequential 层下的第一层时，需指定输入维度，例如，input_shape=(128, 128, 128, 3)表示有 3 个通道的 128x128x128 的图像数据（data_format='channels_last'），非第一层时输入 None，默认 None。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

● Concatenate

- Concatenate：concatenate 层，对输入级联处理。
- axis：选择级联方式，数据按照指定的维度进行级联，Axis 对应的值为需要级联的数据维度，如-1，0，1，默认-1。

● Add

➢ Add：add 层，又称合并层，添加输入列表的层，接受一组形状相同的张量作为输入，然后返回一个形状相同的张量。

- Multiply
  - ➢ Multiply：Multiply 层，计算输入张量列表的（逐元素间的）乘积。

- Flatten
  - ➢ Flatten：Flatten 层，将输入扁平化处理，不影响批次大小。
  - ➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, height, weight, ..., channels），channels_first 对应输入形状为(batch, channels, height, weight, ...)，默认 channels_last。

- RepeatVector
  - ➢ RepeatVector：RepeatVector 层，将输入重复 n 次。
  - ➢ Number of Repeat：整数，指定输入数据的重复次数，默认 0。

- ZeroPadding1D
  - ➢ ZeroPadding1D：ZeroPadding1D 层，一维输入的零填充层。
  - ➢ Padding：整数或 2 个整数的元组，定义填充大小，默认 1。

- ZeroPadding2D
  - ➢ ZeroPadding2D：ZeroPadding2D 层，二维输入的零填充层，可以在图像张量的顶部、底部、左侧和右侧添加由 0 组成的行和列。
  - ➢ Padding：整数或 2 个整数的元组或 2 个整数的元组，定义填充大小，默认(1, 1)。
  - ➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, spatial_dim1, spatial_dim2, spatial_dim3, channels)，channels_first 对应输入形状为(batch,

channels, spatial_dim1, spatial_dim2, spatial_dim3），默认channels_last。

- Cropping1D
  - Cropping1D：Cropping1D层，一维输入的裁剪层(如时间序列)，其沿着时间维度(轴1)生长。
  - Cropping：一维输入(如时序)的裁剪层，沿着时间维度生长，输入整数或长为2的整数元组，指定在序列的首尾要裁剪掉多少个元素，默认(1, 1）。

- Cropping2D
  - Cropping2D：Cropping2D层，二维输入的剪裁层（如图像），其沿着空间维度生长，即高度和宽度。
  - Cropping：二维输入(例如图片)的裁剪层输入，沿着空间维度（高度和宽度）生长。输入整数或2个整数的元组或2个整数的2个元组的元组，如果为整数（如1），表示相同的对称裁剪应用于高度和宽度；如果为 2 个整数的元组（如（1,1））， 表示对高度和宽度的两个不同的对称裁剪值：(symmetric_height_crop, symmetric_width_crop)；如果为2个整数的2个元组的元组：解释为?((top_crop, bottom_crop), (left_crop, right_crop))，默认((0,0),(0,0))。
  - Data Format：选择数据格式，定义输入数据的维度顺序，channels_last对应输入形状为（batch,?height,?width,?channels），channels_first对应输入形状为（batch,?channels,?height,?width），默认channels_last。

- Cropping3D
  - Cropping3D：Cropping3D层，三维数据的剪裁层（如空间或时空）。
  - Cropping：三维数据(如空间或时空)的裁剪层，输入整数或3个整数的元组或2个整数的3个元组的元组，如果为整数，将对深度、宽度和高

度应用相同的对称裁剪；如果为 3 个整数的元组：解释为对深度、高度和宽度的 3 个不同的对称裁剪值：(symmetric_dim1_crop, symmetric_dim2_crop, symmetric_dim3_crop)；如果为 2 个整数的 3 个元组的元组：解释为((left_dim1_crop, right_dim1_crop), (left_dim2_crop, right_dim2_crop), (left_dim3_crop, right_dim3_crop))，默认((1,1),(1,1),(1,1))。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, spatial_dim1, spatial_dim2, spatial_dim3, channels)，channels_first 对应输入形状为(batch, channels, spatial_dim1, spatial_dim2, spatial_dim3)，默认 channels_last。

- UpSampling1D

➢ UpSampling1D：UpSampling1D 层，一维输入的上采样层，沿时间轴重复每个时间步长 n 次。

➢ UpSampling Size：整数，定义上采样因子，指沿着时间轴重复每个时间步的次数，默认 2。

- UpSampling2D

➢ UpSampling2D：UpSampling2D 层，二维输入的上采样层，按参数大小分别重复数据的行和列。

➢ UpSampling Size：整数或 2 个整数的元组，定义行和列的上采样因子，指沿着数据的行和列分别重复的次数，默认（2,2）。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch,?height,?width,?channels），channels_first 对应输入形状为（batch,?channels,?height,?width），默认 channels_last。

➢ Interpolation：选择插值方法，nearest 或 bilinear。

- UpSampling3D

  - UpSampling3D：UpSampling3D 层，三维输入的上采样层，按参数大小分别重复数据的第 1、2、3 维。

  - UpSampling Size：整数或 3 个整数的元组，定义 3 个维度的上采样因子，指沿着数据的 3 个维度分别重复的次数，默认(2,2,2)。

  - Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, spatial_dim1, spatial_dim2, spatial_dim3, channels)，channels_first 对应输入形状为(batch, channels, spatial_dim1, spatial_dim2, spatial_dim3)，默认 channels_last。

- MaxPooling1D

  - MaxPooling1D：MaxPooling1D 层，一维最大池化层，对一维时序数据进行最大池化操作。

  - Pool Size：整数，定义最大池化的窗口大小，默认 2。

  - Strides：整数或 None，表示下采样因子，例如 2 表示将使得输出形状为输入的一半，None 表示默认使用 pool_size 的值，默认 None。

  - Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。

  - Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- MaxPooling2D

  - MaxPooling2D：MaxPooling2D 层，二维最大池化层，对二维空间数据进行最大池化操作。

  - Pool Size：整数，或者 2 个整数表示的元组，沿(垂直，水平)方向缩小比例的因数。(2,2)会把输入张量的两个维度都缩小一半。如果只使用

一个整数，那么两个维度都会使用同样的窗口长度，默认(2,2)。

- Strides：整数或长为 2 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。
- Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch,?height,?width,?channels），channels_first 对应输入形状为（batch,?channels,?height,?width），默认 channels_last。

- MaxPooling3D
  - MaxPooling3D：MaxPooling3D 层，三维最大池化层，对三维数据（空间或时空）进行最大池化操作。
  - Pool Size： 3 个整数表示的元组，缩小(dim1,dim2,dim3)比例的因数。(2,2,2)会把 3D 输入张量的每个维度缩小一半，默认(2,2,2)。
  - Strides：长为 3 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。
  - Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
  - Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。

- AveragePooling1D
  - AveragePooling1D：AveragePooling1D 层，一维平均池化层，对时

序数据进行平均池化操作。

- Pool Size：整数，定义平均池化的窗口大小，默认 2。
- Strides：整数或 None，表示下采样因子，例如 2 表示将使得输出 shape 为输入的一半，None 表示默认使用 pool_size 的值，默认 None。
- Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- AveragePooling2D
  - AveragePooling2D：AveragePooling2D 层，二维平均池化层，对二维空间数据进行平均池化操作。
  - Pool Size：整数或者 2 个整数表示的元组，沿(垂直，水平)方向缩小比例的因数。(2,2)会把输入张量的两个维度都缩小一半。如果只使用一个整数，那么两个维度都会使用同样的窗口长度，默认(2,2)。
  - Strides：整数或长为 2 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。
  - Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
  - Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch,?height,?width,?channels），channels_first 对应输入形状为（batch,?channels,?height,?width），默认 channels_last。

- AveragePooling3D
  - AveragePooling3D：AveragePooling3D 层，三维平均池化层，对三

维数据（空间或时空）进行平均池化操作。

➢ Pool Size：3 个整数表示的元组，缩小(dim1,dim2,dim3)比例的因数。(2,2,2)会把 3D 输入张量的每个维度缩小一半。

➢ Strides：长为 3 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。

➢ Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。

● GlobalAveragePooling1D

➢ GlobalAveragePooling1D：GlobalAveragePooling1D 层，对于时序数据的全局平均池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features),channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

● GlobalAveragePooling2D

➢ GlobalAveragePooling2D：GlobalAveragePooling2D 层，对于空域数据的全局平均池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features),channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

● GlobalAveragePooling3D

➢ GlobalAveragePooling3D：GlobalAveragePooling3D 层，对于 3D 数据的全局平均池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- GlobalMaxPooling1D

➢ GlobalMaxPooling1D：GlobalMaxPooling3D 层，对于时序数据的全局最大池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- GlobalMaxPooling2D

➢ GlobalMaxPooling2D：GlobalMaxPooling2D 层，对于空域数据的全局最大池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- GlobalMaxPooling3D

➢ GlobalMaxPooling3D：GlobalMaxPooling3D 层，对于 3D 数据的全局最大池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- GlobalMaxPooling3D

➢ GlobalMaxPooling3D：GlobalMaxPooling3D 层，对于 3D 数据的全

局最大池化。

➢ Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为(batch, steps, features)，channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

- BatchNormalization

➢ BatchNormalization：BatchNormalization 层，对每个批处理中正则化前一步激活函数的结果。

➢ Axis：整数，指定需要标准化的轴，例如，在 data_format='channels_first'的 Conv2D 层之后，在 BatchNormalization 中设置 axis=1。默认-1。

➢ Momentum：浮点数，指定移动均值和移动方差的动量，默认 0.99。

➢ Epsilon：浮点数，为方差增加一个小的浮点数，避免被除数为零，默认 0.001。

➢ Center：布尔值，如果为 True，表示把偏移量加到标准化张量上，如果为 False，则忽略，默认 True。

➢ Scale：布尔值，如果为 True，则乘以 gamma，如果为 False，则不使用 gamma。

➢ Beta Initializer：指定 beta 权重的初始化方法，默认 zeros。

➢ Gamma Initializer：指定 gamma 权重的初始化方法，默认 ones。

➢ Moving Mean Initializer:指定移动平均的初始化方法,默认 zeros。

➢ Moving Variance Initializer：指定移动方差的初始化方法，默认 ones。

➢ Beta Regularizer：为 beta 权重指定正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ Gamma Regularizer：为 gamma 权重指定正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ Beta Constraint：指定 beta 权值的约束函数，默认 None。

➢ Gamma Constraint：指定 gamma 权值的约束函数，默认 None。

- renorm：是否使用批量再归一化。
- renorm_clipping：一种字典，可以将关键字“rmax”，“rmin”，“dmax”映射到用于剪裁 renorm 校正的标量张量。校正（r，d）用作 corrected_value = normalized_value * r + d，其中 r 被剪裁为[rmin，rmax]，d 被剪裁为[-dmax，dmax]。缺少的 rmax、rmin 和 dmax 分别设置为 inf、0 和 inf，默认 None。
- renorm_momentum：用 renorm 更新滑动方式和标准偏差的动量。与动量不同，会影响训练，既不应太小（会增加噪音），也不应太大（会给出过时的估计），默认 0.99。
- fused：如果是 True，使用更快的融合实现，在没有可用的快速融合实现将抛出异常。如果为 False，则不使用；如果是 None，会使用可以使用的快速融合。
- trainable：如果为 True，还将变量添加到图形集合。
- virtual_batch_size：一个 int。默认情况下，virtual_batch_size 为 None，这意味着在整个批次中执行批次规范化。当 virtual_batch_size 不是 None 时，改为执行“Ghost Batch Normalization”，创建每个单独规范化的虚拟子批（使用共享 gamma、beta 和滑动统计）。必须在执行期间划分实际批大小。
- adjustment：仅在训练期间，采用包含输入张量（动态）形状的张量并返回一对（scale、bias）以应用于标准化值（γ 和 β 之前）的函数。例如，如果 axis=-1，adjustment = lambda shape: (tf.random_uniform(shape[-1:], 0.93, 1.07) , tf.random_uniform(shape[-1:], -0.1, 0.1))将标准化值向上或向下缩放 7%，然后将结果向上滑动 0.1（每个功能都有独立的缩放和偏移，但在所有示例中都有共享），最后应用 gamma 和/或 beta。如果没有，则不用调整。如果指定了 virtual_batch_size，则无法指定。
- name：字符串，层的名称。

- SimpleRNN

- SimpleRNN：SimpleRNN 层，全连接的 RNN 层，该层的输出可以反馈到输入。
- Units：整数，定义输出空间的维度，默认 8。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单位下降的分数，默认 0.0。
- Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。
- Return Sequences：布尔值，指定是返回输出序列中的最后一个输出，还是返回完整序列，默认 False。
- Return State：布尔值，指定是否返回输出之外的最后一个状态，

默认 False。

- Go Backwards：布尔值，指定是否反向处理输入序列并返回相反的序列。如果为真，则反向处理输入序列并返回相反的序列，默认 False。
- Stateful：布尔值，如果为真，则批处理中索引 i 处的每个样本的最后一个状态将用作下一批处理中索引 i 的样本的初始状态，默认 False。
- Unroll：布尔值，如果为真，则网络将展开，否则将使用符号循环。展开可以加快 RNN 的速度，尽管它往往需要更多的内存。展开只适用于短序列，默认 False。
- Input Shape：正整数组成的元组或 None。检索图层的输入形状，默认为 None。

- GRU

- GRU：GRU 层，门控循环单元，作为 LSTM 的一种变体，将忘记门和输入门合成了一个单一的更新门。
- Units：整数，定义输出空间的维度，默认 8。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Recurrent Activation：选择循环时间步的激活函数，如'relu'。若选择'None'，默认使用'linear'。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、

12(0.01)、11_12(l1=0.01,?l2=0.01)或 None，默认 None。

- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单位下降的分数，默认 0.0。
- Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。
- Implementation：执行模式，1 或 2。模式 1 将其操作结构化为大量的小点积和加法，而模式 2 将其批量处理成更少、更大的操作。这些模式在不同的硬件和不同的应用程序上具有不同的性能，默认 2。
- Return Sequences：布尔值，指定是返回输出序列中的最后一个输出，还是返回完整序列，默认 False。
- Return State：布尔值，指定是否返回输出之外的最后一个状态，默认 False。
- Go Backwards：布尔值，指定是否反向处理输入序列并返回相反的序列。如果为真，则反向处理输入序列并返回相反的序列，默认 False。
- Stateful：布尔值，如果为真，则批处理中索引 i 处的每个样本的最后一个状态将用作下一批处理中索引 i 的样本的初始状态，默认 False。
- Unroll：布尔值，如果为真，则网络将展开，否则将使用符号循环。展开可以加快 RNN 的速度，尽管它往往需要更多的内存。展开只适用于短序列，默认 False。
- Unroll：布尔值，指定是否在矩阵乘法之前或之后使用重置门，默认 False，表示之前，True 表示之后。
- Input Shape：正整数组成的元组或 None。检索图层的输入形状，默认为 None。

- LSTM

  - LSTM：LSTM 层，长短期记忆网络层，可以将以往学习的结果应用到当前学习的模型。

  - Units：整数，定义输出空间的维度，默认 8。

  - Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。

  - Recurrent Activation：选择循环时间步的激活函数，如'relu'。若选择'None'，默认使用'linear'。

  - Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。

  - Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。

  - Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。

  - Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。

  - Unit Forget Bias：布尔值。如果为真，则在初始化时对忘记门的偏差加 1。将其设置为 true 也将强制 bias_initializer='zeros'.默认 True。

  - Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

  - Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

  - Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

  - Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

  - Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

  - Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。

  - Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

  - Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单

位下降的分数，默认 0.0。

➢ Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。

➢ Implementation：执行模式，1 或 2。模式 1 将其操作结构化为大量的小点积和加法，而模式 2 将其批量处理成更少、更大的操作。这些模式在不同的硬件和不同的应用程序上具有不同的性能，默认 2。

➢ Return Sequences：布尔值，指定是返回输出序列中的最后一个输出，还是返回完整序列，默认 False。

➢ Return State：布尔值，指定是否返回输出之外的最后一个状态，默认 False。

➢ Go Backwards：布尔值，指定是否反向处理输入序列并返回相反的序列。如果为真，则反向处理输入序列并返回相反的序列，默认 False。

➢ Stateful：布尔值，如果为真，则批处理中索引 i 处的每个样本的最后一个状态将用作下一批处理中索引 i 的样本的初始状态，默认 False。

➢ Unroll：布尔值，如果为真，则网络将展开，否则将使用符号循环。展开可以加快 RNN 的速度，尽管它往往需要更多的内存。展开只适用于短序列，默认 False。

➢ Input Shape：正整数组成的元组或 None。检索图层的输入形状，默认为 None。

- ConvLSTM2D

➢ convLSTM2D：ConvLSTM2D 层，卷积 LSTM 层，类似于 LSTM 层，其输入转换和递归转换都是卷积的。

➢ Filters：正整数，定义卷积核的数目，即输出的维度，默认 32。

➢ Kernel Size：整数或由 2 个整数构成的列表或元组，指定二维卷积窗口的高度和宽度，可以是单个整数，以便为所有空间维度指定相同的值，默认(2,?2)。

➢ Strides：单个整数或由两个整数构成的列表或元组，指定沿高度和宽度的卷积步长。可以是单个整数，以便为所有空间维度指定相同的值，默

认（1,1）。

- Padding：选择填充的补 0 策略，valid 表示只进行有效的卷积，即对边界数据不处理；same 表示保留边界处的卷积结果，使输出形状和输入形状相同；默认 valid。
- Data Format：选择数据格式，定义输入数据的维度顺序，channels_last 对应输入形状为（batch, time, ... , channels），channels_first 对应输入形状为(batch, time, channels, ...)，默认 channels_last。
- Dilation Rate：一个整数或 n 个整数的元组/列表，指定用于扩展卷积的扩展速率。可以是单个整数，以便为所有空间维度指定相同的值，默认(1,?1)。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Recurrent Activation：选择循环时间步的激活函数，如'relu'。若选择'None'，默认使用'linear'。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Unit Forget Bias：布尔值。如果为真，则在初始化时对忘记门的偏差加 1。将其设置为 true 也将强制 bias_initializer='zeros'.默认 True。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Return Sequences：布尔值，指定是返回输出序列中的最后一个输出，还是返回完整序列，默认 False。
- Go Backwards：布尔值，指定是否反向处理输入序列并返回相反的序列。如果为真，则反向处理输入序列并返回相反的序列，默认 False。
- Stateful：布尔值，如果为真，则批处理中索引 i 处的每个样本的最后一个状态将用作下一批处理中索引 i 的样本的初始状态，默认 False。
- Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单位下降的分数，默认 0.0。
- Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。

● SimpleRNNCell

- SimpleRNNCell：SimpleRNNCell 层，全连接 RNN 单元格。
- Units：整数，定义输出空间的维度，默认 16。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单位下降的分数，默认 0.0。
- Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。

● GRUCell

- GRUCell：GRUCell 层，GRU 单元格。
- Units：整数，定义输出空间的维度，默认 16。
- Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
- Recurrent Activation：选择循环时间步的激活函数，如'relu'。若选择'None'，默认使用'linear'。
- Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。

- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单位下降的分数，默认 0.0。
- Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。
- Implementation：执行模式，1 或 2。模式 1 将其操作结构化为大量的小点积和加法，而模式 2 将其批量处理成更少、更大的操作。这些模式在不同的硬件和不同的应用程序上具有不同的性能，默认 2。
- Unroll：布尔值，指定是否在矩阵乘法之前或之后使用重置门，默认 False，表示之前，True 表示之后。

- LSTMCell
  - LSTMCell：LSTMCell 层，LSTM 单元格。
  - Units：整数，定义输出空间的维度，默认 8。
  - Activation：选择激活函数进行非线性转换，如'relu'。若选择'None'，默认使用'linear'。
  - Recurrent Activation：选择循环时间步的激活函数，如'relu'。若选择'None'，默认使用'linear'。
  - Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
  - Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
  - Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
  - Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
  - Unit Forget Bias：布尔值。如果为真，则在初始化时对忘记门的偏差加 1。将其设置为 true 也将强制 bias_initializer='zeros'.默认 True。
  - Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
  - Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Dropout：0 到 1 之间的整数或者浮点数，定义输入的线性变换的单位下降的分数，默认 0.0。
- Recurrent Dropout：0 到 1 之间的整数或者浮点数，定义递归状态的线性变换中单位下降的部分，默认 0.0。
- Implementation：执行模式，1 或 2。模式 1 将其操作结构化为大量的小点积和加法，而模式 2 将其批量处理成更少、更大的操作。这些模式在不同的硬件和不同的应用程序上具有不同的性能，默认 2。

- CuDNNGRU
  - CuDNNGRU：CuDNNGRU 层，由 CuDNN 支持的快速 GRU 实现。
  - Units：整数，定义输出空间的维度，默认 8。
  - Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
  - Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
  - Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
  - Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
  - Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
  - Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
  - Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。

- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Return Sequences：布尔值，指定是返回输出序列中的最后一个输出，还是返回完整序列，默认 False。
- Return State：布尔值，指定是否返回输出之外的最后一个状态，默认 False。
- Stateful：布尔值，如果为真，则批处理中索引 i 处的每个样本的最后一个状态将用作下一批处理中索引 i 的样本的初始状态，默认 False。

● CuDNNLSTM

- CuDNNLSTM：快速 LSTM 层，由 CuDNN 支持的快速 LSTM 实现。
- Units：整数，定义输出空间的维度，默认 8。
- Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
- Recurrent Initializer：选择递归核权值矩阵初始化方法，用于递归状态的线性变换，默认'orthogonal'。
- Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
- Unit Forget Bias：布尔值。如果为真，则在初始化时对忘记门的偏差加 1。将其设置为 true 也将强制 bias_initializer='zeros'.默认 True。
- Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Recurrent Regularizer：定义递归和权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Activity Regularizer：定义该层输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01,?l2=0.01)或 None，默认 None。
- Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。

- Recurrent Constraint：选择递归和权值矩阵的约束（限制）函数，默认 None。
- Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。
- Return Sequences：布尔值，指定是返回输出序列中的最后一个输出，还是返回完整序列，默认 False。
- Return State：布尔值，指定是否返回输出之外的最后一个状态，默认 False。
- Stateful：布尔值，如果为真，则批处理中索引 i 处的每个样本的最后一个状态将用作下一批处理中索引 i 的样本的初始状态，默认 False。

● Attention

- Attention：点乘注意力层，Luong Attention。
- use_scale：如果为 True，将会创建一个标量的变量对注意力分数进行缩放，默认'True'。
- causal：设置为 True 可使解码器 self-attention。添加一个罩，使位置 i 无法参与位置 j> i。这样可以防止信息流从未来传递到过去，默认'True'。
- dropout：输入 0 到 1 之间的整数或者浮点数，attention scores 下降的百分比，默认 0.0。

● AdditiveAttention

- AdditiveAttention：加法注意力层，Bahdanau Attention。
- use_scale：如果为 True，将会创建一个标量的变量对注意力分数进行缩放，默认'True'。
- causal：设置为 True 可使解码器 self-attention。添加一个罩，使位置 i 无法参与位置 j> i。这样可以防止信息流从未来传递到过去，默认'True'。
- dropout：输入 0 到 1 之间的整数或者浮点数，attention scores 下降的百分比，默认 0.0。

- MultiHeadAttention

  - MultiHeadAttention：多头注意力层，多头自注意力是将一个查询（query）到一系列（键 key-值 value）对的线性映射 h 次，再进行 h 次自注意力操作，得到 h 个自注意结果进行拼接状。
  - num_heads：正整数，注意力头的数量，如 2。
  - key_dim：正整数，每个注意力头 query 和 key 的大小，如 2。
  - value_dim：正整数或者 None，每个注意力头的价值大小，默认为 None。
  - dropout：输入 0 到 1 之间的整数或者浮点数，丢弃率，默认 0.0。
  - Use Bias：选择是否使用偏置向量，默认'True'，即包含偏置。
  - output_shape：输出张量的预期形状，除了批次和序列维度。如果未指定，则投影回关键函数维度。默认设置为 None。
  - attention_axes：应用注意力的轴。None 表示对所有轴的注意力，但批处理、头部和特征。默认 None。
  - Kernel Initializer:选择权值初始化方法,默认'glorot_uniform'。
  - Bias Initializer：选择偏置向量的初始化方法，默认'zeros'。
  - Kernel Regularizer：定义权值矩阵的正则化方法，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Bias Regularizer：定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Bias Regularizer：定义激活函数的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
  - Kernel Constraint：选择权值矩阵的约束（限制）函数，默认 None。
  - Bias Constraint：选择偏置向量的约束（限制）函数，默认 None。

- TimeDistributed

  - TimeDistributed：TimeDistributed 包装器，可以把一个层应用到输入的每一个时间步上。

- Input Shape：正整数组成的元组或 None。检索图层的输入形状，默认为 None。

- Bidirectional
  - Bidirectional：双向 RNN 包装器。
  - merge_mode：前向和后向 RNN 输出的结合方式，为 sum,mul,concat,ave 和 None 之一，若设为 None，则返回值不结合，而是以列表的形式返回。
  - Input Shape：正整数组成的元组或 None。检索图层的输入形状，默认为 None。

- 自定义层
  - 自定义层：自定义层。
  - 自定义层名称：填写自定义层函数的名称。
  - 自定义层脚本：按照模板添加自定义层脚本。

3.2.2 基础网络模型

- densenet
  - densenet：DenseNet 模型，稠密卷积网络模型结构。
  - version:DenseNet 网络模型的版本,默认情况下,使用 DenseNet121。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - weights:None 代表随机初始化,'imagenet'代表加载在 ImageNet 上预训练的权值。
  - input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
  - input_shape：可选，输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (244, 244, 3)（对于 channels_last 数据格式），或者 (3, 244, 244)（对于 channels_first 数据格式）。它必须拥

有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。

- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。

- efficientnet
  - efficientnet：efficientnet 模型。
  - version：efficientnet 网络模型的版本，默认情况下，使用 EfficientNetB0。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - weights：None 代表随机初始化，'imagenet' 代表加载在 ImageNet 上预训练的权值。
  - input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
  - input_shape：可选，输入尺寸元组，仅当 include_top=False 时有效，它必须拥有 3 个输入通道，默认为 None。
  - pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
  - classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。

➢ classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- InceptionResNetV2

➢ InceptionResNetV2：Inception-ResNet V2 模型，权值由 ImageNet 训练而来。

➢ include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。

➢ weights：None 代表随机初始化，'imagenet'代表加载在 ImageNet 上预训练的权值。

➢ input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。

➢ input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (299, 299, 3)（对于 channels_last 数据格式），或者 (3, 299, 299)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 139。例如 (150, 150, 3) 是一个合法的输入尺寸，默认为 None。

➢ pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg'代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。

➢ classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。

➢ classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- InceptionV3

➢ InceptionV3：InceptionV3 模型，权值由 ImageNet 训练而来。

- include_top：是否在网络顶部包括全连接层。默认情况下，使用True。
- weights：None 代表随机初始化，'imagenet' 代表加载在 ImageNet 上预训练的权值。
- input_tensor：可选，Keras tensor 作为模型的输入（即layers.Input() 输出的 tensor），默认为 None。
- input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (299, 299, 3)（对于 channels_last 数据格式），或者 (3, 299, 299)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 139。例如 (150, 150, 3) 是一个合法的输入尺寸，默认为 None。
- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
- classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- MobileNet
  - MobileNet：MobileNet 模型，权值由 ImageNet 训练而来。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用True。
  - alpha：非负的浮点数，控制网络的宽度.如果 alpha < 1.0，则同比例减少每层的滤波器个数；如果 alpha > 1.0，则同比例增加每层的滤波器个数；如果 alpha = 1，使用论文默认的滤波器个数；默认 1.0。
  - depth_multiplier：depthwise 卷积的深度乘子，也称为（分辨率乘

子），默认 1。

- Dropout：浮点数或科学计数法，丢弃率，默认 1e-3。
- weights：None 代表随机初始化，'imagenet' 代表加载在 ImageNet 上预训练的权值。
- input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
- input_shape：可选，输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (224, 224, 3)（对于 channels_last 数据格式），或者 (3, 224, 224)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。
- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
- classifier_activation：选择激活函数进行非线性转换，如 'softmax'，默认使用'softmax'。

● MobileNetV2

- MobileNetV2：MobileNetV2 模型，权值由 ImageNet 训练而来。
- include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
- alpha：非负的浮点数，控制网络的宽度.如果 alpha < 1.0，则同比例减少每层的滤波器个数；如果 alpha > 1.0，则同比例增加每层的滤波器个数；如果 alpha = 1，使用论文默认的滤波器个数；默认 1.0。
- weights：None 代表随机初始化，'imagenet' 代表加载在 ImageNet

上预训练的权值。

- input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
- input_shape：可选形状元组，如果要使用输入 img 分辨率不是（224，224，3）的模型，则需要指定，应是 3 个通道（224,224,3）。如果要从输入张量推断输入形状，也可以省略此选项。如果您选择同时包含 input_tensor 和 input_shape，如果它们匹配，则将使用 input_shape，如果形状不匹配，则我们将抛出一个错误。如（160，160，3）将是一个有效值。默认为 None。
- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
- classifier_activation：选择激活函数进行非线性转换，如 'softmax'，默认使用'softmax'。

- nasnet
  - nasnet：NASNet 模型，神经结构搜索网络模型，权值由 ImageNet 训练而来。
  - version：nasnet 网络模型的版本，默认情况下，使用 NASNetMobile。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - weights：None 代表随机初始化，'imagenet' 代表加载在 ImageNet 上预训练的权值。
  - input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。

- input_shape：输入尺寸元组，仅当 include_top=False 时有效，对于 NASNetMobile 模型来说，输入形状必须是 (224, 224, 3)（channels_last 格式）或 (3, 224, 224)（channels_first 格式），对于 NASNetLarge 模型来说，输入形状必须是 (331, 331, 3)（channels_last 格式）或 (3, 331, 331)（channels_first 格式）。它必须为 3 个输入通道，且宽高必须不小于 32，比如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。
- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。

- resnet
  - resnet：resnet 模型，权值由 ImageNet 训练而来。
  - version：ResNet 网络模型的版本，默认情况下，使用 ResNet101。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - weights：None 代表随机初始化，'imagenet' 代表加载在 ImageNet 上预训练的权值。
  - input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
  - input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (244, 244, 3)（对于 channels_last 数据格式），或者 (3, 244, 244)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。

- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg'代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。

- resnet_v2
  - resnet_v2：resnet_v2 模型，权值由 ImageNet 训练而来。
  - version:ResNetV2 网络模型的版本,默认情况下,使用 ResNet101V2。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - weights:None 代表随机初始化,'imagenet'代表加载在 ImageNet 上预训练的权值。
  - input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
  - input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (244, 244, 3)（对于 channels_last 数据格式），或者 (3, 244, 244)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。
  - pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg'代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
  - classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。

➢ classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- VGG16
  - ➢ VGG16：VGG16 模型，权值由 ImageNet 训练而来。
  - ➢ include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - ➢ weights:None 代表随机初始化,'imagenet'代表加载在 ImageNet 上预训练的权值。
  - ➢ input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
  - ➢ input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (224, 224, 3)（对于 channels_last 数据格式），或者 (3, 224, 224)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。
  - ➢ pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg'代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
  - ➢ classes:输入 None 或正整数,图片分类的类别数,仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
  - ➢ classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- VGG19
  - ➢ VGG19：VGG19 模型，权值由 ImageNet 训练而来。
  - ➢ include_top：是否在网络顶部包括全连接层。默认情况下，使用

True。

- weights：None 代表随机初始化，'imagenet'代表加载在 ImageNet 上预训练的权值。
- input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
- input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (244, 244, 3)（对于 channels_last 数据格式），或者 (3, 244, 244)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。
- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg'代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
- classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- Xception
  - Xception：Xception 模型，权值由 ImageNet 训练而来。
  - include_top：是否在网络顶部包括全连接层。默认情况下，使用 True。
  - weights：None 代表随机初始化，'imagenet'代表加载在 ImageNet 上预训练的权值。
  - input_tensor：可选，Keras tensor 作为模型的输入（即 layers.Input() 输出的 tensor），默认为 None。
  - input_shape：输入尺寸元组，仅当 include_top=False 时有效（否

则输入形状必须是 (299, 299, 3)，因为预训练模型是以这个大小训练的）。它必须拥有 3 个输入通道，且宽高必须不小于 71。例如 (150, 150, 3) 是一个合法的输入尺寸，默认为 None。

- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
- classifier_activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- 模型库
  - 模型库：训练完成的模型。
  - 模型名称：现有模型的名称。
  - 模型版本：选择具体使用哪个模型。

3.2.3 模型结果可视化

- TensorBoard
  - TensorBoard：配置 TensorBoard 参数。
  - histogram_freq：对于模型中各个层计算激活值和模型权重直方图的频率（训练轮数中）。 如果设置成 0 ，直方图不会被计算。对于直方图可视化的验证数据（或分离数据）一定要明确地指出。
  - write_graph：是否在 TensorBoard 中可视化图像。
  - write_grads：是否在 TensorBoard 中可视化梯度值直方图，histogram_freq 必须大于 0。
  - batch_size：用以直方图计算的传入神经元网络输入批的大小。
  - write_images:是否在 TensorBoard 中将模型权重以图

片可视化。

- embeddings_freq：被选中的嵌入层会被保存的频率（在训练轮中）。
- embeddings_layer_names：一个列表，会被监测层的名字。 如果是 None 或空列表，那么所有的嵌入层都会被监测。
- embeddings_metadata：一个字典，对应层的名字到保存有这个嵌入层元数据文件的名字。
- embeddings_data：要嵌入在 embeddings_layer_names 指定的层的数据。 Numpy 数组（如果模型有单个输入）或 Numpy 数组列表（如果模型有多个输入）。
- update_freq：'batch' 或 'epoch' 或 整数。当使用 'batch' 时，在每个 batch 之后将损失和评估值写入 TensorBoard Log 中。同样的情况应用到 'epoch' 中。如果使用整数，例如 10000，这个回调会在每 10000 个样本之后将损失和评估值写入到 TensorBoard 中。

- 模型评估

3.3 算法库组件

3.3.1 数值算法

- KNeighborsClassifier
  - KNeighborsClassifier:KNeighborsClassifier,K 近邻算法分类器，基于每个查询点的指定的最近邻数量进行学习分类。
  - 模型名称：保存模型的名称。
  - n_neighbors：整数，查询的默认邻居的数量，默认 5。
  - weights：选择用于预测的权重函数，包括 uniform 和 distance。其中 uniform 表示统一的权重，即在每一个邻居区域里的点的权重都是一样的，distance 表示权重点等于他们距离的倒数，使用此函数，更近的邻居对于所预测的点的影响更大。默认 uniform。
  - algorithm:选择最近邻所用的算法，包括 auto、ball_tree、kd_tree、brute，其中 auto 表示会根据传递给 fit 的数据自动决定使用最合适的算法，

ball_tree 表示使用 BallTree 算法，kd_tree 表示使用 KDTree 算法，brute 表示使用暴力搜索。默认 auto。

- leaf_size：正整数，指定叶子数量。叶子大小传递给 BallTree 或 KDTree，这可能会影响构造和查询的速度，以及存储树所需的内存，最优值取决于问题的性质。默认 30。
- p：整数，闵科夫斯基距离指数或超参数。p=1 等价于使用曼哈顿距离（l1），p=2 等价于欧式距离(l2)，对于任何 p ，使用的是闵科夫斯基距离（l_p）。默认 2。
- metric：选择用于树的距离度量，如果和 p=2 一起使用相当于使用标准欧几里德距离度量，默认 minkowski，目前仅支持实数值向量空间的度量。
- metric_params：字典或 None，度量函数的附加关键字参数。默认 None。
- n_jobs：整数或 None 或-1，表示要为邻居搜索运行的并行作业的数量，-1 表示判断是否可以在这里设置并行，None 表示 1，-1 表示所有，默认 None。

- RadiusNeighborsClassifier
  - RadiusNeighborsClassifier：RadiusNeighborsClassifier，半径近邻分类算法，根据在给定半径内的邻域占比实现分类。
  - 模型名称：保存模型的名称。
  - Radius：浮点数，radius_neighbors 查询默认使用的参数空间范围，默认 1.0。
  - weights：选择用于预测的权重函数，包括 uniform 和 distance。其中 uniform 表示统一的权重,即在每一个邻居区域里的点的权重都是一样的，distance 表示权重点等于他们距离的倒数，使用此函数，更近的邻居对于所预测的点的影响更大。默认 uniform。
  - Algorithm:选择最近邻所用的算法,包括 auto、ball_tree、kd_tree、brute,其中 auto 表示会根据传递给 fit 的数据自动决定使用最合适的算法，

ball_tree 表示使用 BallTree 算法，kd_tree 表示使用 KDTree 算法，brute 表示使用暴力搜索。默认 auto。

- leaf_size：整数，指定叶子数量。叶子大小传递给 BallTree 或 KDTree，这可能会影响构造和查询的速度，以及存储树所需的内存，最优值取决于问题的性质。默认 30。
- p：整数，闵科夫斯基距离指数或超参数。p=1 等价于使用曼哈顿距离（l1），p=2 等价于欧式距离(l2)，对于任何 p ，使用的是闵科夫斯基距离（l_p）。默认 2。
- metric：选择用于树的距离度量，如果和 p=2 一起使用相当于使用标准欧几里德距离度量，默认 minkowski，目前仅支持实数值向量空间的度量。
- outlier_label：用于离群样本(给定半径内没有邻居的样本)的标记。支持字符或整数标签(应与 y 相同类型)或列表的手动标签（如果多输出使用），也可输入“most_frequent”将最频繁出现的 y 标记分配给离群值。默认 None，当检测到任何异常值时，将引发 ValueError 异常。
- metric_params：字典或 None，度量函数的附加关键字参数。默认 None。
- n_jobs：整数或 None 或-1，表示要为邻居搜索运行的并行作业的数量，None 表示 1，-1 表示所有，默认 None。

- NearestCentroid
  - NearestCentroid：NearestCentroid，最小中值距离分类算法，每个类由它的质心表示，测试样本被分类为具有最近的质心的类。
  - 模型名称：保存模型的名称。
  - Metric：选择在计算特性数组中实例之间的距离时使用的度量方法，默认 euclidean。
  - shrink_threshold：浮点数或 None，缩小中心体以去除特征的阈值，默认 None。

- LinearSVC

➢ LinearSVC：LinearSVC，线性支持向量分类算法，与参数kernel='linear'的SVC类似，使用liblinear实现，在惩罚和损失函数的选择上更灵活。

➢ 模型名称：保存模型的名称。

➢ penalty：指定处罚中使用的规范，包含l1和l2，其中l2是SVC中使用的标准，l1会导致系数向量稀疏化，默认l2。

➢ loss：指定损失函数，包含hinge和squared_hinge，其中hinge是标准SVM的损失函数，squared_hinge是hinge损失函数的平方，默认squared_hinge。

➢ Dual：选择求解对偶优化问题或原始优化问题的算法，当样本数大于特征数时，dual=False，默认True。

➢ tol：指定停止判据的公差，默认1e-4。

➢ C：正则化的强度与C成反比，必须严格为正，默认1.0。

➢ multi_class：如果y包含两个以上的类，则需要确定多类策略。包含ovr和crammer_singer,其中ovr训练n类一对多分类器,crammer_singer优化所有类的联合目标，默认ovr。

➢ fit_intercept：指定是否计算该模型的截距，如果设置为false，则不会在计算中使用截距(即数据应该已经居中)，默认True。

➢ intercept_scaling：当self.fit_intercept为True时，实例向量x变为[x，self.intercept_scaling]，即具有等于intercept_scaling的常量值的“合成”特征被附加到实例向量。 截距变为intercept_scaling * 合成特征权重注意！ 合成特征权重与所有其他特征一样经受l1 / l2正则化。 为了减小正则化对合成特征权重（并因此对截距）的影响，必须增加intercept_scaling。默认1。

➢ class_weight：设置类的参数，包含dict和balance，其中balance模式使用y的值自动调整与输入数据中的类频率成反比的权重，如果设置为None，所有类均使用1为权重，默认None。

➢ Verbose：指定是否启用详细输出，0表示不启用，其他正整数表示

启用，默认 0。

- random_state：正整数或随机数实例或 None，伪随机数生成器的种子，用于对数据进行变换以实现对偶坐标下降。如果是 int，则 random_state 是随机数生成器使用的种子， 如果是 RandomState 实例，则 random_state 是随机数生成器， 如果为 None，则随机数生成器是 np.random 使用的 RandomState 实例，默认 None。
- max_iter：要运行的最大迭代次数，默认 1000。

- SVC
  - SVC：SVC，C 支持向量分类算法，利用 libsvm 实现的支持向量机分类器。
  - 模型名称：保存模型的名称。
  - C：大于 0 的浮点数，惩罚参数，误差项的惩罚参数 C。C 值越小，对误分类的惩罚减小，允许容错，泛化能力较强，C 值越大，效果相反，默认 1.0。
  - Kernel：选择算法使用的核函数，包含 linear、poly、rbf、sigmoid、precomputed，其中 linear 表示线性函数，poly 表示多项式函数，rbf 表示 RBF 高斯核函数，sigmoid 表示 S 形曲线函数，默认 rbf。
  - Degree：整数，多项式 poly 函数的维度，选择其他核函数时会被忽略，默认 3。
  - gamma：auto 或浮点数或 scale，poly、rbf、sigmoid 核函数参数，默认 auto。
  - coef0：浮点数，核函数的常数项，针对 poly 和 sigmoid 核函数有效，默认 0.0。
  - Shrinking：是否采用 shrinking heuristic 方法，默认 True。
  - Probability：是否采用概率估计，默认 False。
  - tol：0 到 1 之间的小数，停止训练的误差值大小，默认 1e-3。
  - cache_size：整数，核函数缓存大小，默认 200。
  - class_weight：选择类别的权重设置方式，balanced 模式使用 y 值

自动调整输入数据中与类频率成反比的权重，None 表示所有的类都使用权重 1，默认 None。

- Verbose：选择是否启用详细输出，默认 False。
- max_iter：整数，指定在求解器中迭代的硬限制，-1 表示无限制，默认-1。
- decision_function_shape：选择框，选择返回的决策函数的形状，ovr 表示一对多，ovo 表示一对一，默认 ovr。
- random_state：整数，随机数的种子值，默认 None。

- NuSVC
  - NuSVC：NuSVC，Nu-支持向量分类算法，类似于 SVC，但是使用一个参数来控制支持向量的数量。
  - 模型名称：保存模型的名称。
  - nu：浮点数，训练误差分数的上界和支持向量分数的下界，在区间(0,1.0]内，默认 0.5。
  - Kernel：选择算法使用的核函数，包含 linear、poly、rbf、sigmoid、precomputed，其中 linear 表示线性函数，poly 表示多项式函数，rbf 表示 RBF 高斯核函数，sigmoid 表示 S 形曲线函数，默认 rbf。
  - Degree：整数，多项式 poly 函数的维度，选择其他核函数时会被忽略，默认 3。
  - gamma：auto 或浮点数或 scale，poly、rbf、sigmoid 核函数参数，默认 auto。
  - coef0：浮点数，核函数的常数项，针对 poly 和 sigmoid 核函数有效，默认 0.0。
  - shrinking：是否采用 shrinking heuristic 方法，默认 True。
  - Probability：是否采用概率估计，默认 False。
  - Tolerance：0 到 1 之间的小数，停止训练的误差值大小，默认 1e-3。
  - cache_size：整数，核函数缓存大小，默认 200。
  - class_weight：选择类别的权重设置方式，balanced 模式使用 y 值

自动调整输入数据中与类频率成反比的权重，None 表示所有的类都使用权重 1，默认 None。

- Verbose：选择是否启用详细输出，默认 False。
- max_iter：整整，指定在求解器中迭代的硬限制，-1 表示无限制，默认-1。
- decision_function_shape：选择框，选择返回的决策函数的形状，ovr 表示一对多，ovo 表示一对一，默认 ovr。
- random_state：整数或 None，随机数的种子值，默认 None。

- DecisionTreeClassifier
  - DecisionTreeClassifier：DecisionTreeClassifier，决策树分类算法。
  - 模型名称：保存模型的名称。
  - Criterion：选择测量分割质量的函数，包含 gini 和 entropy，gini 表示基尼系数，entropy 表示信息熵，默认 gini。
  - Splitter：选择每个节点的分割策略，包含 best 和 random，best 表示在所有特征中找最好的切分点，适合样本量不大的时候；random 表示在部分特征中找切分点，适合样本数据量非常大的时候，默认 best。
  - max_depth：正整数或 None，设置决策随机森林中的决策树的最大深度，深度越大，越容易过拟合，推荐树的深度为 5-20 之间，默认 None，None 表示不对决策树的最大深度作约束，直到每个叶子结点上的样本均属于同一类，或者少于 min_samples_leaf 参数指定的叶子结点上的样本个数。
  - min_samples_split：整数，设置分割内部节点所需的最小样本数，默认 2，参数为整数时应不小于 2，参数为浮点数参数范围为(0,1.0]。
  - min_samples_leaf：整数或者浮点数，设置叶节点所需的最小样本数，默认 1，参数为整数时应不小于 1，参数为浮点数时参数范围为(0,0.5]。
  - min_weight_fraction_leaf：浮点数，叶节点所需的(所有输入样本的)总权值的最小加权分数，参数范围为[0.0,0.5]。当不提供 sample_weight 时，样本的权重相等，默认 0.0。

- max_features：整数或 None 或 auto 或 sqrt 或 log2，指定寻找最佳分割时要考虑的特性的数量，None 表示将所有特征作为 max_features,auto 表示 max_features=sqrt(n_features)，sqrt 表示 max_features=sqrt(n_features)，log2 表示 max_features=log2(n_features)，默认 None。
- random_state：整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- max_leaf_nodes：正整数或 None，最大叶子节点数，通过限制该值可以防止过拟合，None 表示不限制最大的叶子节点数，默认 None。
- min_impurity_decrease：浮点数，设置节点分裂阈值，如果节点分裂导致杂质的减少大于或等于该值，该节点将被分裂，默认 0.0。
- class_weight：字典或字典组成的列表或 None，指定样本各类别的权重，为了防止训练集某些类别的样本过多导致训练的决策树过于偏向这些类别，默认 None。

● BernoulliNB

- BernoulliNB：BernoulliNB，伯努利朴素贝叶斯分类算法，用于处理二项分布问题。
- 模型名称：保存模型的名称。
- alpha：浮点数，拉普拉斯或利德斯通平滑的参数，如果设置为 0 则表示完全没有平滑，默认 1.0。
- binarize：浮点数或 None，用于对样本特征进行二值化(映射到布尔值)的阈值。如果没有，则假定输入已经由二进制向量组成，默认 0.0。
- fit_prior：选择是否学习类先验概率，False 表示使用统一的先验，默认 True。
- class_prior：类数组结构（形状为(n_classes,)）或 None，类的先验概率，如果指定，则不根据数据调整先验，None 表示自动根据数据来进行计算，默认 None。

- LogisticRegression
  - LogisticRegression：LogisticRegression，逻辑回归分类算法。
  - 模型名称：保存模型的名称。
  - penalty：指定处罚中使用的规范，包含 l1 和 l2，newton-cg、sag 和 lbfgs 求解器仅支持 l2，elasticnet 仅支持 saga 求解器，none 表示不适用求解器，默认 l2。
  - Dual：选择求解对偶优化问题或原始优化问题的算法，只适用于 liblinear 求解器且 penalty 为 l2 的情况，当样本数大于特征数时，dual=False，默认 False。
  - tol：0 到 1 之间的小数，停止训练的误差值大小，默认 1e-4。
  - C：正则化的强度与 C 成反比，必须严格为正，默认 1.0。
  - fit_intercept：选择是否应将常数(偏差或截距)添加到决策函数，默认 True。
  - intercept_scaling：当 self.fit_intercept 为 True 且使用 liblinear 求解器时，实例向量 x 变为[x，self.intercept_scaling]，即具有等于 intercept_scaling 的常量值的“合成”特征被附加到实例向量。为了减小正则化对合成特征权重（并因此对截距）的影响，必须增加 intercept_scaling。默认 1.0。
  - class_weight：设置类的参数，包含 dict 和 balance 和 None，其中 balance 模式使用 y 的值自动调整与输入数据中的类频率成反比的权重，如果设置为 None，所有类均使用 1 为权重，默认 None。
  - random_state：整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
  - solver：选择用于优化问题的算法，包含 liblinear、newton-cg、lbfgs、sag，对于小的数据集，建议选择 liblinear；而对于大的数据集，sag 和 saga 更快；对于多类问题，newton-cg、sag、saga 和 lbfgs 可以处理多项损失；newton-cg、lbfgs 和 sag 只处理 L2 处罚，而 liblinear 和 saga 处理 L1 处罚，默认 liblinear。
  - max_iter：整数，指定求解器收敛所需的最大迭代次数，只适用于

newton-cg, sag 和 lbfgs 算法。

- multi_class：ovr 表示每个标签都适合一个二进制问题，multinomial 表示即使数据是二进制的，损失最小化也是整个概率分布的多项式损失，当 solver 选择 liblinear 时，multinomial 不可用，默认 ovr。
- verbose：整数，对于 liblinear 和 lbfgs 求解器，将 verbose 设置为任何正数均表示启用详细输出，默认 0。
- warm_start：True 表示重用前一个调用的解决方案以适应初始化，False 表示擦除前一个解决方案。liblinear 求解器不可用，默认 False。
- n_jobs：当 multi_class=ovr，表示在类之间并行化时使用的 CPU 核数。当求解器被设置为 liblinear 时，不管是否指定了 multi_class，此参数都将被忽略。None 表示 1，-1 表示使用所有可用的求解器，默认 None。

- ExtraTreeClassifier
  - ExtraTreeClassifier：ExtraTreeClassifier，随机树分类算法。
  - 模型名称：保存模型的名称。
  - Criterion：选择测量分割质量的函数，包含 gini 和 entropy，gini 表示基尼系数，entropy 表示信息熵，默认 gini。
  - Splitter：选择每个节点的分割策略，包含 best 和 random，best 表示在所有特征中找最好的切分点，适合样本量不大的时候；random 表示在部分特征中找切分点，适合样本数据量非常大的时候，默认 random。
  - max_depth：正整数或 None，设置决策随机森林中的决策树的最大深度，深度越大，越容易过拟合，推荐树的深度为 5-20 之间，默认 None。
  - min_samples_split：整数，设置分割内部节点所需的最小样本数，默认 2，参数为整数时不小于 2，参数为浮点数时参数范围为(0,1.0]。
  - min_samples_leaf：整数，设置叶节点所需的最小样本数，默认 1，参数为整数时应不小于 1，参数为浮点数时参数范围为(0,0.5]。
  - min_weight_fraction_leaf：浮点数，叶节点所需的(所有输入样本的)总权值的最小加权分数。当不提供 sample_weight 时，样本的权重相等，默认 0.0，参数范围为[0.0,0.5]。

- max_features：整数或 None 或 auto 或 sqrt 或 log2，指定寻找最佳分割时要考虑的特性的数量，None 表示将所有特征作为 max_features,auto 表示 max_features=sqrt(n_features)，sqrt 表示 max_features=sqrt(n_features)，log2 表示 max_features=log2(n_features)，默认 None。
- random_state：整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- max_leaf_nodes：正整数或 None，最大叶子节点数，通过限制该值可以防止过拟合，None 表示不限制最大的叶子节点数，默认 None。
- min_impurity_decrease：浮点数，设置节点分裂阈值，如果节点分裂导致杂质的减少大于或等于该值，该节点将被分裂，默认 0.0。
- ccp_alpha：非负浮点数，最小剪枝系数，该参数用来限制树过拟合的剪枝参数，ccp_alpha=0 时，决策树不剪枝；ccp_alpha 越大，越多的节点被剪枝，默认 0.0。
- class_weight：字典或字典组成的列表或 None，指定样本各类别的权重，为了防止训练集某些类别的样本过多导致训练的决策树过于偏向这些类别，默认 None。

- GaussianNB
  - GaussianNB：GaussianNB，高斯朴素贝叶斯分类算法。
  - 模型名称：保存模型的名称。
  - prior：类数组结构（形状为(n_classes,)）或 None，类的先验概率，如果指定，则不根据数据调整先验，None 表示自动根据数据来进行计算，默认 None。
  - var_smoothing：浮点数，所有特征的最大方差的一部分，添加到方差中以保持计算稳定性，默认 1e-9。

- KNeighborsRegressor
  - KNeighborsRegressor：KNeighborsRegressor，K 近邻回归。

- 模型名称：保存模型的名称。
- n_neighbors：正整数，kneighbors 查询默认使用的邻居数，默认 5。
- weights：选择用于预测的权重函数，包括 uniform 和 distance。其中 uniform 表示统一的权重,即在每一个邻居区域里的点的权重都是一样的，distance 表示权重点等于他们距离的倒数，使用此函数，更近的邻居对于所预测的点的影响更大。默认 uniform。
- Algorithm:选择最近邻所用的算法,包括 auto、ball_tree、kd_tree、brute,其中 auto 表示会根据传递给 fit 的数据自动决定使用最合适的算法，ball_tree 表示使用 BallTree 算法，kd_tree 表示使用 KDTree 算法，brute 表示使用暴力搜索。默认 auto。
- leaf_size：正整数，指定叶子数量。叶子大小传递给 BallTree 或 KDTree，这可能会影响构造和查询的速度，以及存储树所需的内存，最优值取决于问题的性质。默认 30。
- p：整数，闵科夫斯基距离指数或超参数。p=1 等价于使用曼哈顿距离（l1），p=2 等价于欧式距离(l2)，对于任何 p ，使用的是闵科夫斯基距离（l_p）。默认 2。
- metric：选择用于树的距离度量，如果和 p=2 一起使用相当于使用标准欧几里德距离度量，默认 minkowski，目前仅支持实数值向量空间的度量。
- metric_params：字典或 None，度量函数的附加关键字参数。默认 None。
- n_jobs：整数或 None 或-1，表示要为邻居搜索运行的并行作业的数量，None 表示 1，-1 表示所有搜索到的作业数，默认 None。

- RadiusNeighborsRegressor
  - RadiusNeighborsRegressor：RadiusNeighborsRegressor，基于固定半径内邻居的回归。
  - 模型名称：保存模型的名称。
  - Radius：浮点数，radius_neighbors 查询默认使用的参数空间范围，

默认 1.0。

- Weight：选择用于预测的权重函数，包括 uniform 和 distance。其中 uniform 表示统一的权重,即在每一个邻居区域里的点的权重都是一样的，distance 表示权重点等于他们距离的倒数，使用此函数，更近的邻居对于所预测的点的影响更大。默认 uniform。
- Algorithm:选择最近邻所用的算法,包括 auto、ball_tree、kd_tree、brute,其中 auto 表示会根据传递给 fit 的数据自动决定使用最合适的算法，ball_tree 表示使用 BallTree 算法，kd_tree 表示使用 KDTree 算法，brute 表示使用暴力搜索。默认 auto。
- leaf_size：整数，指定叶子数量。叶子大小传递给 BallTree 或 KDTree，这可能会影响构造和查询的速度，以及存储树所需的内存，最优值取决于问题的性质。默认 30。
- p：整数，闵科夫斯基距离指数或超参数。p=1 等价于使用曼哈顿距离（l1），p=2 等价于欧式距离(l2)，对于任何 p ，使用的是闵科夫斯基距离（l_p）。默认 2。
- metric：选择用于树的距离度量，如果和 p=2 一起使用相当于使用标准欧几里德距离度量，默认 minkowski，目前仅支持实数值向量空间的度量。
- metric_params：字典或 None，度量函数的附加关键字参数。默认 None。
- n_jobs：整数或 None 或-1，表示要为邻居搜索运行的并行作业的数量，None 表示 1，-1 表示所有，默认 None。

● LinearSVR

- LinearSVR：LinearSVR，线性支持向量回归。
- 模型名称：保存模型的名称。
- Epsilon：浮点数，损失函数选择 epsilon-insensitive 时的参数，该参数依赖于目标变量 y 的大小，如果不确定则设置为 0.0，默认 0.1。
- tol：0 到 1 之间的小数，停止训练的误差值大小，默认 1e-4。

- C：大于 0 的浮点数，误差项的惩罚参数 C。惩罚是 12 的平方，该值越大，表示使用的正则化越少，默认 1.0。
- Loss：选择损失函数，epsilon_insensitive（标准 SVR）是 11 损失，squared_epsilon_insensitive 是 12 损失，默认 epsilon_insensitive。
- fit_intercept：是否计算该模型的截距，False 表示不会在计算中使用截距，默认 True。
- intercept_scaling：当 self.fit_intercept 为 True 且使用 liblinear 求解器时，实例向量 x 变为[x，self.intercept_scaling]，即具有等于 intercept_scaling 的常量值的“合成”特征被附加到实例向量。为了减小正则化对合成特征权重（并因此对截距）的影响，必须增加 intercept_scaling。默认 1.0。
- Dual：选择求解对偶优化问题或原始优化问题的算法，True 表示求解对偶优化问题，当样本数大于特征数时，dual=False，默认 True。
- Verbose：整数，是否启用详细输出，默认 0。
- random_state：整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- max_iter：整数，要运行的最大迭代次数，默认 1000。

- SVR
  - SVR：SVR，Epsilon-支持向量回归，模型中的自由参数为 C 和 epsilon。
  - 模型名称：保存模型的名称。
  - Kernel：选择算法使用的核函数，包含 linear、poly、rbf、sigmoid、precomputed，其中 linear 表示线性函数，poly 表示多项式函数，rbf 表示 RBF 高斯核函数，sigmoid 表示 S 形曲线函数，默认 rbf。
  - Degree：整数，多项式 poly 函数的维度，选择其他核函数时会被忽略，默认 3。
  - gamma：auto 或浮点数或 scale，poly、rbf、sigmoid 核函数参数，默认 auto。

- coef0：浮点数，核函数的常数项，针对 poly 和 sigmoid 核函数有效，默认 0.0。
- tol：0 到 1 之间的小数，停止训练的误差值大小，默认 1e-3。
- C：大于 0 的浮点数，误差项的惩罚参数 C，默认 1.0。
- epsilon：浮点数，惩罚参数，默认 0.1。
- Shrinking：是否采用 shrinking heuristic 方法，默认 True。
- cache_size：整数，核函数缓存大小（以 MB 为单位），默认 200。
- Verbose：是否启用详细输出，默认 False。
- max_iter：整数，指定在求解器中迭代的硬限制，-1 表示无限制，默认-1。

● NuSVR

- NuSVR：NuSVR，Nu-支持向量回归，使用参数 nu 代替 SVR 的 epsilon 参数控制支持向量的数量。
- 模型名称：保存模型的名称。
- nu：浮点数，训练误差分数的上界和支持向量分数的下界，在区间(0,1.0]内，默认 0.5。
- C：大于 0 的浮点数，误差项的惩罚参数 C，默认 1.0。
- Kernel：选择算法使用的核函数，包含 linear、poly、rbf、sigmoid、precomputed，其中 linear 表示线性函数，poly 表示多项式函数，rbf 表示 RBF 高斯核函数，sigmoid 表示 S 形曲线函数，默认 rbf。
- Degree：正整数，多项式 poly 函数的维度，选择其他核函数时会被忽略，默认 3。
- gamma：auto 或浮点数或 scale，poly、rbf、sigmoid 核函数参数，默认 auto。
- coef0：浮点数，核函数的常数项，针对 poly 和 sigmoid 核函数有效，默认 0.0。
- Shrinking：是否采用 shrinking heuristic 方法，默认 True。
- tol：0 到 1 之间的浮点数或科学计数法，停止训练的误差值大小，

默认 1e-3。

- cache_size：正整数，核函数缓存大小（以 MB 为单位），默认 200。
- Verbose：是否启用详细输出，默认 False。
- max_iter：整数，指定在求解器中迭代的硬限制，-1 表示无限制，默认-1。

- DecisionTreeRegressor
  - DecisionTreeRegressor：DecisionTreeRegressor，决策树回归。
  - 模型名称：保存模型的名称。
  - Criterion：选择测量分割质量的函数，支持的标准有：均方误差的'squared_error'，方差缩减作为特征选择标准，使用每个终端节点的平均值最小化 L2 损失，'friedman_mse'，表示费尔德曼均方误差算法，'absolute_error'的平均绝对误差，使用每个终端节点的中位数最小化 L1 损失，'poisson'，表示泊松偏差算法。默认'squared_error'。
  - Splitter：选择每个节点的分割策略，包含 best 和 random，best 表示选择最好的特征进行分割；random 表示随机选择特征进行分割，默认 best。
  - max_depth：正整数或 None，设置决策树的最大深度，None 表示展开节点直到所有叶子都是纯的，或者直到所有叶子包含的样本小于 min_samples_split，默认 None。
  - min_samples_split：正整数，设置分割内部节点所需的最小样本数，默认 2，当入参为整数时不小于 2，入参为浮点数时参数范围为(0,1.0]。
  - min_samples_leaf：正整数，设置叶节点所需的最小样本数，默认 1，参数为整数时应不小于 1，参数为浮点数时参数范围为(0,0.5]。
  - min_weight_fraction_leaf：浮点数，叶节点所需的(所有输入样本的)总权值的最小加权分数。当不提供 sample_weight 时，样本的权重相等，默认 0.0，参数范围为[0.0，0.5]。
  - max_features：整数或者浮点数或 None 或 auto 或 sqrt 或 log2，指定寻找最佳分割时要考虑的特性的数量，None 表示将所有特征作为

max_features,auto 表示 max_features=sqrt(n_features)，sqrt 表示 max_features=sqrt(n_features)，log2 表示 max_features=log2(n_features)，默认 None。

- random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- max_leaf_nodes：正整数或 None，最大叶子节点数，通过限制该值可以防止过拟合，None 表示不限制最大的叶子节点数，默认 None。
- min_impurity_decrease：浮点数，设置节点分裂阈值，如果节点分裂导致杂质的减少大于或等于该值，该节点将被分裂，默认 0.0。
- ccp_alpha：非负浮点数，最小剪枝系数，该参数用来限制树过拟合的剪枝参数，ccp_alpha=0 时，决策树不剪枝；ccp_alpha 越大，越多的节点被剪枝，默认 0.0。

- ExtraTreeRegressor
  - ExtraTreeRegressor：ExtraTreeRegressor，随机树回归。
  - 模型名称：保存模型的名称。
  - Criterion：选择测量分割质量的函数，包含 squared_error 和 friedman_mse，squared_error 表示均方差，利用末端节点的平均最小化 12 损失，mae 表示平均绝对误差，利用每个末端节点的中值最小化 11 损失，默认 squared_error。
  - Splitter：选择每个节点的分割策略，包含 best 和 random，best 表示选择最好的特征进行分割；random 表示随机选择特征进行分割，默认 best。
  - max_depth：正整数或 None，设置决策树的最大深度，None 表示展开节点直到所有叶子都是纯的，或者直到所有叶子包含的样本小于 min_samples_split，默认 None。
  - min_samples_split：正整数，设置分割内部节点所需的最小样本数，默认 2，参数为整数时不小于 2，参数为浮点型时参数范围为(0,1.0]。
  - min_samples_leaf：正整数，设置叶节点所需的最小样本数，默认 1，

参数为整数时应不小于 1，参数为浮点数时参数范围为(0,0.5]。

- min_weight_fraction_leaf：浮点数，叶节点所需的(所有输入样本的)总权值的最小加权分数。当不提供 sample_weight 时，样本的权重相等，默认 0.0，参数范围为[0.0,0.5]。
- max_features：整数或 None 或 auto 或 sqrt 或 log2，指定寻找最佳分割时要考虑的特性的数量，None 表示将所有特征作为 max_features,auto 表示 max_features=sqrt(n_features)，sqrt 表示 max_features=sqrt(n_features)，log2 表示 max_features=log2(n_features)，默认 None。
- random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- min_impurity_decrease：浮点数，设置节点分裂阈值，如果节点分裂导致杂质的减少大于或等于该值，该节点将被分裂，默认 0.0。
- max_leaf_nodes：正整数或 None，最大叶子节点数，通过限制该值可以防止过拟合，None 表示不限制最大的叶子节点数，默认 None。
- ccp_alpha：非负浮点数，最小剪枝系数，该参数用来限制树过拟合的剪枝参数，ccp_alpha=0 时，决策树不剪枝；ccp_alpha 越大，越多的节点被剪枝，默认 0.0。

- ARDRegression
  - ARDRegression：ARDRegression，贝叶斯自相关回归。
  - 模型名称：保存模型的名称。
  - n_iter：正整数，指定最大迭代次数，默认 300。
  - tol：0 到 1 之间的小数，停止训练的误差值大小，如果 w 已经收敛，则停止算法，默认 0.001。
  - alpha_1：浮点数，超参数，在 alpha 参数之前的 Gamma 分布的形状参数，默认 1e-6。
  - alpha_2：浮点数，超参数：在 alpha 参数之前的 Gamma 分布的反比例参数（速率参数），默认 1e-6。

- lambda_1：浮点数，超参数，在 lambda 参数之前的 Gamma 分布的形状参数，默认 1e-6。
- lambda_2：浮点数，超参数：在 lambda 参数之前的 Gamma 分布的反比例参数（速率参数），默认 1e-6。
- compute_score：True 表示在模型的每一步计算目标函数，False 则不计算，默认 False。
- threshold_lambda：浮点数，从计算中去除(修剪)高精度权值的阈值，默认 10000.0。
- fit_intercept：是否计算该模型的截距，False 表示不会在计算中使用截距，默认 True。
- normalize：当 fit_intercept 设置为 False 时，将忽略该参数。若为真，则回归前对回归量 X 进行归一化处理，默认 False。
- copy_X：True 表示复制 X，False 表示覆盖 X，默认 True。
- verbose：是否启用详细输出，默认 False。

- BayesianRidge
  - BayesianRidge：BayesianRidge，贝叶斯岭回归。
  - 模型名称：保存模型的名称。
  - n_iter：正整数，指定最大迭代次数，默认 300。
  - tol：0 到 1 之间的小数，停止训练的误差值大小，如果 w 已经收敛，则停止算法，默认 0.001。
  - alpha_1：浮点数，超参数，在 alpha 参数之前的 Gamma 分布的形状参数，默认 1e-6。
  - alpha_2：浮点数，超参数：在 alpha 参数之前的 Gamma 分布的反比例参数（速率参数），默认 1e-6。
  - lambda_1：浮点数，超参数，在 lambda 参数之前的 Gamma 分布的形状参数，默认 1e-6。
  - lambda_2：浮点数，超参数：在 lambda 参数之前的 Gamma 分布的反比例参数（速率参数），默认 1e-6。

- compute_score：True 表示在模型的每一步计算目标函数，False 则不计算，默认 False。
- fit_intercept：是否计算该模型的截距，False 表示不会在计算中使用截距，默认 True。
- normalize：当 fit_intercept 设置为 False 时，将忽略该参数。若为真，则回归前对回归量 X 进行归一化处理，默认 False。
- copy_X：True 表示复制 X，False 表示覆盖 X，默认 True。
- verbose：是否启用详细输出，默认 False。

- ElasticNet
  - ElasticNet：ElasticNet，弹性网络回归算法。
  - 模型名称：保存模型的名称。
  - alpha：浮点数，乘以惩罚项的常数，默认 1.0。
  - l1_ratio：介于 0 到 1 之间的浮点数，弹性网络的混合参数，l1_ratio=0 表示 l2 惩罚，l1_ratio=1 表示 l1 惩罚，0 到 1 之间表示 l1 和 l2 的组合，默认 0.5。
  - fit_intercept：是否计算该模型的截距，False 表示不会在计算中使用截距，默认 True。
  - normalize：当 fit_intercept 设置为 False 时，将忽略该参数。若为真，则回归前对回归量 X 进行归一化处理，默认 False。
  - precompute：是否使用预先计算好的 Gram 矩阵来加速计算，默认 False。
  - max_iter：正整数，要运行的最大迭代次数，默认 1000。
  - copy_X：True 表示复制 X，False 表示覆盖 X，默认 True。
  - tol：浮点数，优化公差，默认 0.0001。
  - warm_start：True 表示重用前一个调用的解决方案以适应初始化，False 表示擦除前一个解决方案，默认 False。
  - positive：当设置为 True 时，强制系数为正，默认 False。
  - random_state：正整数或 None，随机数的种子值，如果为 None，随

机数生成器就是 np.random 使用的 RandomState 实例，默认 None。

- selection：random 表示每一次迭代都会更新一个随机系数，cyclic 表示按顺序遍历特性，默认 cyclic。

- Lars
  - Lars：Lars，最小角回归。
  - 模型名称：保存模型的名称。
  - fit_intercept：是否计算该模型的截距，False 表示不会在计算中使用截距，默认 True。
  - verbose：是否启用详细输出，默认 False。
  - normalize：当 fit_intercept 设置为 False 时，将忽略该参数。若为真，则回归前对回归量 X 进行归一化处理，默认 True。
  - precompute：是否使用预先计算好的 Gram 矩阵来加速计算，默认 False。
  - n_nonzero_coefs：正整数，非零系数的目标数，np.inf 暂时不支持，默认 500。
  - eps：浮点数，计算 Cholesky 对角因子的机器精度规范，默认 2.220446049250313e-16。
  - copy_X：True 表示复制 X，False 表示覆盖 X，默认 True。
  - fit_path：True 表示将完整路径存储在 coef_path_属性中，默认 True。
  - random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。

- Lasso
  - Lasso：Lasso，用 l1 先验作为正则化器(即 Lasso)的线性模型。
  - 模型名称：保存模型的名称。
  - alpha：浮点数，乘以 l1 惩罚项的常数，默认 1.0。
  - fit_intercept：是否计算该模型的截距，False 表示不会在计算中

使用截距，默认 True。

- normalize：当 fit_intercept 设置为 False 时，将忽略该参数。若为真，则回归前对回归量 X 进行归一化处理，默认 False。
- precompute：是否使用预先计算好的 Gram 矩阵来加速计算，默认 False。
- copy_X：True 表示复制 X，False 表示覆盖 X，默认 True。
- max_iter：正整数，要运行的最大迭代次数，默认 1000。
- tol：浮点数，优化公差，默认 0.0001。
- warm_start：True 表示重用前一个调用的解决方案以适应初始化，False 表示擦除前一个解决方案，默认 False。
- positive：当设置为 True 时，强制系数为正，默认 False。
- random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- selection：random 表示每一次迭代都会更新一个随机系数，cyclic 表示按顺序遍历特性，默认 cyclic。

- LinearRegression
  - LinearRegression：LinearRegression，普通最小二乘线性回归。
  - 模型名称：保存模型的名称。
  - fit_intercept：是否计算该模型的截距，False 表示不会在计算中使用截距，默认 True。
  - normalize：当 fit_intercept 设置为 False 时，将忽略该参数。若为真，则回归前对回归量 X 进行归一化处理，默认 False。
  - copy_X：True 表示复制 X，False 表示覆盖 X，默认 True。
  - n_jobs：正整数或 None 或-1，要用于计算的作业数，None 表示 1，-1 表示所有的作业数，默认 None。

- AgglomerativeClustering
  - AgglomerativeClustering：AgglomerativeClustering，聚合聚类。

- 模型名称：保存模型的名称。
- n_clusters：正整数或 None，要查找的聚类数量，distance_threshold 为 None 时该值必须设置为 None，默认 2。
- Affinity：选择用于计算 linkage 的度量方法，包含 euclidean、precomputed、cosine、l1、l2 和 manhattan，如果 linkage 为 ward，只能选用 euclidean，默认 euclidean。
- Memory：用于缓存计算树的输出，None 表示不进行缓存，如果给定一个字符串，即缓存目录的路径，默认 None。
- Connectivity：类数组格式或 None，连接矩阵，根据给定的数据结构为每个样本定义相邻的样本，默认 None。
- compute_full_tree：是否在 n_clusters 处提前停止树的构造，默认 auto。
- Linkage：选择使用哪种连接标准，决定了观察集之间使用的距离，该算法将合并最小化该准则的簇对，ward 最小化簇合并的方差，average 使用两组观测值距离的平均值，complete 使用两个集合的所有观测值之间的最大距离，single 使用两个集合的所有观测值之间的最小距离，默认 ward。

- KMeans
  - KMeans：KMeans，K 均值聚类。
  - 模型名称：保存模型的名称。
  - n_clusters：正整数，要形成的簇数以及要生成的质心数，默认 8。
  - init：选择初始化方法，k-means++表示智能选择 k 均值聚类的初始聚类中心加快收敛速度，random 表示从初始质心的数据中随机选择 k 个观测值，暂时不支持输入 ndarray，默认 k-means++。
  - n_init：正整数，k-means 算法在不同的质心种子下运行的时间，默认 10。
  - max_iter：正整数，k-means 算法单次运行的最大迭代次数，默认 300。
  - tol：浮点数，惯性收敛的相对公差，默认 1e-4。

- verbose：是否启用详细输出，0 表示不输出，默认 0。
- random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
- copy_x：True 表示复制 X，False 表示覆盖 X，默认 True。
- Algorithm：选择 K-means 使用的算法，full 表示经典的 EM-style 算法，elkan 使用三角不等式有更高的效率（不支持稀疏数据），auto 会自动对稀疏数据选择 full，对稠密数据选择 elkan，默认 auto。

- SpectralClustering
  - SpectralClustering：SpectralClustering，谱聚类，将聚类应用于标准化拉普拉斯投影。
  - 模型名称：保存模型的名称。
  - n_clusters：正整数，投影子空间的维数，默认 8。
  - eigen_solver：选择特征值分解策略，默认 None。
  - random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
  - n_init：正整数，k-means 算法在不同的质心种子下运行的时间，默认 10。
  - gamma：非负数，rbf、poly、sigmoid、laplacian 和 chi2 的核系数，affinity=nearest_neighbors 时忽略，默认 1.0。
  - affinity：选择构建 affinity 矩阵的方法，nearest_neighbors 通过计算最近邻的图来构造 affinity 矩阵，rbf 利用径向基函数(RBF)核构造 affinity 矩阵，precomputed 将 X 解释为一个预先计算的 affinity 和矩，默认 rbf。
  - n_neighbors：正整数，使用最近邻法构造 affinity 矩阵时要使用的近邻数，当 affinity=rbf 时忽略，默认 10。
  - eigen_tol：浮点数，eigen_solver=arpack 是拉普拉斯矩阵特征分解的停止准则，默认 0.0。
  - assign_labels：选择用于在嵌入空间中分配标签的策略，默认

kmeans。

- degree：浮点数，多项式核的维度，其他内核忽略，默认 3。
- coef0：浮点数，polynomial 和 sigmoid 核函数的零系数，其他内核忽略，默认 1。
- kernel_params：None，核参数，可忽略，默认 None。
- n_jobs：正整数或 None 或-1，要用于并行的作业数，None 表示 1，-1 表示所有，默认 None。

● DBSCAN

- DBSCAN：DBSCAN，基于密度的空间聚类，从向量阵列或距离矩阵执行 DBSCAN 聚类。
- 模型名称：保存模型的名称。
- eps：正浮点数，两个样本之间的最大距离，其中一个被认为是在另一个的邻域内，默认 0.5。
- min_samples：正整数，将一个点视为一个核心点的邻域内样本的数量(或总重量)，默认 5。
- Metric：选择在计算特征数组中实例之间的距离时使用的度量，默认 euclidean。
- metric_params：字典或 None，度量函数的附加关键字参数。默认 None。
- Algorithm：选择将被最近邻模块用来计算点距离和找到最近邻的算法，默认 auto。
- leaf_size：正整数，传递给 BallTree 或 cKDTree 的叶子大小，默认 30。
- p：浮点数或 None，用来计算点之间距离的 Minkowski 距离的幂，默认 None。
- n_jobs：正整数或 None 或-1，要用于计算的作业数，None 表示 1，-1 表示左右，默认 None。

- MiniBatchKMeans
  - MiniBatchKMeans：MiniBatchKMeans，小批量 K 均值聚类。
  - 模型名称：保存模型的名称。
  - n_clusters：正整数，要形成的簇数以及要生成的质心数，默认 8。
  - init：选择初始化方法，k-means++表示智能选择 k 均值聚类的初始聚类中心加快收敛速度，random 表示从初始质心的数据中随机选择 k 个观测值，暂时不支持输入 ndarray，默认 k-means++。
  - max_iter：正整数，要运行的最大迭代次数，默认 100。
  - batch_size：正整数，最小批次大小，默认 100。
  - verbose：是否启用详细输出，0 表示不输出，默认 0。
  - compute_labels：选择是否在小批量优化收敛到合适的程度后计算整个数据集的标签分配和惯性，默认 True。
  - random_state：正整数或 None，随机数的种子值，如果为 None，随机数生成器就是 np.random 使用的 RandomState 实例，默认 None。
  - tol：浮点数，根据中心位置变化的平滑、方差标准化的相对中心变化来控制早期停止，默认 0.0。
  - max_no_improvement：正整数，根据连续的小批量数量控制提前停止，默认 10。
  - init_size：正整数或 None，初始化大小，通过在数据的随机子集上运行批处理 KMeans 进行初始化，如果为 None，则 init_size= 3 * batch_size，默认 None。
  - n_init：正整数，尝试随机初始化的次数，默认 3。
  - reassignment_ratio：浮点数，控制要重新分配的中心的最大计数的分数，默认 0.01。

- IsolationForest
  - IsolationForest：IsolationForest，孤立森林。
  - 模型名称：保存模型的名称。
  - n_estimators：正整数，构建多少个 itree，指定该森林中生成的随

机树数量，默认 100。

- max_samples：采样数，auto 是 256，浮点数或者 auto，默认 auto。
- contamination：浮点数或者'auto'，异常数据占给定的数据集的比例，就是数据集中污染的数量，定义该参数值的作用是在决策函数中定义阈值。如果设置为'auto'，则决策函数的阈值就和论文一样。默认值为'auto'。
- max_features：正整数，最大特征数，指定从总样本 X 中抽取来训练每棵树 iTree 的属性的数量，默认只使用一个属性，默认为 1。
- bootstrap：构建 Tree 时，下次是否替换采样，为 True 为替换，则各个树可放回地对训练数据进行采样；为 False 为不替换，即执行不放回的采样，默认 False。
- n_jobs：正整数或 None，在运行 fit() 和 predict() 函数时并行运行的作业数量。除了在 joblib.parallel_backend 上下文的情况下，None 表示为 1，设置为 -1 则表示使用所有可以使用的处理器，默认 None。
- random_state：整数或者 None，如果设置为 int 常数，则该 random_state 参数值是用于随机数生成器的种子；如果设置为 None，则该随机数生成器就是使用在 np.random 中 RandomState 实例.默认 0.0
- verbose：整数，控制树构建过程的冗长性，默认 0。
- warm_start：当设置为 TRUE 时，重用上一次调用的结果去 fit，添加更多的树到上一次的森林 1 集合中；否则就 fit 一整个新的森林，默认 False。

3.3.2 视觉算法

- 图像分类算法
  - 图像分类算法：图像分类训练。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 VGG16，则保存为 VGG16；组件用于预测时，该属性没有实际意义。
  - 3D 可视化：是否开启 3D 可视化，3D 可视化仅支持深度学习模型且为 Sequential 式模型，其他类型的模型暂不支持。

- 分类模型：图像分类算法常用来判断图像内容所属的类别，每张图片输出一个类别。该组件提供了常用的 Resnet、Inception 系列、mobilenet 系列模型，并提供了基于 imagenet 预训练的模型，方便用户 finetune。默认情况下，使用 'VGG16'。
- weights：None 代表随机初始化，'imagenet'代表加载在 ImageNet 上预训练的权值。
- input_shape：输入尺寸元组，仅当 include_top=False 时有效，否则输入形状必须是 (244, 244, 3)（对于 channels_last 数据格式），或者 (3, 244, 244)（对于 channels_first 数据格式）。它必须拥有 3 个输入通道，且宽高必须不小于 32。例如 (200, 200, 3) 是一个合法的输入尺寸，默认为 None。
- pooling：当 include_top 为 False 时，该参数指定了特征提取时的池化方式。None 代表不池化，直接输出最后一层卷积层的输出，该输出是一个四维张量；'avg' 代表全局平均池化（GlobalAveragePooling2D），相当于在最后一层卷积层后面再加一层全局平均池化层，输出是一个二维张量。
- classes：正整数，图片分类的类别数，仅当 include_top 为 True 并且不加载预训练权值时可用，默认 1000。
- 迭代次数：训练模型的迭代次数，默认为 100。
- 损失函数：模型训练的损失函数，暂时不支持 Reduction 函数。
- 批次大小：一次训练所选取的样本数。
- metrics：metrics 函数，暂时不支持 MeanRelativeError，MeanIoU,PrecisionAtRecall，RecallAtPrecision，SensitivityAtSpecificity 和 SpecificityAtSensitivity。
- 优化器：设置模型优化器。
- 学习率调整策略：设置学习率调整策略：None 表示不设置任何学习策略，学习率为固定值；CosineDecay，余弦衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),alpha(最小学习率值作为 initial_learning_rate 的一部

分);CosineDecayRestarts，包含重新启动的余弦衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),first_decay_steps(要衰减的步数),t_mul(用于导出 i-th 周期内的迭代次数),m_mul(用于导出 i-th 周期的初始学习率),alpha(最小学习率值作为 initial_learning_rate 的一部分);ExponentialDecay，指数衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),decay_rate(衰减率),staircase(如果 True 以离散间隔衰减学习率);InverseTimeDecay，反时间衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),decay_rate(衰减率),staircase(是否在离散梯度中使用衰减，而不是在连续梯度);PiecewiseConstantDecay，分段常数衰减策略，与该策略相关的参数有：boundaries(Tensors 或 ints 或 floats 的列表具有严格增加的条目,并且所有元素具有与优化器步骤相同的类型),values(Tensor 或 float 或 int 列表，指定由 boundaries 定义的间隔的值，它应该比 boundaries 多一个元素，并且所有元素都应该具有相同的类型);PolynomialDecay，多项式学习能力衰减策略，与该策略相关的参数有：initial_learning_rate(初始学习率),decay_steps(衰减步数),end_learning_rate(最小的最终学习率),power(多项式的幂),cycle(是否应该循环超过 decay_steps)。

- learning_rate：设置固定学习率，0 到 1 之间的浮点数，默认 0.001。
- initial_learning_rate：初始学习率，0 到 1 之间的浮点数，默认 0.1。
- decay_steps：衰减步数，正整数，默认 1。
- alpha：最小学习率值作为 initial_learning_rate 的一部分，0 到 1 之间的浮点数，默认 0.0。
- first_decay_steps：衰减步数，正整数，默认 1。
- t_mul：用于导出 i-th 周期内的迭代次数，非负浮点数，默认 2.0。
- m_mul：用于导出 i-th 周期的初始学习率，非负浮点数，默认 1.0。
- decay_rate：衰减率，非负浮点数，默认 0.5。
- staircase：是否在离散梯度中使用衰减，而不是在连续梯度，默认

'False'。

- boundaries：严格递增的由浮点型或者整型组成的列表，默认[100000,110000]。
- values：指定由 boundaries 定义的间隔值，它应该比 boundaries 多一个元素，并且所有元素都应该具有相同的类型，默认[1.0,0.5,0.1]。
- end_learning_rate：最小的最终学习率，非负浮点数，默认 0.5。
- power：多项式的幂，非负浮点数，默认 1.0。
- cycle：是否应该循环超过 decay_steps，默认'False'。
- rho：Adadelta 梯度平方移动均值的衰减率，非负浮点数，默认 0.95。
- epsilon：epsilon，防止除 0 错误，0 到 1 之间的浮点数或者科学计数法，默认 1e-07。
- initial_accumulator_value：初始梯度累加和，非负浮点数，默认 0.1。
- beta_1：0 到 1 之间，非常接近 1 的浮点数，默认 0.9。
- beta_2：0 到 1 之间，非常接近 1 的浮点数，默认 0.999。
- amsgrad：是否应用此算法的 AMSGrad 变种，默认'False'。
- centered：如果 True，则通过梯度的估计方差对梯度进行归一化；如果为 False，则通过非居中的第二时刻。将此设置为 True 可能有助于训练，但在计算和内存方面稍微贵一些，默认'False'。
- momentum：用于加速 SGD 在相关方向上前进，并抑制震荡，非负浮点数，默认 0.0。
- nesterov：是否使用 Nesterov 动量，默认'False'。
- beta_2：浮点值，必须小于或等于零，控制在训练期间学习率如何降低，使用零表示固定的学习率，默认-0.5。
- l1_regularization_strength：浮点值，必须大于或等于零。默认为 0.0。
- l2_regularization_strength：浮点值，必须大于或等于零。默认为 0.0。
- l2_shrinkage_regularization_strength：浮点值，必须大于或等

于零。这与上面的 L2 不同，因为上面的 L2 是一个稳定惩罚，而这个 L2 收缩是一个幅度惩罚。当输入稀疏时，收缩只会发生在活动权重上。默认为 0.0。

- beta：浮点值，beta 值。默认为 0.0。

- ssd
  - ssd：ssd 目标识别算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 ssd，模型名称保存为 ssd;组件用于预测时，该属性没有实际意义。
  - weights：None 代表随机初始化，'VOC2007' 代表加载在 VOC2007 上预训练的权值。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- yolov3
  - yolov3：yolov3 目标识别算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 yolov3，则模型名称保存为 yolov3;组件用于预测时，该属性没有实际意义。
  - weights：None 代表随机初始化，'coco' 代表加载在 coco 上预训练的权值。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- yolov4
  - yolov4：yolov4 目标识别算法。

- 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 yolov4，模型名称保存为 yolov4;组件用于预测时，该属性没有实际意义。
- weights：None 代表随机初始化，'coco' 代表加载在 coco 上预训练的权值。
- 迭代次数：训练模型的迭代次数，默认为 100。
- 批次大小：一次训练所选取的样本数。
- learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

● yolov5

- yolov5:yolov5 目标识别算法。
- 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 yolov5，模型名称保存为 yolov5；组件用于预测时，该属性没有实际意义。
- weights：None 代表随机初始化，'coco' 代表加载在 coco 上预训练的权值。
- backbone：表示使用的 yolov5 的版本。
- 迭代次数：训练模型的迭代次数，默认为 100。
- 批次大小：一次训练所选取的样本数。
- learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

● m2det

- m2det：m2det 目标识别算法。
- 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 m2det，模型名称保存为 m2det；组件用于预测时，该属性没有实际意义。
- weights：None 代表随机初始化，'VOC2007' 代表加载在 VOC2007 上预训练的权值。
- 迭代次数：训练模型的迭代次数，默认为 100。

- 批次大小：一次训练所选取的样本数。
- learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- faster_rcnn
  - faster_rcnn：faster_rcnn 目标识别算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 faster_rcnn，模型名称保存为 faster_rcnn；组件用于预测时，该属性没有实际意义。
  - weights：None 代表随机初始化，'VOC2007' 代表加载在 VOC2007 上预训练的权值。
  - weights：resnet50 代表使用 resnet50 主干网络训练，'vgg' 代表使用 vgg 主干网络。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- centernet
  - centernet：centernet 目标识别算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 centernet，模型名称保存为 centernet；组件用于预测时，该属性没有实际意义。
  - weights：None 代表随机初始化，'VOC2007' 代表加载在 VOC2007 上预训练的权值。
  - backbone：表示使用的 centernet 的版本。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- maskrcnn

- maskrcnn：maskrcnn 图像实例分割算法。
- 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 maskrcnn，模型名称保存为 maskrcnn；组件用于预测时，该属性没有实际意义。
- weights：None 代表随机初始化，'coco' 代表加载在 coco 上预训练的权值。
- 迭代次数：训练模型的迭代次数，默认为 100。
- 批次大小：一次训练所选取的样本数。
- learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

● Unet

- Unet：Unet 图像语义分割算法。
- 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 Unet，模型名称保存为 Unet；组件用于预测时，该属性没有实际意义。
- backbone：unet 图像分割模型使用的 backbone。默认情况下，使用 'vgg16'。
- weights：None 代表随机初始化，'CamVid' 代表加载在 CamVid 上预训练的权值。
- CLASSES：需要训练的类别名称，原始模型使用的是 CamVid 数据集总类别数为 33，分别为 ['Animal','Archway','Bicyclist','Bridge','Building','Car','CartLuggagePram','Child','Column_Pole','Fence','LaneMkgsDriv','LaneMkgsNonDriv','Misc_Text','MotorcycleScooter','OtherMoving','ParkingBlock','Pedestrian','Road','RoadShoulder','Sidewalk','SignSymbol','Sky','SUVPickupTruck','TrafficCone','TrafficLight','Train','Tree','Truck_Bus','Tunnel','VegetationMisc','Void','Wall','unlabelled']，该属性所输入名称应该为该类别的子集 ，默认['Car','Pedestrian']。

- 迭代次数：训练模型的迭代次数，默认为 100。
- 批次大小：一次训练所选取的样本数。
- learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- FPN
  - FPN：FPN 图像语义分割算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 FPN，模型名称保存为 FPN；组件用于预测时，该属性没有实际意义。
  - backbone：FPN 图像分割模型使用的 backbone。默认情况下，使用'vgg16'。
  - weights：None 代表随机初始化，'CamVid' 代表加载在 CamVid 上预训练的权值。
  - CLASSES：需要训练的类别名称，原始模型使用的是 CamVid 数据集总类别数为 33，分别为 ['Animal','Archway','Bicyclist','Bridge','Building','Car','CartLuggagePram','Child','Column_Pole','Fence','LaneMkgsDriv','LaneMkgsNonDriv','Misc_Text','MotorcycleScooter','OtherMoving','ParkingBlock','Pedestrian','Road','RoadShoulder','Sidewalk','SignSymbol','Sky','SUVPickupTruck','TrafficCone','TrafficLight','Train','Tree','Truck_Bus','Tunnel','VegetationMisc','Void','Wall','unlabelled']，该属性所输入名称应该为该类别的子集 ，默认['Car','Pedestrian']。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- PSPNet
  - PSPNet：PSPNet 图像语义分割算法。

➢ 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 PSPNet，模型名称保存为 PSPNet；组件用于预测时，该属性没有实际意义。

➢ backbone：PSPNet 图像分割模型使用的 backbone。默认情况下，使用 'vgg16'。

➢ weights：None 代表随机初始化，'CamVid' 代表加载在 CamVid 上预训练的权值。

➢ CLASSES：需要训练的类别名称，原始模型使用的是 CamVid 数据集总类别数为 33，分别为 ['Animal','Archway','Bicyclist','Bridge','Building','Car','CartLuggagePram','Child','Column_Pole','Fence','LaneMkgsDriv','LaneMkgsNonDriv','Misc_Text','MotorcycleScooter','OtherMoving','ParkingBlock','Pedestrian','Road','RoadShoulder','Sidewalk','SignSymbol','Sky','SUVPickupTruck','TrafficCone','TrafficLight','Train','Tree','Truck_Bus','Tunnel','VegetationMisc','Void','Wall','unlabelled']，该属性所输入名称应该为该类别的子集 ，默认['Car','Pedestrian']。

➢ 迭代次数：训练模型的迭代次数，默认为 100。

➢ 批次大小：一次训练所选取的样本数。

➢ learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

● Linknet

➢ Linknet：Linknet 图像语义分割算法。

➢ 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 Linknet，模型名称保存为 Linknet；组件用于预测时，该属性没有实际意义。

➢ backbone：Linknet 图像分割模型使用的 backbone。默认情况下，使用 'vgg16'。

➢ weights：None 代表随机初始化，'CamVid' 代表加载在 CamVid 上

预训练的权值。

- CLASSES：需要训练的类别名称，原始模型使用的是 CamVid 数据集总类别数为 33，分别为 ['Animal','Archway','Bicyclist','Bridge','Building','Car','CartLuggagePram','Child','Column_Pole','Fence','LaneMkgsDriv','LaneMkgsNonDriv','Misc_Text','MotorcycleScooter','OtherMoving','ParkingBlock','Pedestrian','Road','RoadShoulder','Sidewalk','SignSymbol','Sky','SUVPickupTruck','TrafficCone','TrafficLight','Train','Tree','Truck_Bus','Tunnel','VegetationMisc','Void','Wall','unlabelled']，该属性所输入名称应该为该类别的子集 ，默认['Car','Pedestrian']。
- 迭代次数：训练模型的迭代次数，默认为 100。
- 批次大小：一次训练所选取的样本数。
- learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- FaceNet
  - FaceNet：FaceNet 人脸识别算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 Linknet，模型名称保存为 Linknet；组件用于预测时，该属性没有实际意义。
  - backbone：FaceNet 人脸识别模型使用的 backbone。默认情况下，使用 'mobilenet'。
  - weights：None 代表随机初始化，'CASIA-WebFace' 代表加载在 CASIA-WebFace 上预训练的权值。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数，需要是 3 的倍数，默认 24。
  - 优化器：设置模型优化器。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

- humanpose
  - humanpose：人体关键点检测算法。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 humanpose，模型名称保存为 humanpose；组件用于预测时，该属性没有实际意义。
  - weights：None 代表随机初始化，'coco' 代表加载在 coco 上预训练的权值。
  - 迭代次数：训练模型的迭代次数，默认为 100。
  - 批次大小：一次训练所选取的样本数。
  - learningrate：学习率，0 到 1 之间的浮点数，默认 0.001。

3.3.3 仿真回归算法

- 基于网格和工况预测仿真云图结果的模型
  - meshgraphnets：基于网格和工况预测仿真云图结果的模型。
  - 模型名称：保存模型的名称；当模型名称为空时，保存模型名称默认为选择的算法名称，如 meshgraphnets，模型名称保存为 meshgraphnets；组件用于预测时，该属性没有实际意义。
  - weights：None 代表不加载预训练权重，'w1' 代表加载在 w1 上预训练的权值。
  - 迭代次数：训练模型的迭代次数。
  - 批次大小：一次训练所选取的样本数，过大可能造成内存不足的问题。
  - 潜在向量宽度：潜在向量或者 embedding 的宽度。
  - MLP 层数：多层感知器（MLP）的层数。
  - 信息传递次数：图神经网络节点间信息传递次数。
  - 前向传播步数：模型在训练开始时仅进行前向传播的步数。
  - 最大 adapt 次数：正则化层进行 adapt 的最大次数，建议为 batch 数的整数倍。
  - 初始学习率：多项式学习率衰减算法初始时的学习率，0 到 1 之间的

浮点数。

- 结束学习率：多项式学习率衰减算法结束时的学习率，0 到 1 之间的浮点数。
- 递减次数：多项式学习率衰减算法的递减次数。
- 多项式的幂：多项式学习率衰减算法的幂。
- ε :Adam 优化器的 epsilon，0 到 1 之间的浮点数。

- Sequential（序贯模型）
  - 名称(name)：字符串，作为输入层下的第一层，用于指定模型框架。

- Dense（全连接层）
  - 名称（name）：输入框，字符串，为该 Dense 层指定名称，作为其标识，默认 Dense。
  - 神经元数（units）：输入框，整型数值，定义该层的神经元数量，即输出空间维度，默认 64。
  - 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 None。
  - 输入维度(Input Shape)：元组或整数类型输入框，n 维张量，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需根据明确指定输入维度(batch_size, ..., input_dim)，输入为二维时输入维度为 (batch_size, input_dim)，非第一层时输入 None，默认 None。
  - 阈值(use_bias)：选择框（True 或 False），布尔型，定义是否使用偏置向量，默认 True。
  - 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform。

- 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros。
- 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。
- 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

● Conv1D（一维卷积层）

- 名称(name)：字符串，为该 Conv1D 层指定名称(name)，作为其标识，默认 Conv1D；
- 滤波器数(filters)：输入框，整型数值，定义卷积核的数目，即输出的维度，默认 32；
- 卷积核大小(kernel_size)：输入框，整数或由单个整数构成的列表或元组，定义卷积核的空域或时域窗口的长度，默认 2；
- 步幅(strides)：输入框，整数或由单个整数构成的列表或元组，定义卷积的步长，默认 1；
- 填充(padding)：选择框，定义补 0 策略，包括 valid、same 和 causal，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，causal 表示将产生因果（膨胀的）卷积，即 output[t]不依赖于 input[t+1：]，当对不能违反时间顺序的时序

信号建模时有用，默认 valid；

- 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, steps, channels），channels_first 对应输入形状为（batch, channels, steps），默认 channels_last；
- 膨胀率(dilation_rate)：输入框，整数或由单个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认 1；
- 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 None；
- 阈值(use_bias)：选择框，布尔型，定义是否使用偏置项，默认 True；
- 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform；
- 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros；
- 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None；
- 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None；
- 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None；
- 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None；

➢ 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None；

➢ 输入维度(input_shape)：输入框，整数元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如(10,128)代表一个长为 10 的序列，序列中每个信号为 128 向量，而(None, 128)代表变长的 128 维向量序列，非第一层时输入 None，默认 None。

● Conv2D（二维卷积层）

➢ 名称(name)：输入框，字符串，为该 Conv2D 层指定名称，作为其标识，默认 Conv2D。

➢ 滤波器数(filters)：输入框，整型数值，定义卷积核的数目，即输出的维度，默认 32。

➢ 卷积核大小(kernel_size)：输入框，单个整数或由两个整数构成的列表或元组，定义卷积核宽度和长度，默认（2,2）。

➢ 步幅(strides)：输入框，单个整数或由两个整数构成的列表或元组，定义卷积的步长，默认（1,1）。

➢ 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, height, width, channels），channels_first 对应输入形状为（batch, channels, height, width），默认 channels_last。

➢ 膨胀率(dilation_rate)：输入框，单个整数或由两个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认（1，1）。

➢ 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 None。

➢ 阈值(use_bias)：选择框，布尔型，定义是否使用偏置项，默认 True。

➢ 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform。

➢ 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros。

➢ 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

➢ 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

➢ 输入维度(Input Shape)：输入框，整数元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (128,128,3)代表 128*128 的彩色 RGB 图像（data_format='channels_last'），非第一层时输入 None，默认 None。

● Conv2DTranspose（转置卷积层）

➢ 名称(name)：字符串，为该 Conv2DTranspose 层指定名称(name)，作为其标识，默认 Conv2DTranspose。

➢ 滤波器数(filters)：输入框，整型数值，定义卷积核的数目，即输

出的维度，默认 32。

- 卷积核大小(kernel_size)：输入框，单个整数或由两个整数构成的列表或元组，定义卷积核宽度和长度，默认（2,2）。
- 步幅(strides)：输入框，单个整数或由两个整数构成的列表或元组，定义卷积的步长，默认（1,1）。
- 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。
- 输出填充(output_padding)：输入框，单个整数或由两个整数构成的列表或元组，指定沿输出张量的高度和宽度的填充量，沿给定维度的输出填充量必须小于沿同一维度的步幅，默认 None，表示输出形状根据网络自动推算。
- 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, height, width, channels），channels_first 对应输入形状为（batch, channels, height, width），默认 channels_last。
- 膨胀率(dilation_rate)：输入框，单个整数或由两个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认（1，1）。
- 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 None。
- 阈值(use_bias)：选择框，布尔型，定义是否使用偏置项，默认 True。
- 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform。
- 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros。

➢ 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

➢ 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

➢ 输入维度(Input Shape)：输入框，整数元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (128,128,3)代表 128*128 的彩色 RGB 图像（data_format='channels_last'），非第一层时输入 None，默认 None。

● Conv3D（三维卷积层）

➢ 名称(name)：字符串，为该 Conv3D 层指定名称，作为其标识，默认 Conv3D。

➢ 滤波器数(filters)：输入框，整型数值，定义卷积核的数目，即输出的维度，默认 32。

➢ 卷积核大小(kernel_size)：输入框，单个整数或由 3 个整数构成的列表或元组，定义卷积核深度、高度和宽度，默认（2,2,2）。

➢ 步幅(strides)：输入框，单个整数或由 3 个整数构成的列表或元组，定义卷积沿每个方向的步长，默认（1,1,1）。

➢ 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积

结果，使输出形状和输入形状相同，默认 valid。

- 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。
- 膨胀率(dilation_rate)：输入框，单个整数或由 3 个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认（1,1,1）。
- 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 None。
- 阈值(use_bias)：选择框，布尔型，定义是否使用偏置项，默认 True。
- 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform。
- 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros。
- 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，

选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

- 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。
- 输入维度(Input Shape)：输入框，整数元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (10,128,128,3)代表对 10 帧 128*128 的彩色 RGB 图像进行卷积（data_format='channels_last'），非第一层时输入 None，默认 None。

- Conv3D（三维卷积层）
  - 名称(name)：字符串，为该 Conv3D 层指定名称，作为其标识，默认 Conv3D。
  - 滤波器数(filters)：输入框，整型数值，定义卷积核的数目，即输出的维度，默认 32。
  - 卷积核大小(kernel_size)：输入框，单个整数或由 3 个整数构成的列表或元组，定义卷积核深度、高度和宽度，默认（2,2,2）。
  - 步幅(strides)：输入框，单个整数或由 3 个整数构成的列表或元组，定义卷积沿每个方向的步长，默认（1,1,1）。
  - 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。
  - 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。
  - 膨胀率(dilation_rate)：输入框，单个整数或由 3 个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认（1,1,1）。
  - 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、

exponential、None，选 None 表示不指定激活函数，默认选择 None。

- 阈值(use_bias)：选择框，布尔型，定义是否使用偏置项，默认 True。
- 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform。
- 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros。
- 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。
- 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。
- 输入维度(Input Shape)：输入框，整数元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (10,128,128,3)代表对 10 帧 128*128 的彩色 RGB 图像进行卷积（data_format='channels_last'），非第一层时输入 None，默认 None。

- Conv3DTranspose（转置卷积层）
  - 名称(name)：字符串，为该 Conv3DTranspose 层指定名称，作为其

标识，默认 Conv3DTranspose。

- 滤波器数(filters)：输入框，整型数值，定义卷积核的数目，即输出的维度，默认 32。
- 卷积核大小(kernel_size)：输入框，单个整数或由 3 个整数构成的列表或元组，定义卷积核深度、高度和宽度，默认（2,2,2）。
- 步幅(strides)：输入框，单个整数或由 3 个整数构成的列表或元组，定义卷积沿深度、高度和宽度的步长，默认（1,1,1）。
- 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。
- 输出填充(output_padding)：输入框，单个整数或由 3 个整数构成的列表或元组，指定沿深度、高度和宽度的填充量，沿给定维度的输出填充量必须小于沿同一维度的步幅，默认 None，表示输出形状根据网络自动推算。
- 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。
- 膨胀率(dilation_rate)：输入框，单个整数或由 3 个整数构成的列表或元组，定义膨胀卷积的膨胀率，默认（1,1,1）。
- 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 None。
- 阈值(use_bias)：选择框，布尔型，定义是否使用偏置项，默认 True。
- 内核初始化（kernel_initializer）：选择框，定义内核权值矩阵的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 glorot_uniform。

- 阈值初始化（bias_initializer）：选择框，定义偏置向量的初始化方法，选项内容同内核初始化，默认 zeros。
- 内核正则化(kernel_regularizer)：输入框，字符串，定义内核权值矩阵的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 阈值正则化（bias_regularizer）：输入框，字符串，定义偏置向量的正则化函数,如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 活动正则化(activity_regularizer)：输入框，字符串，定义层的输出的正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。
- 内核约束(kernel_constraint)：选择框，定义权值矩阵的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。
- 阈值约束(bias_constraint)：选择框，定义偏置向量的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。
- 输入维度(Input Shape)：输入框，整数元组，定义输入数据的维度，该层作为 Sequential 层下的第一层时，需指定输入维度，例如 input_shape = (10,128,128,3)代表对 10 帧 128*128 的彩色 RGB 图像进行卷积（data_format='channels_last'），非第一层时输入 None，默认 None。

● Cropping1D（一维裁剪层）

- 名称(name)：输入框，字符串，为该 Cropping1D 层指定名称，作为其标识，默认 Cropping1D；
- 裁剪大小：输入框，整数或长为 2 的整数元组，指定在序列的首尾要裁剪掉多少个元素，默认（1,1）。

● Cropping2D（二维裁剪层）

- 名称(name)：输入框，字符串，为该 Cropping2D 层指定名称，作为其标识，默认 Cropping2D。

➢ 裁剪大小(cropping)：输入框，整数或 2 个整数的元组或 2 个整数的 2 个元组的元组，如果为整数（如 1），表示相同的对称裁剪应用于高度和宽度；如果为 2 个整数的元组（如（1,1）），表示对高度和宽度的两个不同的对称裁剪值：(symmetric_height_crop, symmetric_width_crop)；如果为 2 个整数的 2 个元组的元组：解释为 ((top_crop, bottom_crop), (left_crop, right_crop))，默认((0,0),(0,0))。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, height, weight, channels），channels_first 对应输入形状为（batch, channels, height, weight），默认 channels_last。

- Cropping3D（三维裁剪层）

➢ 名称(name)：输入框，字符串，为该 Cropping3D 层指定名称，作为其标识，默认 Cropping3D。

➢ 裁剪大小(cropping)：输入框，整数或 3 个整数的元组或 2 个整数的 3 个元组的元组，如果为整数，将对深度、宽度和高度应用相同的对称裁剪；如果为 3 个整数的元组：解释为对深度、高度和宽度的 3 个不同的对称裁剪值：(symmetric_dim1_crop, symmetric_dim2_crop, symmetric_dim3_crop)；如果为 2 个整数的 3 个元组的元组：解释为 ((left_dim1_crop, right_dim1_crop), (left_dim2_crop, right_dim2_crop), (left_dim3_crop, right_dim3_crop))，默认 ((1,1),(1,1),(1,1))。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。

- Cropping3D（三维裁剪层）

➢ 名称(name)：输入框，字符串，为该 Cropping3D 层指定名称，作为其标识，默认 Cropping3D。

➢ 裁剪大小(cropping)：输入框，整数或 3 个整数的元组或 2 个整数的 3 个元组的元组，如果为整数，将对深度、宽度和高度应用相同的对称裁剪；如果为 3 个整数的元组：解释为对深度、高度和宽度的 3 个不同的对称裁剪值：(symmetric_dim1_crop, symmetric_dim2_crop, symmetric_dim3_crop)；如果为 2 个整数的 3 个元组的元组：解释为((left_dim1_crop, right_dim1_crop), (left_dim2_crop, right_dim2_crop), (left_dim3_crop, right_dim3_crop))，默认((1,1),(1,1),(1,1))。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。

● UpSampling1D（一维上采样层）

➢ 名称(name)：字符串，为该 UpSampling1D 层指定名称，作为其标识，默认 UpSampling1D。

➢ 上采样大小(size)：整数，定义上采样因子，指沿着时间轴重复每个时间步的次数，默认 2。

● UpSampling2D（二维上采样层）

➢ 名称(name)：字符串，为该 UpSampling2D 层指定名称，作为其标识，默认 UpSampling2D。

➢ 上采样大小(size)：整数或 2 个整数的元组，定义行和列的上采样因子，指沿着数据的行和列分别重复的次数，默认（2,2）。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为

（batch, height, weight, channels），channels_first 对应输入形状为（batch, channels, height, weight），默认 channels_last。

- 插值：选择框，nearest 或 bilinear，指定插值方法，默认 nearest。

- UpSampling3D（三维上采样层）
  - 名称(name)：字符串，为该 UpSampling3D 层指定名称，作为其标识，默认 UpSampling3D。
  - 上采样大小(size)：整数或 3 个整数的元组，定义 3 个维度的上采样因子，指沿着数据的 3 个维度分别重复的次数，默认(2,2,2)。
  - 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认 channels_last。

- Activation（激活层）
  - 名称(name)：输入框，字符串，为该 Activation 层指定名称，作为其标识，默认 Activation。
  - 激活函数(activation)：选择框，为该层选择激活函数，选项包括 relu、tanh、sigmoid、linear、softmax、softplus、softsign、hard_sigmoid、exponential、None，选 None 表示不指定激活函数，默认选择 relu。

- MaxPooling1D（时序最大池化层）
  - 名称(name)：输入框，字符串，为该 MaxPooling1D 层指定名称，作为其标识，默认 MaxPooling1D。
  - 池化大小(pool_size)：输入框，整数，定义最大池化的窗口大小，默认 2。
  - 步幅(strides)：输入框，整数或 None，表示下采样因子，例如 2 表示将使得输出 shape 为输入的一半，None 表示默认使用 pool_size 的值，

默认 None。

➢ 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, steps, features），channels_first 对应输入形状为（batch, features, steps），默认 channels_last。

● MaxPooling2D（空间最大池化层）

➢ 名称(name)：字符串，为该 MaxPooling2D 层指定名称，作为其标识，默认 MaxPooling2D。

➢ 池化大小(pool_size)：输入框，整数或 2 个整数的元组，表示沿（垂直，水平）方向的下采样因子，如取（2，2）将使图片在两个维度上均变为原长的一半，为整数意为各个维度值相同且为该数字，默认(2,2)。

➢ 步幅(strides)：整数或长为 2 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。

➢ 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, height, weight, channels），channels_first 对应输入形状为（batch, channels, height, weight），默认 channels_last。

● MaxPooling3D（时/空间最大池化层）

➢ 名称(name)：字符串，为该 MaxPooling3D 层指定名称，作为其标识，默认 MaxPooling3D。

➢ 池化大小(pool_size)：输入框，3 个整数的元组，表示在 3 个维度

上的下采样因子,如取(2,2,2)表示将每个维度的3D输入减半,默认(2,2,2)。

- 步幅(strides)：长为3的整数元组或None，定义步长大小，如果为None，表示pool_size的值，默认None。
- 填充(padding)：选择框，定义补0策略，包括valid和same，valid表示只进行有效的卷积，即对边界数据不处理，same表示保留边界处的卷积结果，使输出形状和输入形状相同，默认valid。
- 数据格式(data_format)：选择框，包括channels_last和channels_first，定义输入中维度的顺序，channels_last对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first对应输入形状为（batch, channels, spatial_dim1, spatial_dim2, spatial_dim3），默认channels_last。

- AveragePooling1D（时序平均池化层）
  - 名称(name)：字符串，为该AveragePooling1D层指定名称，作为其标识，默认AveragePooling1D。
  - 池化大小(pool_size)：输入框，整数，定义平均池化的窗口大小，默认2。
  - 步幅(strides)：输入框，整数或None，表示下采样因子，例如2表示将使得输出shape为输入的一半，None表示默认使用pool_size的值，默认None。
  - 填充(padding)：选择框，定义补0策略，包括valid和same，valid表示只进行有效的卷积，即对边界数据不处理，same表示保留边界处的卷积结果，使输出形状和输入形状相同，默认valid。
  - 数据格式(data_format)：选择框，包括channels_last和channels_first，定义输入中维度的顺序，channels_last对应输入形状为（batch, steps, features），channels_first对应输入形状为（batch, features, steps），默认channels_last。

- AveragePooling2D（空间平均池化层）

➢ 名称(name)：字符串，为该 AveragePooling2D 层指定名称，作为其标识，默认 AveragePooling2D。

➢ 池化大小(pool_size)：输入框，整数或 2 个整数的元组，表示沿（垂直，水平）方向的下采样因子，如取（2，2）将两个空间维度的输入减半，为整数意为各个维度值相同且为该数字，默认(2,2)。

➢ 步幅(strides)：整数或长为 2 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。

➢ 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, height, weight, channels），channels_first 对应输入形状为（batch, channels, height, weight），默认 channels_last。

● AveragePooling3D（时空间平均池化层）

➢ 名称(name)：字符串，为该 AveragePooling3D 层指定名称，作为其标识，默认 AveragePooling3D。

➢ 池化大小(pool_size)：输入框，3 个整数的元组，表示在 3 个维度上的下采样因子,如取(2,2,2)表示将每个维度的 3D 输入减半,默认(2,2,2)。

➢ 步幅(strides)：长为 3 的整数元组或 None，定义步长大小，如果为 None，表示 pool_size 的值，默认 None。

➢ 填充(padding)：选择框，定义补 0 策略，包括 valid 和 same，valid 表示只进行有效的卷积，即对边界数据不处理，same 表示保留边界处的卷积结果，使输出形状和输入形状相同，默认 valid。

➢ 数据格式(data_format)：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, spatial_dim1, spatial_dim2, spatial_dim3, channels），channels_first 对应输入形状为（batch, channels, spatial_dim1,

spatial_dim2, spatial_dim3），默认 channels_last。

- BatchNormalization（批量标准化层）

  - 名称(name)：输入框，字符串，为该 BatchNormalization 层指定名称，作为其标识，默认 BatchNormalization。
  - 特征轴(axis)：输入框，整数，指定需要标准化的轴，默认-1。
  - 动量(momentum)：输入框，浮点数，指定移动均值和移动方差的动量，默认 0.99。
  - epsilon：输入框，浮点数，为方差增加一个小的浮点数，避免被除数为零，默认 0.001。
  - 中心(center)：选择框，布尔值，如果为 True，表示把偏移量加到标准化张量上，如果为 False，则忽略，默认 True。
  - 缩放(scale)：选择框，布尔值，如果为 True，则乘以 gamma，如果为 False，则不使用 gamma。
  - beta 初始化(beta_initializer)：选择框，指定 beta 权重的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 zeros。
  - gamma 初始化(gamma_initializer)：选择框，指定 gamma 权重的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 ones。
  - 移动平均初始化(moving_mean_initializer)：选择框，指定移动平均的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 zeros。

➢ 移动方差初始化(moving_variance_initializer)：选择框，指定移动方差的初始化方法，选项包括 initializer、random_normal、Random_uniform、truncated_normal、identity、lecun_uniform、lecun_normal、orthogonal、zeros、glorot_normal、glorot_uniform、he_normal、he_uniform、ones，默认 ones。

➢ beta 正则化(beta_regularizer)：输入框，字符串，为 beta 权重指定正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ Gamma 正则化(gamma_regularizer)：输入框，字符串，为 gamma 权重指定正则化函数，如 l1(0.01)、l2(0.01)、l1_l2(l1=0.01, l2=0.01)或 None，默认 None。

➢ beta 约束(beta_constraint)：选择框，指定 beta 权值的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

➢ Gamma 约束(Gamma_constraint)：选择框，指定 Gamma 权值的约束函数，选项内容包括 MaxNorm、NonNeg、UnitNorm、MinMaxMorm 或 None，默认 None。

● Dropout（丢失层）

➢ 名称（name）：输入框，字符串，为该 Dropout 层指定名称，作为其标识。

➢ 比例因子(rate)：输入框，0 到 1 之间的浮点数，定义输入数据需要丢弃的比例，默认 0.0。

● Flatten（扁平化层）

➢ 名称（name）：输入框，字符串，为该 Activation 层指定名称，作为其标识。

➢ 数据格式（data_format）：选择框，包括 channels_last 和 channels_first，定义输入中维度的顺序，channels_last 对应输入形状为（batch, height, weight, ..., channels），channels_first 对应输入形

状为（batch, channels, height, weight, ...），默认 channels_last。

- Reshape（重塑层）
  - 名称（name）：字符串，为该 Reshape 层指定名称，作为其标识。
  - 目标形状(target_shape)：整型元组，将数据转换为特定的形状，默认(None, None)。

- RepeatVector（重复层）
  - 名称（name）：字符串，为该 RepeatVector 层指定名称，作为其标识。
  - 重复次数(n)：整型数值，定义输入的重复系数，默认 0。

3.4 模型预测组件

- 数值模型预测
  - 数值模型预测：机器学习模型预测，包括分类算法，回归算法和聚类算法。
  - 待推理模型类型：算法框架，默认使用 'classification'。
  - Confidence：输入 0 到 1 之间的整数或者浮点数，定义输入置信度，默认 0.5。

- 图像模型预测
  - 图像模型预测：图像模型预测。
  - 待推理模型类型：图像检测使用的算法框架，默认使用 'ssd'。
  - Confidence：输入 0 到 1 之间的整数或者浮点数，定义输入置信度，默认 0.5。
  - Confidence：浮点数，定义输入两张图片的差异，数值越大差异越大，数值越小差异越小，0.0 的距离表示面是相同的，4.0 对应的是相反的，两个不同的身份，默认 1.1。
  - nms_iou：输入 0 到 1 之间的整数或者浮点数，定义非极大抑制所用

到的 nms_iou 大小，默认 0.3。

- backbone:unet 图像分割模型使用的 backbone。默认情况下，使用'vgg16'。
- backbone:FaceNet 人脸识别模型使用的 backbone。默认情况下，使用 'mobilenet'。
- Activation：选择激活函数进行非线性转换，如'softmax'，默认使用'softmax'。

- 仿真结果预测
  - 仿真结果预测：仿真结果预测。
  - 待推理模型类型：仿真回归使用的算法框架，默认使用'meshgraphnets'。

- 自定义预测
  - 自定义预测：自定义模型预测。
  - 选择数据文件：选择需要的依赖的文件。
  - 模型预测函数名称：填写模型预测函数名称。
  - 自定义读取数据脚本：按照模板添加自定义读取数据脚本。

4. 附录二：强化学习组件说明

4.1 数据处理组件

4.1.1 仿真环境

- 仿真环境选择：
  - 名称：文本框，更改仿真环境名称，默认值为仿真环境。
  - 选择仿真环境：单选下拉框，选择仿真管理界面成功添加的仿真环境，必填项。
- Gym：
  - 名称：文本框，更改 Gym 算子名称，默认值为 Gym。
  - Scenario：单选下拉框，选择训练想定，默认值为 Pendulum-v1。

SC2：

- 名称：文本框，更改 SC2 算子名称，默认值为 SC2。
- map_name：星际争霸 mingame 地图名称，可以选择 3m、8m、5m_vs_6m、8m_vs_9m、MMM、2s3z 和 3s_vs_3z。
- 生成回放的频率：设置生成回放的频率。

- 自定义开发模板：
  - 名称：文本框，更改自定义开发模板算子名称，默认值为自定义开发模板。
  - 选择开发模板目录：目录选择框，可通过远端进行文件目录选择，必填项无默认值。
  - 环境文件名：文本框，填写文件目录中的环境文件名，必填项无默认值。
  - 环境类名：文本框，填写环境文件中的环境类名，用于创建 python 仿真实例，必填项无默认值。

4.1.2 离线数据集

- 离线数据集选择
  - 名称：文本框，更改组件的名称，默认值为离线数据集选择。
- Gym：
  - 名称：文本框，更改 Gym 算子名称，默认值为 Gym。

➢ Scenario：单选下拉框，选择训练想定，默认值为 Pendulum-v1。

SC2：

➢ 名称：文本框，更改 SC2 算子名称，默认值为 SC2。

➢ map_name：星际争霸 mingame 地图名称，可以选择 3m、8m、5m_vs_6m、8m_vs_9m、MMM、2s3z 和 3s_vs_3z。

➢ 生成回放的频率：设置生成回放的频率。

- 自定义开发模板：

➢ 名称：文本框，更改自定义开发模板算子名称，默认值为自定义开发模板。

➢ 选择开发模板目录：目录选择框，可通过远端进行文件目录选择，必填项无默认值。

➢ 环境文件名：文本框，填写文件目录中的环境文件名，必填项无默认值。

➢ 环境类名：文本框，填写环境文件中的环境类名，用于创建 python 仿真实例，必填项无默认值。

4.2 算法设计

4.2.1 自定义模型

➢ 名称：文本框，更改组件名称，默认值为自定义模型。

➢ 自定义模型脚本：按照模板添加自定义模型脚本，目前自定义模型只适用于在线强化学习算法，离线强化学习算法不支持。

4.2.2 模型训练

➢ 名称：文本框，更改模型训练算子名称，默认值为模型训练。

➢ framework：单选下拉框，对训练框架 pytorch 和 tensorflow 进行选择，默认值为 pytorch。

➢ 训练终止平均奖励：文本框，类型为数值型，当达到平均奖励时训练终止，默认值 50。

➢ 训练终止总步长：文本框，类型为数值型，当达到总步长时训练终止，默认值为 1000。

➢ 训练终止总局数：文本框，类型为数值型，当达到总局数时训练终止，默认值为 10。

➢ 并行训练数量：文本框，类型为数值型，设置并行训练数量，默认值为 1。

4.2.3 算法库

4.2.3.1 内置算法

4.2.3.1.1 在线强化学习算法

####### 4.2.3.1.1.1 单智能体算法

- A2C：
  - ➢ Train_bath_size：正整数，训练单个批量大小，如 3000；
  - ➢ Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - ➢ Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - ➢ Microbatch_size：正整数，支持小批量训练，当 train_batch_size 为该值的 3 倍以上时，小批量训练才有意义，默认 10；
- A3C：
  - ➢ Train_bath_size：正整数，训练单个批量大小，如 3000；
  - ➢ Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - ➢ Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - ➢ Use_critic：布尔值，表示是否使用 critic 作为基线计算收益，默认 True；
  - ➢ Gae：布尔值，表示是否使用广义优势估计计算收益，默认 True；
  - ➢ Lambda_：0 到 1 之间的浮点数，表示 GAE（lambda）参数，默认 1；
  - ➢ Grad_clip：正整数，梯度最大值，默认 40；
  - ➢ Vf_loss_coeff：0 到 1 之间的浮点数，价值函数损失系数，默认 0.5；
  - ➢ Entropy_coeff：0 到 1 之间的浮点数，熵正则表达式的系数，默认 0.001；

- Sample_async：布尔值，表示是否异步采样，默认 False；

- APEX_DDPG：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - No_local_replay_buffer：布尔值，是否本地存储经验数据，默认 True；
  - Capacity：正整数，replaybuffer 容量，默认 10000；
  - Prioritized_replay_alpha：0 到 1 之间的浮点数，alpha 参数，默认 0.6；
  - Prioritized_replay_beta：0 到 1 之间的浮点数，beta 参数，默认 0.4；
  - Prioritized_replay_eps：正整数，epsilon 在更新优先级时添加到 TD 误差中，默认 50000；

- APEX_DQN：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Num_atoms：正整数，表示回报分布的原子数，默认 1；
  - V_min：整数，最小值估计，默认-10；
  - V_max：整数，最大值估计，默认 10；
  - Noisy：布尔值，是否使用干扰网络来帮助探索，默认 False；
  - Sigma0：0 到 1 之间的浮点数，控制噪声网络的初始参数噪声，默认 0.5；
  - Dueling：布尔值，是否使用 Dueling 网络，默认 True；
  - Double_q：布尔值，是否使用双 DQN，默认 True；
  - N_step：正整数，Q 学习的 N 步，默认 1；
  - Td_error_loss_fn：选择框，TD 误差的损失函数，选项内容包括 Huber、MSE，默认 Huber；

- Categorical_distribution_temperature：0 到 1 之间的浮点数，分类分布区间，默认 1.0；
- No_local_replay_buffer：布尔值，是否本地存储经验数据，默认 True；
- Capacity：正整数，replaybuffer 容量，默认 10000；
- Prioritized_replay_alpha：0 到 1 之间的浮点数，alpha 参数，默认 0.6；
- Prioritized_replay_beta：0 到 1 之间的浮点数，beta 参数，默认 0.4；
- Prioritized_replay_eps：正整数，epsilon 在更新优先级时添加到 TD 误差中，默认 50000；

● APPO：

- Train_bath_size：正整数，训练单个批量大小，如 3000；
- Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
- Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
- Vtrace：布尔值，是否使用 V-trace 计算优势，默认 True；
- Use_critic：布尔值，是否使用评论家作为基线，默认 True；
- Use_gae：布尔值，是否使用广义优势估算器 （GAE），默认 True；
- Lambda_：0 到 1 之间的浮点数，GAE 中 lambda 参数，默认 1.0；
- Clip_param：0 到 1 之间的浮点数，截断参数，默认 0.4；
- Use_kl_loss：布尔值，是否在损失函数中使用 KL 项，默认 False；
- Kl_coeff：0 到 1 之间的浮点数，用于加权 KL 损失项的系数，默认 1.0；
- Kl_target：0 到 1 之间的浮点数，KL 要达到的目标期限，默认 0.01；

● DDPG：

- Train_bath_size：正整数，训练单个批量大小，如 3000；
- Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
- Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
- Twin_q：布尔值，是否使用双 Q 网络，默认 False；

- Policy_delay：正整数，策略更新延长局数，默认 1；
- Smooth_target_policy：布尔值，目标策略平滑，默认 False；
- Target_noise：0 到 1 之间的浮点数，用于平滑的目标动作噪声的高斯标准偏差，默认 0.2；
- Target_noise_clip：0 到 1 之间的浮点数，目标噪声限制（边界），默认 0.5；
- Use_state_preprocessor：布尔值，应用状态预处理器，默认 False；
- N_step：正整数，N 步 Q 学习，默认 1；
- Critic_lr：0 到 1 之间的浮点数，评论家（Q 函数）优化器的学习率，默认 0.001；
- Actor_lr：0 到 1 之间的浮点数，演员（策略）优化器的学习率，默认 0.001；
- Tau：0 到 1 之间的浮点数， 按 tau * 策略 + （1- tau） * target_policy 更新目标，默认 0.002；
- Use_huber：布尔值，是否使用 Huber 损失函数，默认 False；
- Huber_threshold：大于 0 的浮点数，Huber 损失的阈值，默认 1.0；
- L2_reg：0 到 1 之间的浮点数，L2 正则化的权重，默认 0.000001；

- DQN：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Num_atoms：正整数，表示回报分布的原子数，默认 1；
  - V_min：整数，最小值估计，默认-10；
  - V_max：整数，最大值估计，默认 10；
  - Noisy：布尔值，是否使用干扰网络来帮助探索，默认 False；
  - Sigma0：0 到 1 之间的浮点数，控制噪声网络的初始参数噪声，默认 0.5；
  - Dueling：布尔值，是否使用 Dueling 网络，默认 True；

- Double_q：布尔值，是否使用双 DQN，默认 True；
- N_step：正整数，Q 学习的 N 步，默认 1；
- Td_error_loss_fn：选择框，TD 误差的损失函数，选项内容包括 Huber、MSE，默认 Huber；
- Categorical_distribution_temperature：0 到 1 之间的浮点数，分类分布区间，默认 1.0；
- No_local_replay_buffer：布尔值，是否本地存储经验数据，默认 True；
- Capacity：正整数，replaybuffer 容量，默认 10000；
- Prioritized_replay_alpha：0 到 1 之间的浮点数，alpha 参数，默认 0.6；
- Prioritized_replay_beta：0 到 1 之间的浮点数，beta 参数，默认 0.4；
- Prioritized_replay_eps：正整数，epsilon 在更新优先级时添加到 TD 误差中，默认 50000；

● IMPALA：

- Train_bath_size：正整数，训练单个批量大小，如 3000；
- Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
- Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
- Minibatch_buffer_size：正整数，小批量时保存经验数据大小，默认 1；
- Num_sgd_iter：正整数，sgd 迭代次数，默认 1；
- Replay_proportion：正整数，设置大于 0 以启用体验重播，保存的样本将以与新数据样本的 p：1 比例重播，默认 0；
- Replay_buffer_num_slots：正整数，要存储以供重播的样本批次数，默认 0；
- Learner_queue_size：正整数，学习器大小，默认 16；
- Learner_queue_timeout：正整数，等待训练时间，默认 300；
- Grad_clip：正整数，梯度的全局范数裁剪为此量，默认 40；
- Opt_type：选择框，选择优化器，选择内容为 adam、rmsprop：默认 adam；

- Decay：0 到 1 之间的浮点数，优化器衰减系数，默认 0.99；
- Momentum：0 到 1 之间的浮点数，动量，默认 0.0；
- Epsilon：0 到 1 之间的浮点数，优化器 Epsilon，默认 0.1；
- Vf_loss_coeff：0 到 1 之间的浮点数，损失函数中价值函数项的系数，默认 0.5；
- Entropy_coeff：0 到 1 之间的浮点数，损失中熵正则表达式项的系数功能，默认 0.01；
- _separate_vf_optimizer：布尔值，是否拥有两个单独的优化器，默认 False；
- _lr_vf：0 到 1 之间的浮点数，价值网络学习率，默认 0.0005；

● PG：

- Train_bath_size：正整数，训练单个批量大小，如 3000；
- Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
- Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；

● PPO：

- Train_bath_size：正整数，训练单个批量大小，如 3000；
- Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
- Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
- Use_critic：布尔值，是否使用评论家作为基线，默认 True；
- Use_gae：布尔值，是否使用广义优势估算器 （GAE），默认 True；
- Lambda_：0 到 1 之间的浮点数，GAE 中 lambda 参数，默认 1.0；
- Kl_coeff：0 到 1 之间的浮点数，kl 散度的初始系数，默认 0.2；
- Sgd_minibatch_size：正整数，用于 sgd 的小批处理大小，默认 128；
- Num_sgd_iter：正整数，sgd 迭代次数，默认 30；
- Shuffle_sequences：布尔值，训练时是否在批次中打乱序列，默认 True；
- Vf_loss_coeff：0 到 1 之间的浮点数，值函数损失系数，默认 1；

- Entropy_coeff：0 到 1 之间的浮点数，熵正则化子的系数，默认 0；
- Clip_param：0 到 1 之间的浮点数，截断参数，默认 0.4；
- Vf_clip_param：正整数，值函数的裁剪参数，默认 10.0；
- Kl_target：0 到 1 之间的浮点数，KL 要达到的目标期限，默认 0.01；

- R2D2：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Num_atoms：正整数，表示回报分布的原子数，默认 1；
  - V_min：整数，最小值估计，默认-10；
  - V_max：整数，最大值估计，默认 10；
  - Noisy：布尔值，是否使用干扰网络来帮助探索，默认 False；
  - Sigma0：0 到 1 之间的浮点数，控制噪声网络的初始参数噪声，默认 0.5；
  - Dueling：布尔值，是否使用 Dueling 网络，默认 True；
  - Double_q：布尔值，是否使用双 DQN，默认 True；
  - N_step：正整数，Q 学习的 N 步，默认 1；
  - Td_error_loss_fn：选择框，TD 误差的损失函数，选项内容包括 Huber、MSE，默认 Huber；
  - Categorical_distribution_temperature：0 到 1 之间的浮点数，分类分布区间，默认 1.0；
  - No_local_replay_buffer：布尔值，是否本地存储经验数据，默认 True；
  - Capacity：正整数，replaybuffer 容量，默认 10000；
  - Prioritized_replay_alpha：0 到 1 之间的浮点数，alpha 参数，默认 0.6；
  - Prioritized_replay_beta：0 到 1 之间的浮点数，beta 参数，默认 0.4；
  - Prioritized_replay_eps：正整数，epsilon 在更新优先级时添加到 TD 误差中，默认 50000；

- SAC：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Twin_q：布尔值，是否使用两个 Q 网络进行操作值估计，默认 True；
  - Clip_actions：布尔值，是否截断动作，默认 False；
  - Initial_alpha：0 到 1 之间的浮点数，用于熵权重 alpha 的初始值，默认 1.0；
  - N_step：正整数，Q 学习的 N 步，默认 1；
  - Store_buffer_in_checkpoints：布尔值，经验缓存存储到模型路径下，默认 False；
  - Target_network_update_freq：正整数，更新目标网络频率，默认 0；
- TD3：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Twin_q：布尔值，是否使用双 Q 网络，默认 False；
  - Policy_delay：正整数，策略更新延长局数，默认 1；
  - Smooth_target_policy：布尔值，目标策略平滑，默认 False；
  - Target_noise：0 到 1 之间的浮点数，用于平滑的目标动作噪声的高斯标准偏差，默认 0.2；
  - Target_noise_clip：0 到 1 之间的浮点数，目标噪声限制（边界），默认 0.5；
  - Use_state_preprocessor：布尔值，应用状态预处理器，默认 False；
  - N_step：正整数，N 步 Q 学习，默认 1；
  - Critic_lr：0 到 1 之间的浮点数，评论家（Q 函数）优化器的学习率，默认 0.001；

- Actor_lr：0 到 1 之间的浮点数，演员（策略）优化器的学习率，默认 0.001；
- Tau：0 到 1 之间的浮点数， 按 tau * 策略 + （1- tau） * target_policy 更新目标，默认 0.002；
- Use_huber：布尔值，是否使用 Huber 损失函数，默认 False；
- Huber_threshold：大于 0 的浮点数，Huber 损失的阈值，默认 1.0；
- L2_reg：0 到 1 之间的浮点数，L2 正则化的权重，默认 0.000001；

4.2.3.1.1.2 多智能体算法

- MADDPG：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Use_local_critic：布尔值，是否使用本地评论家，默认 False；
  - Use_state_preprocessor：布尔值，是否使用全局状态处理器，默认 False；
  - N_step：正整数，Q 学习的 N 步，默认 1；
  - Num_steps_sampled_before_learning_starts：正整数，当收集到一定时间步数时开始学习，默认 3000；
  - Critic_lr：0 到 1 之间的浮点数，评论家（Q 函数）优化器的学习率，默认 0.001；
  - Actor_lr：0 到 1 之间的浮点数，演员（策略）优化器的学习率，默认 0.001；
  - Target_network_update_freq：正整数，更新网络频率，默认 0；
  - Tau：0 到 1 之间的浮点数， 按 tau * 策略 + （1- tau） * target_policy 更新目标，默认 0.002；
  - Actor_feature_reg：0 到 1 之间的浮点数，演员特征正则化的权重，默认 0.001；
  - Grad_norm_clipping：0 到 1 之间的浮点数，优化截断系数，默认 0.5；

- QMIX：
  - Train_bath_size：正整数，训练单个批量大小，如 3000；
  - Lr：0 到 1 之间的浮点数，学习率，默认 0.00005；
  - Gamma：0 到 1 之间的浮点数，表示马尔可夫决策过程折现因子，默认 0.99；
  - Mixer：选择框，混合网络，选择内容 qmix、vdn，默认 qmix；
  - Mixing_embed_dim：正整数，混合网络嵌入的大小，默认 32；
  - Double_q：布尔值，是否使用 Double_Q 学习，默认 True；
  - Target_network_update_freq：正整数，更新目标网络频率，默认 500；
  - Num_steps_sampled_before_learning_starts：正整数，开始学习步数，默认 1000；
  - Optim_alpha：0 到 1 之间的浮点数，RMSProp 的 alpha 参数，默认 0.99；
  - Optim_eps：0 到 1 之间的浮点数，RMSProp 的 epsilon 参数，默认 0.00001；
  - Grad_clip：大于 0 浮点数，梯度裁剪，默认 10.0；
  - Simple_optimizer：布尔值，简易优化器，默认 True；

4.2.3.1.2 离线强化学习算法

- BC：
  - Beta：大于 0 浮点数，表示以指数形式衡量优势,当 beta 为 0.0 时，MARWIL 被简化为模仿学习，默认值为 0。
  - Bc_logstd_coeff：大于 0 浮点数,表示鼓励更高行动分布熵的系数用于探索，默认值为 0。
  - moving_average_sqd_adv_norm_start：0 到 1 之间的浮点数，表示移动平均优势范数的起始值，默认值 0.0001。
  - use_gae：是否将广义优势估计器（GAE）与值函数一起使用，默认值为 True。
  - vf_coeff：大于 0 浮点数，平衡价值估计损失和策略优化损失，默认值 1.0。

- grad_clip：大于 0 浮点数，剪裁渐变的全局范数，默认值为 0。

- CRR：
  - Temperature：exp-weight 类型中使用的指数温度，默认值 1.0。
  - max_weight：exp-weight 类型的最大重量限制，默认值 20.0。
  - n_action_sample：动作采样个数，默认值 4。
  - target_network_update_freq：目标网络更新频率,默认值 100。
  - actor_hiddens：采样网络的隐藏单元个数，默认值[256,256]。
  - actor_hidden_activation：采样网络的隐藏层激活函数，默认值 relu。
  - critic_hiddens：评估网络的隐藏单元格式，默认值[256,256]。
  - critic_hidden_activation：评估网络的隐藏层激活函数，默认值 relu。
  - Tau：Polyak 平均系数，默认值 0.0005。
  - categorical_distribution_temperature：设置类别动作分布所使用的温度参数，有效值在 0 到 1 之间,默认值 1.0。
  - Weight_type：使用 bin | exp 的权重类型，默认值 bin。
  - advantage_type：q 值减少为 v_t 值 max|mean|expection 的方法，默认值 mean。
  - twin_q：是否使用悲观的 q 值估计，默认值 True。
  - td_error_loss_fn：计算 td_error 的损失函数，选择 huber 或者 mse，默认值 mse。

- CQL：
  - bc_iters：行为克隆预训练的迭代次数，默认值 2000。
  - Temperature：CQL 损耗温度，默认值 1.0。
  - num_actions：CQL 丢失的采样操作数，默认值 10。
  - lagrangian_thresh：拉格朗日阈值,默认值 5.0。
  - min_q_weight：最小 Q 权重，默认值 5.0。
  - Lagrangian：是否对 Alpha Prime 使用拉格朗日量，默认值 False。

- MARWIL2：
  - Beta：大于 0 浮点数，表示以指数形式衡量优势,当 beta 为 0.0 时，MARWIL 被简化为模仿学习，默认值为 1。

- Bc_logstd_coeff：大于 0 浮点数,表示鼓励更高行动分布熵的系数用于探索，默认值为 0。
- moving_average_sqd_adv_norm_start：0 到 1 之间的浮点数，表示移动平均优势范数的起始值，默认值 0.0001。
- Vf_coeff：大于 0 浮点数，平衡价值估计损失和策略优化损失，默认值 1.0。
- grad_clip：大于 0 浮点数，剪裁渐变的全局范数，默认值为 0。
- use_gae：否将广义优势估计器（GAE）与值函数一起使用，默认值为 True。

4.2.4 模型库

- 模型名称：现有模型的名称
- 模型版本：选择具体使用哪个模型

4.2.5 模型预测

- 名称：文本框，更改强化学习模型预测算子名称，默认值为强化学习模型预测。
- stop_iters：文本框，类型为数值型，设置模型预测终止局数，默认值为 10。
- 并行预测数量：文本框，类型为数值型，设置模型预测数量，默认值为 1。

5. 附录三：镜像以及相关包说明

（一）jhinno/tensorflow:2.13

- 镜像说明：Tensorflow 框架内置镜像，用于 Tensorflow 任务提交和方案设计运行。
- 主要库版本说明：

名称	版本
python	3.10.12
tensorflow	2.13.0
keras	2.13.1
cuda	12.2
cudnn	8.9.5.29-1
nccl	2.19.3
tensorrt	8.6.1.6
horovd	0.28.1

（二）jhinno/pytorch:2.1.0

- 镜像说明：Pytorch 框架内置镜像，可用于 Pytorch 任务提交。
- 主要库版本说明：

名称	版本
python	3.10.12
torch	2.1.0
cuda	12.2
cudnn	8.9.5.27
nccl	2.18.5-1
tensorrt	8.6.1.6
horovd	0.28.1

（三）jhinno/mxnet:1.9.1

- 镜像说明：提交 mxnet 作业所用镜像。

- 主要库版本说明：

名称	版本
python	3.10.12
mxnet	1.9.1
cuda	12.2
cudnn	8.9.5.29
nccl	2.19.3-1
tensorrt	8.6.1.6
horovd	0.28.0

（四）jhinno/paddle:2.5.0

- 镜像说明：PaddlePaddle 框架镜像，可用于 PaddlePaddle 任务提交。
- 主要库版本说明：

名称	版本
python	3.10.12
paddlepaddle	2.5.0
cuda	12.2
cudnn	8.9.5.29
nccl	2.19.3-1
tensorrt	8.6.1.6

（五）jhinno/mindspore:2.2.10

- 镜像说明：Mindspore 框架镜像，可用于 Mindspore 任务提交。
- 主要库版本说明：

名称	版本
python	3.7.5
mindspore	2.0.0
mindspore-serving	2.0.0
cuda	11.6

cudnn	8.4.0.27
nccl	2.12.10-1

（六）jhinno/webide-vscode:1.95.3

➢ 镜像说明：vscode 开发环境镜像，提供 vscode web 版，内置 conda 环境已预装 tensorflow 和 pytorch，可以通过 conda activate 激活需要的环境。

➢ 主要库版本说明：

名称	版本	备注
code-server	4.6.0	/opt/code-server-4.6.0-linux-amd64
miniconda	4.4.0	/opt/miniconda3
jupyterlab	4.2.3	
jupyter_core	5.7.2	
cuda	11.8	
cudnn	8.9.6.50	
nccl	2.15.5-1	
tensorflow	2.13.0	/opt/miniconda3/envs/tensorflow
torch	2.3.1	/opt/miniconda3/envs/pytorch

（七）jhinno/webide-jupyter:4.4.5

➢ 镜像说明：JupyterLab 开发环境镜像，提供 jupyterLab web 版，内置 conda 环境已预装 tensorflow 和 pytorch，可以通过 conda activate 激活需要的环境。

➢ 主要库版本说明：

名称	版本	备注
miniconda	4.4.0	/opt/miniconda3
jupyter_core	5.7.2	
cuda	11.8	
cudnn	8.9.6.50	

nccl	2.15.5-1	
tensorrt	8.6.1.6	
tensorflow	2.13.0	/opt/miniconda3/envs/tensorflow
torch	2.3.1	/opt/miniconda3/envs/pytorch

（八）jhinno/tf-serving:2.13.0

➢ 镜像说明：tensorflow-serving 镜像，用于部署 tensorflow serving 模型推理服务。

名称	版本
tensorflow_model_server	2.13.0

（九）jhinno/torch-serving:0.11.0

➢ 镜像说明：torch-serving 镜像，用于部署 torch serving 模型推理服务。

名称	版本
torchserve	0.11.0

（十）jhinno/ubuntu-desktop:22.04

➢ 镜像说明:ubuntu 桌面开发环境镜像,内置 conda 环境已预装 tensorflow 和 pytorch，可以通过 conda activate 激活需要的环境。

➢ 主要库版本说明：

名称	版本	备注
ubuntu	22.04	
pycharm	community-2022.3	/opt/pycharm-community-2022.3
miniconda	4.4.0	/opt/miniconda3
cuda	12.2	

cudnn	8.9.6.50	
nccl	2.15.5-1	
tensorflow	2.15.0	/opt/miniconda3/envs/tensorflow
torch	2.3.1	/opt/miniconda3/envs/pytorch

（十一）jhinno/centos-desktop:7.9

➢ 镜像说明:centos 桌面开发环境镜像,内置 conda 环境已预装 tensorflow 和 pytorch，可以通过 conda activate 激活需要的环境。

➢ 主要库版本说明：

名称	版本	备注
centos	7.9	
pycharm	community-2022.3	/opt/pycharm-community-2022.3
miniconda	4.4.0	/opt/miniconda3
cuda	12.2	
cudnn	8.9.6.50	
nccl	2.15.5-1	
tensorflow	2.15.0	/opt/miniconda3/envs/tensorflow
torch	2.3.1	/opt/miniconda3/envs/pytorch

（十二）jhinno/mindinsight:2.2.10

➢ 镜像说明：mindsight 镜像，mindspore 框架可视化工具。

名称	版本
mindsight	2.2.10

（十三）jhinno/tensorboard:2.5.0

- 镜像说明：tensorboard 镜像，用于模型和训练指标可视化。

名称	版本
tensorboard	2.5.0

（十四）jhinno/visualdl:2.5.3

- 镜像说明：visualdl 镜像，用于 PaddlePaddle 训练数据、参数等信息的可视化。

名称	版本
VisualDL	2.5.3

（十五）jhinno/tritonserver:2.34.0

- 镜像说明：tritonserver 镜像，用于部署 Triton Inference Server 模型推理服务。

名称	版本
Triton Inference Server	2.34.0

（十六）jhinno/deepspeed:0.16.4

- 镜像说明：DeepSpeed 镜像。一个深度学习优化库,旨在提供高效、可扩展的大规模模型训练能力。

名称	版本
deepspeed	0.16.4

（十七）jhinno/modelstore:1.7.0

- 镜像说明：模型格式转换镜像

（十八）jhinno/ray:2.7

- 镜像说明：强化学习方案镜像

名称	版本

ray	2.7

（十九）jhinno/rstudio:4.4.1

➢ 镜像说明： R 编程语言的集成开发环境。包含使用 R 所需的所有工具位置（控制台、源、绘图、工作区、帮助、历史记录等）。具有代码完成功能的语法高亮编辑器。可以直接从源代码编辑器（行、选定内容或文件）执行代码。

➢ 主要库版本说明：

名称	版本
R 语言	4.4.1
rstudio-server	2024.09.0-375
pandoc	3.5.1
quarto	1.5.57
cuda	11.6
cudnn	8.4.0.27
nccl	2.12.10-1

6. 附录四：libaiflow API

（一）实验管理相关 API

实验初始化

```
 libaiflow.init(experiment_name: str, experiment_tags:
Optional[Dict[str, Any]] = None, token: str = None) ->
mlflow.entities.experiment.Experiment
```

参数：

experiment_name：实验名称，实验名称具有唯一性且区分大小写，一个有效的实验名称可以包含 a-z、A-Z、全中文、0-9、_和-，最大长度为 255 个字符。

experiment_tags：设置实验标记。

返回：

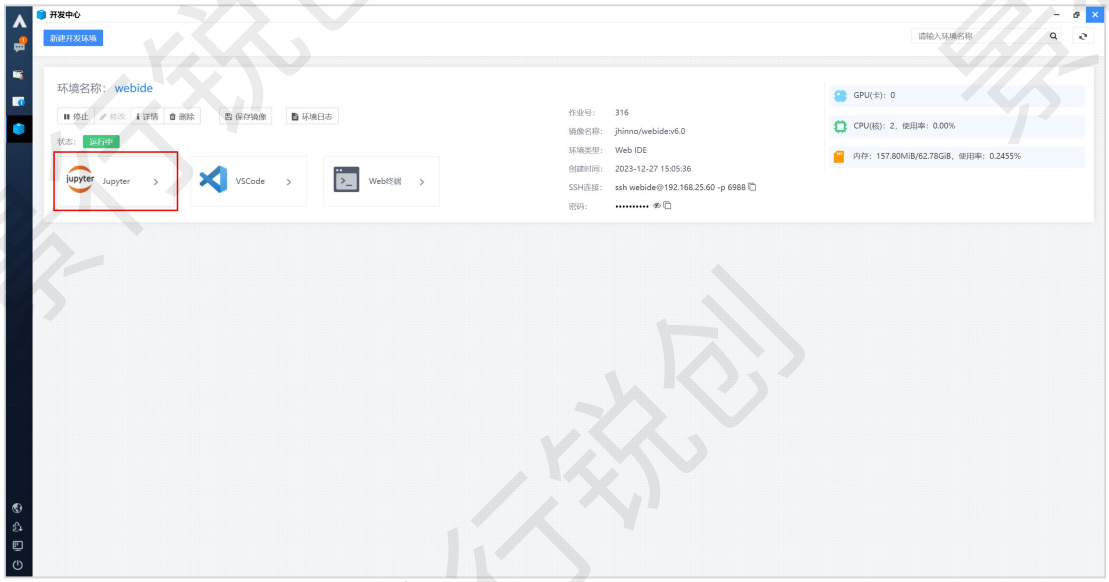
返回实验（Experiment）对象

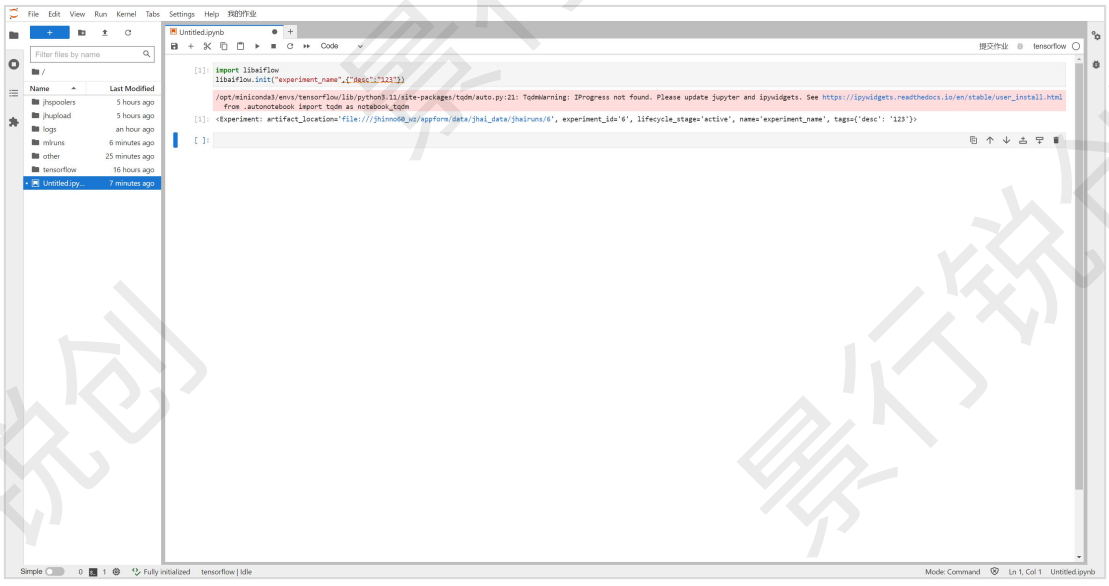
样例：

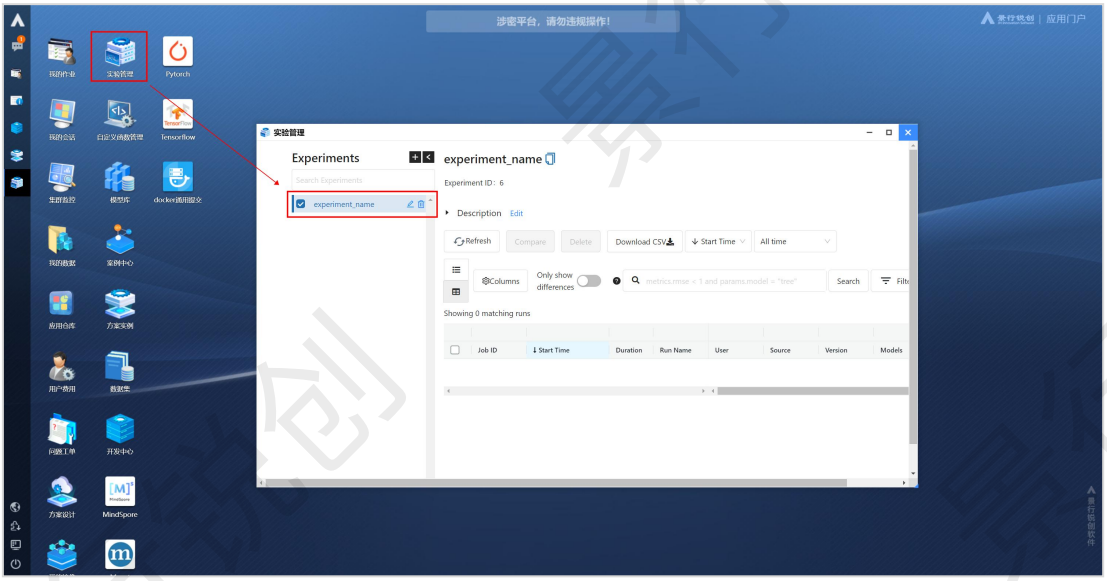
```
import libaiflow

libaiflow.init("experiment_name",{"desc":"123"})
```

具体操作详情，如下图所示：







启动一个运行实例

启动一个新的运行，将其设置为活动运行，在活动运行下将记录指标和参数。返回值可以结合 with 块一起使用；否则，必须调用 end_run()来终止当前运行。（类似文件 IO 的 open 和 close），使用 with 方式可避免忘记结束 run。

```
libaiflow.start_run(experiment_id: Union[str, NoneType], run_name: Union[str, NoneType] = None, nested: bool = False, tags: Union[Dict[str, Any], NoneType] = None, description: Union[str, NoneType] = None) -> ActiveRun
```

参数：

experiment_id：运行实例所属实验的实验 id，此参数是必须的。

run_name：运行实例名称。

nested：控制“运行”是否嵌套在“父运行”中。如果 True 创建嵌套运行。

tags：一个可选的字典，包含在运行时设置为标记的字符串键和值。如果正在恢复运行，则在恢复的运行上设置这些标记。如果正在创建新运行，则在新运行上设置这些标记。

description：填充运行的描述框的可选字符串。如果正在恢复运行，则在恢复的运行上设置描述。如果正在创建新运行，则在新运行上设置描述。

返回：

返回 libaiflow.ActiveRun 对象，封装的运行的状态。

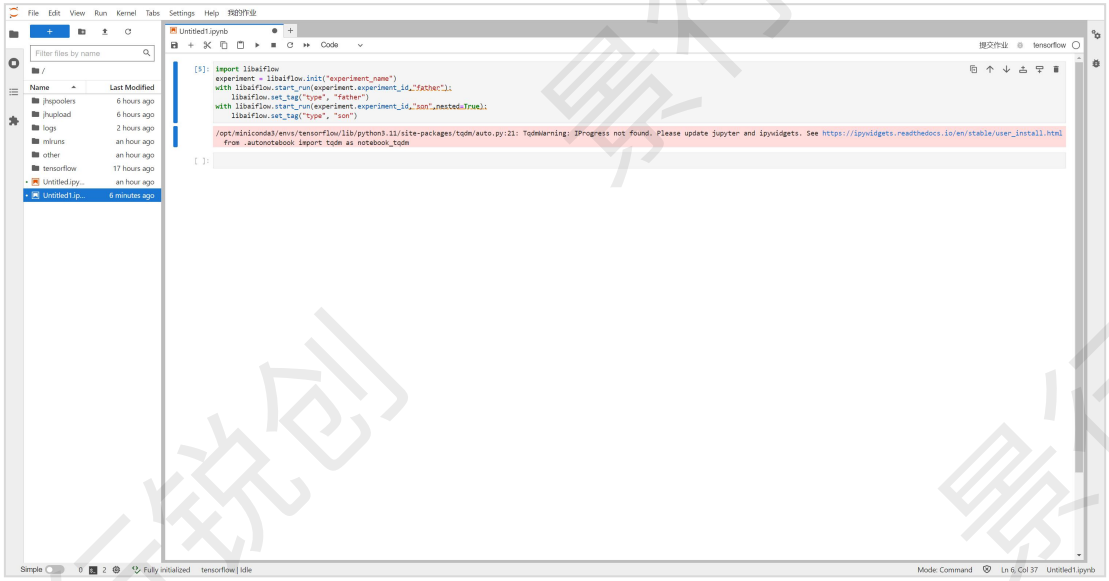
样例：

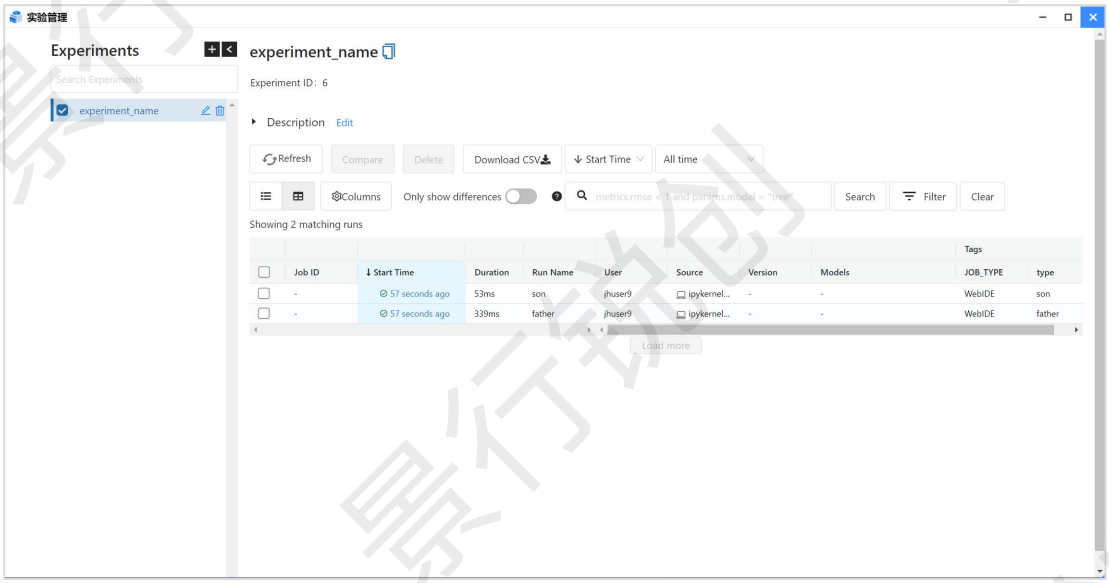
```
experiment = libaiflow.init("experiment_name")
libaiflow.start_run(experiment.experiment_id,"father")
libaiflow.set_tag("type", "father")
libaiflow.start_run(experiment.experiment_id,"son",nested=True)
libaiflow.set_tag("type", "son")
libaiflow.end_run()
libaiflow.end_run()
```

或者

```
import libaiflow
experiment = libaiflow.init("experiment_name")
with libaiflow.start_run(experiment.experiment_id,"father"):
libaiflow.set_tag("type", "father")
with
libaiflow.start_run(experiment.experiment_id,"son",nested=True):
libaiflow.set_tag("type", "son")
```

具体操作详情，如下图所示：





（二）指标和超参数相关 API

记录当前运行下的指标。如果没有活动的运行，此方法将创建一个新的活动运行。

```
libaiflow.log_metric(key: str, value: float, step: Optional[int] = None) -> None>
```

参数：

key：Metric 名称(String)。该字符串只能包含字母数字、下划线(_)、破折号(-)、句号(.)、空格()和斜线(/)。所有后端存储都将支持长度不超过 250 的键。

value：Metric 值(float)。所有后端存储将支持长度不超过 5000 的值。

step：Metric step(int)如果未指定，默认为零。

返回：

无返回值

样例：

```
experiment = libaiflow.init("experiment_name")
with libaiflow.start_run(experiment.experiment_id):
log_metric("mse", 2500.00)
```

记录当前运行实验下的多个指标。如果没有活动的运行，此方法将创建一个新的活动运行。

```
libaiflow.log_metrics(metrics: Dict[str, float], step: Union[int,
NoneType] = None) -> None
```

参数：

metrics：字典类型的 Metric: Metric 名称(String) → Metric 值(Float)；

step：记录 Metric 步长值。如果未指定，则在步骤 0 记录每个指标。

返回：

无返回值

样例：

```
experiment = libaiflow.init("experiment_name")
metrics = {"mse": 2500.00, "rmse": 50.00}
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_metrics(metrics)
```

记录当前运行下的一个参数。如果没有活动的运行，此方法将创建一个新的

活动运行。

```
libaiflow.log_param(key: str, value: Any) -> None
```

参数：

key：参数名称(String)。该字符串只能包含字母数字、下划线(_)、破折号(-)、句号(.)、空格()和斜线(/)。所有后端存储都将支持长度不超过 250 的键。

value：参数值(字符串，如果不是将被字符串化)。所有后端存储将支持长度不超过 5000 的值。

返回：

无返回值

样例：

```
experiment = libaiflow.init("experiment_name")
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_param("learning_rate", 0.01)
```

记录当前运行的一批参数。如果没有活动的运行，此方法将创建一个新的活动运行。

```
libaiflow.log_params(params: Dict[str, Any]) -> None
```

参数：

params：字典类型的 param: Param 名称（String） → Param 值：(字符串，如果不是将被字符串化)。

返回：

无返回值

样例：

```
experiment = libaiflow.init("experiment_name")
```

```
params = {"learning_rate": 0.01, "n_estimators": 10}
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_params(params)
```

（三）文件和可视化相关 API

将本地文件或目录记录为当前活动运行的 Artifact。如果没有活动的运行，此方法将创建一个新的活动运行。

```
libaiflow.log_artifact(local_path: str, artifact_path: Union[str, NoneType] = None) -> None
```

参数：

local_path：要写入的文件的路径。

artifact_path：如果指定目录，将写入到 artifact_uri 中的目录。

返回：

无返回值

样例：

```
experiment = libaiflow.init("experiment_name")
features = "rooms, zipcode, median_price, school_rating, transport"
with open("features.txt", 'w') as f:
    f.write(features)
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_artifact("features.txt","data")
```

日志文本作为 artifact。

```
libaiflow.log_text(text: str, artifact_file: str) -> None
```

参数：

text：包含要记录的文本的字符串。

artifact_file：文本保存的 posixpath 格式的运行相关工件文件路径(例如“dir/file.txt”)。

返回：

无返回值

样例：

```
experiment = libaiflow.init("experiment_name")
with libaiflow.start_run(experiment.experiment_id):
    # Log text to a file under the run's root artifact directory
    libaiflow.log_text("text1", "file1.txt")
    # Log text in a subdirectory of the run's root artifact directory
    libaiflow.log_text("text2", "dir/file2.txt")
    # Log HTML text
    libaiflow.log_text("<h1>header</h1>", "index.html")
```

将 figure 对象作为 Artifact

```
libaiflow.log_figure(figure:
Union[ForwardRef('matplotlib.figure.Figure'),
ForwardRef('plotly.graph_objects.Figure')], artifact_file: str) -> None
```

参数：

figure：figure 对象，支持以下 figure 对象：

- matplotlib.figure.Figure
- plotly.graph_objects.Figure

artifact_file：以 posixpath 格式保存图形的运行相关工件文件路径(例如“dir/file.png”)

返回：

无返回值

样例：

Matplotlib 样例代码如下所示：

```
import libaiflow
import matplotlib.pyplot as plt
experiment = libaiflow.init('experiment_name')
fig, ax = plt.subplots()
ax.plot([0, 1], [2, 3])
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_figure(fig, "figure.png")
```

Plotly 样例代码如下所示：

```
import libaiflow
from plotly import graph_objects as go
experiment = libaiflow.init('experiment_name')
fig = go.Figure(go.Scatter(x=[0, 1], y=[2, 3]))
with libaiflow.start_run(experiment_id= experiment. experiment_id):
    libaiflow.log_figure(fig, "figure.html")
```

将 image 对象记录为 Artifact

```
libaiflow.log_image(image:        Union[ForwardRef('numpy.ndarray'),
ForwardRef('PIL.Image.Image')], artifact_file: str) -> None
```

参数：

image：figure 对象，支持以下 image 对象：

- numpy.ndarray
- PIL.Image.Image

artifact_file：以 posixpath 格式保存图形的运行相关工件文件路径(例如“dir/file.png”)

返回：

无返回值

样例：

Numpy 样例代码如下所示：

```
import libaiflow
import numpy as np
experiment = libaiflow.init('experiment_name')
image = np.random.randint(0, 256, size=(100, 100, 3), dtype=np.uint8)
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_image(image, "image.png")
```

Pillow 样例代码如下所示：

```
import libaiflow
from PIL import Image
experiment = libaiflow.init('experiment_name')
image = Image.new("RGB", (100, 100))
with libaiflow.start_run(experiment.experiment_id):
    libaiflow.log_image(image, "image.png")
```

（四）模型相关 API

启用自动记录 tensorflow 和 tf.keras 的参数，指标和模型。

```
libaiflow.tensorflow.autolog(every_n_iter=1)
```

参数：

every_n_iter：它是训练指标的每个日志之间的训练时期数。 例如，如果传递的值为 2，则每 2 个时期记录一次训练指标(损失，准确性和验证损失等)。

返回：

没有返回值

启用自动记录 PyTorch Lightning 的参数、指标和模型。

```
libaiflow.pytorch.autolog(log_every_n_epoch=1)
```

参数：

log_every_n_epoch：它是训练指标的每个日志之间的训练时期数。 例如，如果传递的值为 2，则每 2 个时期记录一次训练指标(损失，准确性和验证损失等)。

返回：

没有返回值

（五）数据集相关 API

根据数据集名称获取数据集信息

```
libaiflow.get_dataset_by_name(dataset_name: str) ->
Optional[Dict[str, Any]]
```

参数：

dataset_name：数据集名称。

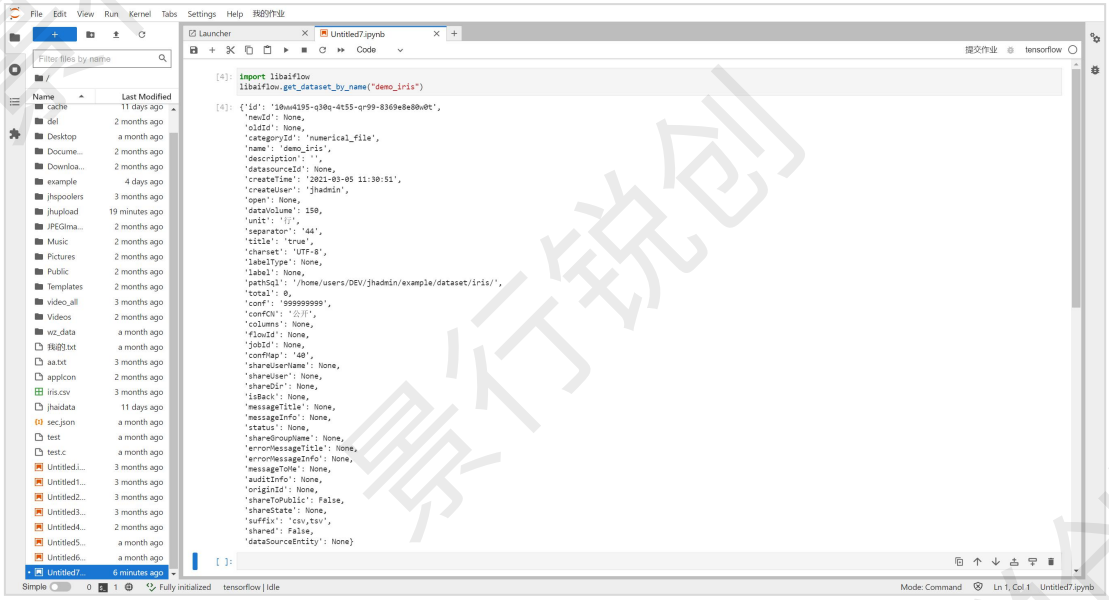
返回：

返回 JSON 格式的数据集信息

样例：

```
import libaiflow
libaiflow.get_dataset_by_name("demo_iris")
```

具体操作详情，如下图所示：



根据数据集 ID 获取数据集信息

```
Libaiflow.get_dataset_by_id(dataset_id: str) -> Optional[Dict[str, Any]]
```

参数：

dataset_id：数据集 ID

返回：

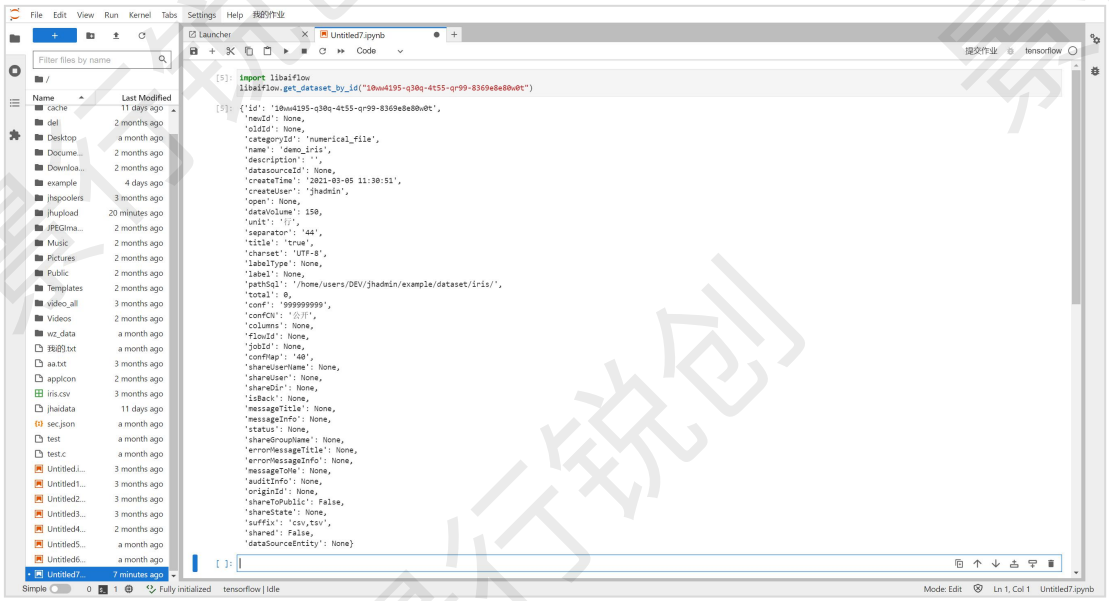
返回 JSON 格式的数据集信息

样例：

```
import libaiflow

libaiflow.get_dataset_by_id("w7wqrwey-2626-433e-8591-18r360ee7ry8")
```

具体操作详情，如下图所示：



创建数据集，目前仅支持创建“数值数据-数据文件”数据集：

```
libaiflow.create_dataset(name: Optional[str], directory_path:
Optional[str], file_suffix: Optional[List[str]] = ['csv', 'tsv'],
description: Optional[str] = '', category:
Optional[libaiflow.client.datasets.Category] =
<Category.numerical_file: 'numerical_file'>) -> Optional[Dict[str,
Any]]
```

参数：

name：数据集名称。

directory_path：数据文件路径。

file_suffix：文件后缀。

description：数据集描述。

category：数据集类型，默认为 numerical_file，用于创建“数值数据-数据文件”数据集；目前仅支持创建“数值数据-数据文件”数据集。

返回：

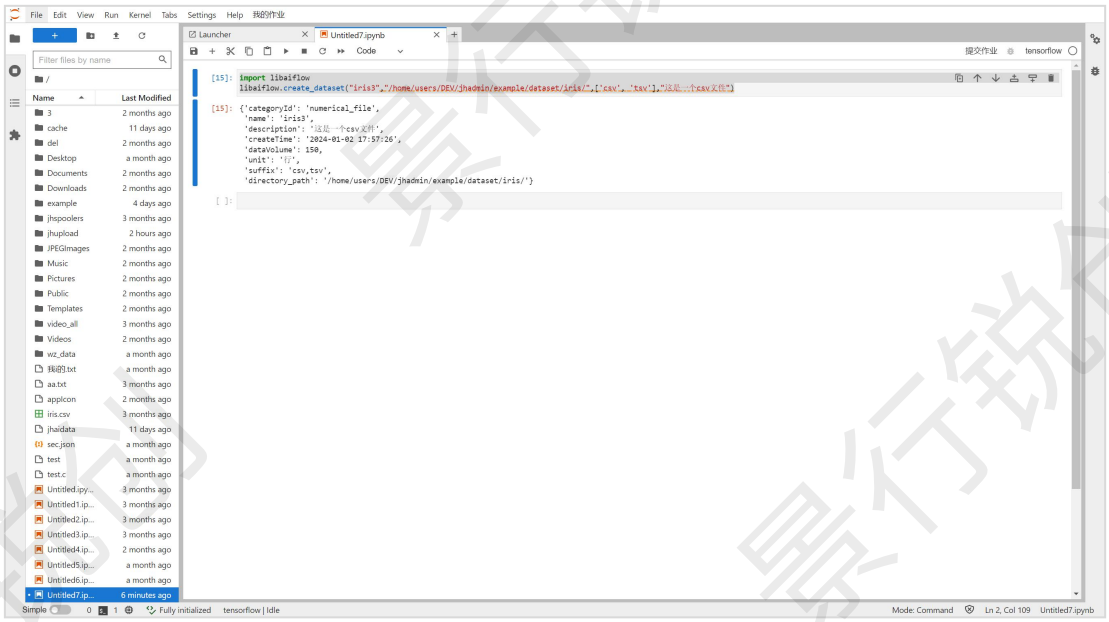
返回 JSON 格式的数据集信息

样例：

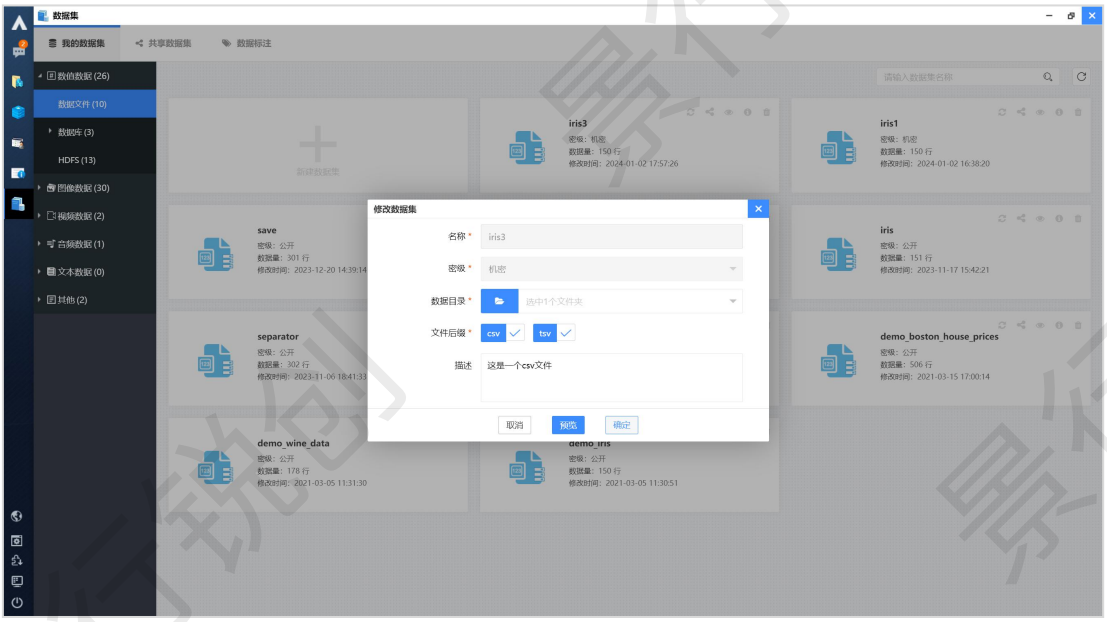
样例代码如下所示：

```
import libaiflow
libaiflow.create_dataset("iris","/home/users/DEV/jhadmin/example/dataset/iris/",['csv', 'tsv'],"这是一个 csv 文件")
```

具体操作详情，如下图所示：



创建完成后，可在我的数据集中查看。



修改数据集

```
libaiflow.modify_dataset(name: Optional[str], directory_path:
Optional[str], file_suffix: Optional[List[str]] = ['csv', 'tsv'],
description: Optional[str] = '', category:
Optional[libaiflow.client.datasets.Category] =
<Category.numerical_file: 'numerical_file'>) -> Optional[Dict[str,
Any]]
```

参数：

name：数据集名称。

file_path：数据文件路径。

description：数据集描述。

category：数据集类型，默认为 numerical_file，用于创建“数值数据-数据文件”数据集；目前仅支持创建“数值数据-数据文件”数据集。

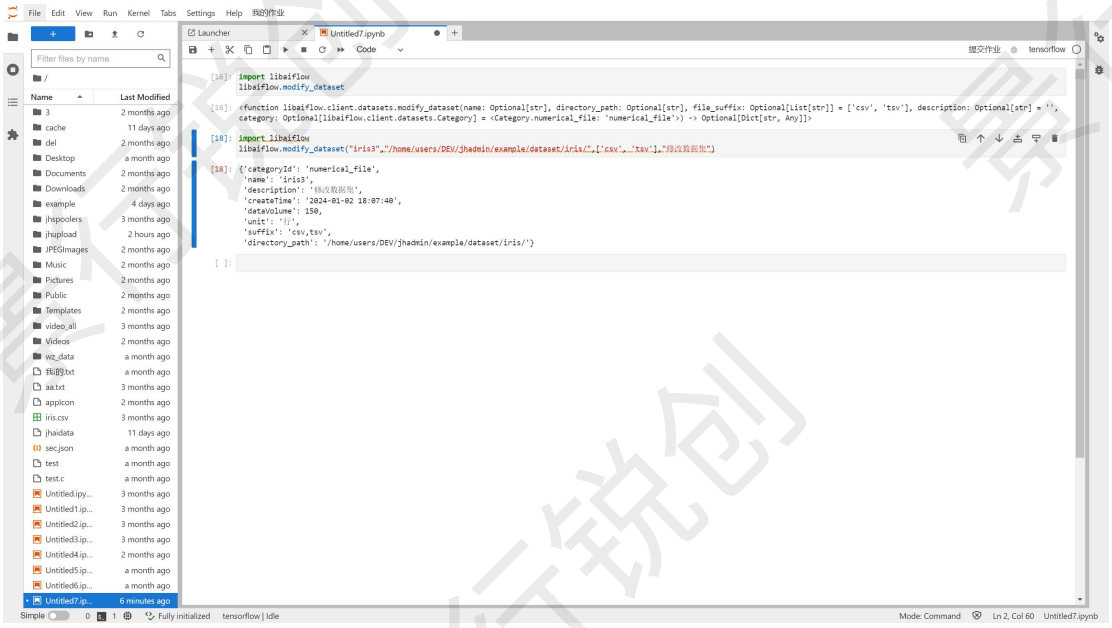
返回：

返回 JSON 格式的数据集信息

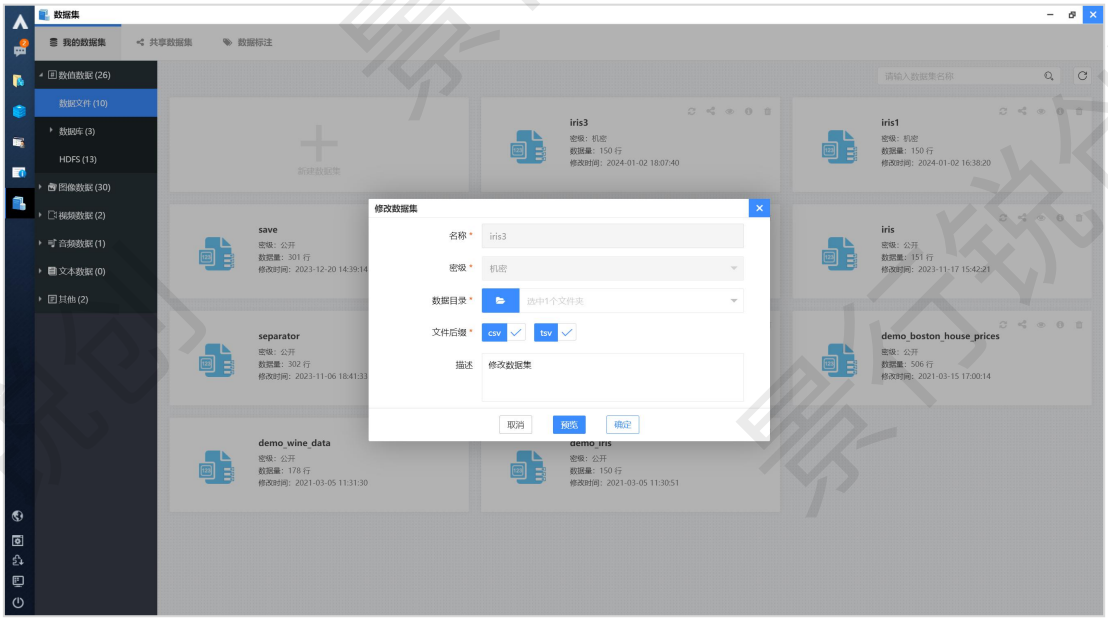
样例：

样例代码如下所示：

```
import libaiflow
libaiflow.modify_dataset("iris","/home/users/DEV/jhuser1/example/dataset/heart.csv","修改数据集
",separator=Separator.comma,title=False,charset=Charset.GB2312)
```



修改完成后，可在我的数据集中进行查看。



删除“数值数据-数据文件”数据集

```
libaiflow.delete_dataset(name: str = None)
```

参数：

name：“数值数据-数据文件”数据集名称。

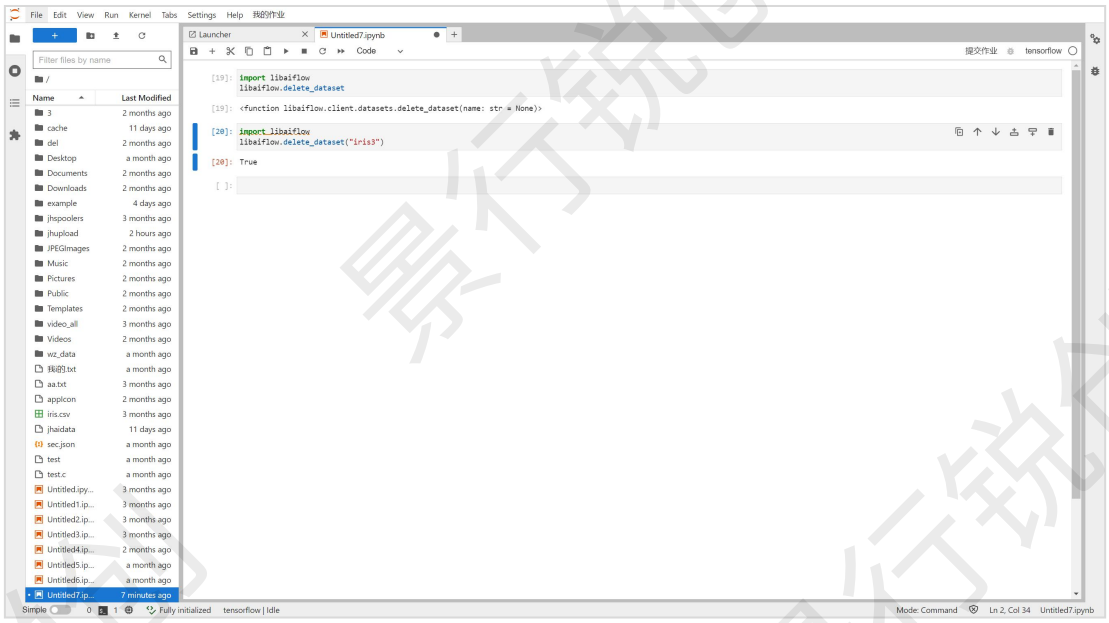
返回：

返回 True，代表着删除成功了。

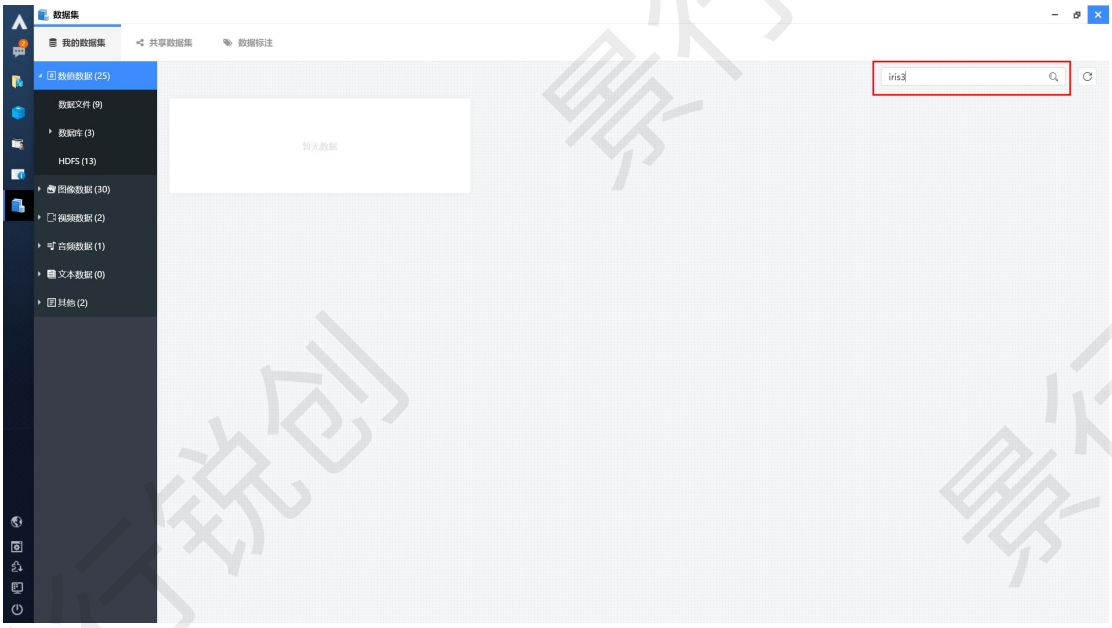
样例：

```
import libaiflow
libaiflow.delete_dataset("iris")
```

具体操作详情，如下图所示：



删除完成后，可在我的数据集中进行查看。


数据集
我的数据集
共享数据集
数据标注
数值数据 (25)
数据文件 (9)
数据库 (3)
HDFS (13)
图像数据 (30)
视频数据 (2)
音频数据 (1)
文本数据 (0)
其他 (2)
iris
暂无数据

7. 附录五：实验管理 SDK 代码样例

样例一：

```
    import tensorflow as tf
    from tensorflow.keras import layers, Sequential, optimizers, losses,
metrics, datasets
    import os
    import numpy as np
    import libaiflow
    # 预处理函数
    def preprocess(x, y):
        x = tf.cast(x, tf.float32) / 255.0 # 将 MNIST 数据映射到[0, 1]
        x = tf.expand_dims(x, axis=-1) # 由于卷积层维度为[None, 28, 28,
1], 故在 axis=3 扩展一维
        y = tf.one_hot(y, depth=10)
        return x, y

    # 加载数据
    def load_data():
      path = "./data/mnist.npz"
      with np.load(path) as f:
        x_train, y_train = f['x_train'], f['y_train']
        x_test, y_test = f['x_test'], f['y_test']
        return (x_train, y_train), (x_test, y_test)

    expt = libaiflow.init(experiment_name="mnist_tf2_cpu")
    libaiflow.tensorflow.autolog()

    (x, y), (x_test, y_test) = load_data()

    print(x.shape, y.shape, x.min(), x.max()) # 显示加载进来的数据的维
度信息，便于理解

    train_db = tf.data.Dataset.from_tensor_slices((x, y))
    train_db = train_db.shuffle(1000).map(preprocess).batch(100)

    test_db = tf.data.Dataset.from_tensor_slices((x_test, y_test))
    test_db = test_db.map(preprocess).batch(100)

    # 创建网络模型并装配模型
    network = Sequential([
        layers.Conv2D(6, kernel_size=5, strides=1, padding='SAME',
activation='relu'),
        layers.MaxPooling2D(pool_size=2, strides=2),
```

```
        layers.Conv2D(16, kernel_size=5, strides=1, padding='SAME',
activation='relu'),
        layers.MaxPooling2D(pool_size=2, strides=2),

        layers.Flatten(),

        layers.Dense(256, activation='relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(10, activation=None)
    ])
    network.compile(optimizer=optimizers.Adam(lr=0.01),

loss=losses.CategoricalCrossentropy(from_logits=True),
                    metrics=['accuracy'])

    tf.summary.trace_on() #显示结构图

    # 显示网络结构信息
    network.build(input_shape=[None, 28, 28, 1])
    network.summary()
    pb_file_path = "model"
    if not os.path.exists(pb_file_path):  #判断是否存在文件夹如果不存
在则创建为文件夹
        os.makedirs(pb_file_path)

    # 设置回调功能
    filepath = './model/my_model.h5' # 保存模型地址

    saved_model = tf.keras.callbacks.ModelCheckpoint(filepath, verbose
= 1) # 回调保存模型功能
    tensorboard = tf.keras.callbacks.TensorBoard(log_dir = './logs') #
回调可视化数据功能

    # 执行训练与验证
    history = network.fit(train_db, epochs = 2, validation_data =
test_db, validation_freq = 1,callbacks = [saved_model, tensorboard])
```

样例二：

```
import logging
import libaiflow
import pandas as pd
from sklearn.model_selection import train_test_split
import sys
```

```
    from sklearn.linear_model import ElasticNet
    import numpy as np
    from                    sklearn.metrics                    import
mean_squared_error,mean_absolute_error,r2_score

    logging.basicConfig(level=logging.WARN)
    logger = logging.getLogger(__name__)

    def eval_metrics(actual, pred):
        rmse = np.sqrt(mean_squared_error(actual, pred))
        mae = mean_absolute_error(actual, pred)
        r2 = r2_score(actual, pred)
        return rmse, mae, r2

    if __name__ == "__main__":
        np.random.seed(40)
        # Read the wine-quality csv file from the URL
        csv_url
='http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality
/winequality-red.csv'
        try:
            data = pd.read_csv(csv_url, sep=';')
        except Exception as e:
            logger.exception(
                "Unable to download training & test CSV, check your internet
connection. Error: %s", e)

        # Split the data into training and test sets. (0.75, 0.25) split.
        train, test = train_test_split(data)

        # The predicted column is "quality" which is a scalar from [3, 9]
        train_x = train.drop(["quality"], axis=1)
        test_x = test.drop(["quality"], axis=1)
        train_y = train[["quality"]]
        test_y = test[["quality"]]
        alpha = 0.5
        l1_ratio = 0.5
        expt = libaiflow.init(experiment_name="mnist_tf2_test1")
        with libaiflow.start_run(expt.experiment_id):
            lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
            lr.fit(train_x, train_y)
            predicted_qualities = lr.predict(test_x)
            (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
            print("Elasticnet  model  (alpha=%f,  l1_ratio=%f):"  %  (alpha,
```

```
l1_ratio))
          print("  RMSE: %s" % rmse)
          print("  MAE: %s" % mae)
          print("  R2: %s" % r2)
          libaiflow.log_param("alpha", alpha)
          libaiflow.log_param("l1_ratio", l1_ratio)
          libaiflow.log_metric("rmse", rmse)
          libaiflow.log_metric("r2", r2)
          libaiflow.log_metric("mae", mae)
```

8. 附录六：自动调参组件属性说明

（一）搜索算法

1. RandomMultiObjectiveSampler：随机采样多目标搜索算法

- "seed":"None" #随机数种子，默认为 None

2. NSGAIIMultiObjectiveSampler：基于 NSGA-II 算法的多目标搜索算法

- "population_size":50, #每一代中的 个体（trial）数，默认为 50
- "mutation_prob":"None", #在创建新个体时每个参数突变的概率。如果设置成 None 的话，那么该值实际上会是 1.0 / len(parent_trial.params) 。其中 parent_trial 是 target trial 的父代 trial，默认为 None
- "crossover_prob":0.9, #产生新的个体时发生交叉互换的概率，默认为 0.9
- "swapping_prob":0.5, #在交叉互换中交换父代参数中每一个参数的概率，默认为 0.5
- "seed":"None" #随机数生成器种子，默认为 None

3. MOTPEMultiObjectiveSampler：基于 MOTPE 算法的多目标搜索算法

- "consider_prior":["True","False"] #当 True 时通过一个高斯先验来提升 Parzen estimator 的稳定性。这个先验只有当 sampling 分布是以下这些分布中间的一个时起作用： UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution, IntUniformDistribution, 或者 IntLogUniformDistribution.
- "prior_weight":1.0, #先验的权重，该选项用于 UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution, IntUniformDistribution, IntLogUniformDistribution, 和 CategoricalDistribution 中。
- "consider_magic_clip":["True","False"] #启用一个启发式算法以限制 Parzen estimator 所使用的高斯分布的最小方差。
- "consider_endpoints":["True","False"] #在 Parzen estimator 中计算高斯方差时，请考虑域的端点。

- "n_startup_trials":10, #在给定的 trial 次数在同一 study 中完成之前，将使用随机抽样代替 TPE 算法，默认为 10
- "n_ehvi_candidates":24, #用于计算预期改进的候选样本数，默认为 24
- "seed":"None" #随机采样数，默认为 None

4. TPESampler：基于 TPE (Tree-structured Parzen Estimator) 算法的搜索算法

- "consider_prior":["True","False"] #当 True 时通过一个高斯先验来提升 Parzen estimator 的稳定性。这个先验只有当 sampling 分布是以下这些分布中间的一个时起作用： UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution, IntUniformDistribution, 或者 IntLogUniformDistribution.
- "prior_weight":1.0,#先验的权重，该选项用于 UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution, IntUniformDistribution, IntLogUniformDistribution, 和 CategoricalDistribution 中。
- "consider_magic_clip":["True","False"] #先验的权重，该选项用于 UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution, IntUniformDistribution, IntLogUniformDistribution, 和 CategoricalDistribution 中。
- "consider_endpoints":["True","False"] #启用一个启发式算法以限制 Parzen estimator 所使用的高斯分布的最小方差。
- "n_startup_trials":10, #在给定的 trial 次数在同一 study 中完成之前，将使用随机抽样代替 TPE 算法，默认为 10
- "n_ei_candidates":24, #用于计算预期改进的候选样本数，默认为 24
- "seed":"None", #随机采样数，默认为 None
- "multivariate":["False","True"] #当设置为 True 时，使用多元 TPE 算法

- "group":["False","True"] #当 multivariate 参数和该参数都为 True 时，建议参数时使用具有组分解搜索空间的多元 TPE。
- "warn_independent_sampling":["True","False"] # 如果这是 True，那么当一个参数的值是通过独立 sampler 来采样时，它会触发一个 warning 信息。
- "constant_liar":["False","True"] #运行试验中是否启用惩罚，以避免临近值造成的影响。
- "constraints_func":"None" #一个可选函数，用于计算目标约束

5. QMCSampler：一种生成低差异序列的 QMC 搜索算法

- "qmc_type":["sobol","halton"] #采样 QMC 序列的类型，必须是“halton”和“sobol”之一。 默认为“sobol”。
- "scramble":["False","True"] #该参数为 True 时，随机采样被应用于 QMC 序列。
- "seed":"None", #随机采样数，默认为 None
- "independent_sampler":"None", #``optuna.samplers.BaseSampler`实例，别用于采用算法中。
- "warn_asynchronous_seeding":["True","False"] #该参数为 True 时，QMC 序列应用随机采样且为手动设置随机种子时，会发出警告消息。
- "warn_independent_sampling":["True","False"] #该参数为 True 时，使用独立采样器对参数值进行采样时会发出警告消息。

6. MOTPESampler：基于 MOTPE 算法的搜索算法

- "consider_prior":["True","False"] #当 True 时通过一个高斯先验来提升 Parzen estimator 的稳定性。这个先验只有当 sampling 分布是以下这些分布中间的一个时起作用： UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution, IntUniformDistribution, 或者 IntLogUniformDistribution.
- "prior_weight":1.0, #先验的权重，该选项用于 UniformDistribution, DiscreteUniformDistribution, LogUniformDistribution,

IntUniformDistribution, IntLogUniformDistribution, 和 CategoricalDistribution 中。

- "consider_magic_clip":["True","False"] #启用一个启发式算法以限制 Parzen estimator 所使用的高斯分布的最小方差。
- "consider_endpoints":["True","False"] #在 Parzen estimator 中计算高斯方差时，请考虑域的端点。
- "n_startup_trials":10, #在给定的 trial 次数在同一 study 中完成之前，将使用随机抽样代替 TPE 算法，默认为 10
- "n_ehvi_candidates":24, #用于计算预期改进的候选样本数，默认为 24
- "seed":"None" #随机采样数，默认为 None

7. RandomSampler：随机搜索算法.

- "seed":"None" #随机采样数，默认为 None

8. GridSampler：基于网格的搜索算法.

- "search_space":"", #一个键为参数名值为对应的待选参数的字典。
- "seed":"None" #随机数种子，默认为 None

9. CmaEsSampler：基于 CMA-ES 算法的搜索算法

- "x0":"None", #一个包含了 CMA-ES 的初始参数的字典。在默认情况下，每个分布的 low and high 的平均数是被采用的。
- "sigma0":"None", # CMA-ES 的初始标准差。默认情况下， sigma0 会被设置成 min_range / 6, 而 min_range 代表着搜索空间中最小的分布范围。
- "n_startup_trials":1, #除非同一个 study 中指定个数的 trial 已经完成，否则将会采用独立 sampling 而不是 CMA-ES 算法。
- "independent_sampler":"None", #一个 BaseSampler 实例，它用于独立 sampling.那些并不包含在相对搜索空间中的参数是通过这个 sampler 来采样的。CmaEsSampler 的搜索空间 是通过 intersection_search_space() 来确定的。如果设置成 None 的话，RandomSampler 会作为默认 sampler 被使用。.. seealso:: optuna.samplers 模块提供了内置的独立 sampler, 比如 RandomSampler 和 TPESampler.

- "warn_independent_sampling":["True","False"] #如果这是 True，那么当一个参数的值是通过独立 sampler 来采样时，它会触发一个 warning 信息。注意，由于一个 study 中的第一个 trial 的参数总要通过独立 sampler 来采样，因此这种情况下 warning 信息不会被触发。
- "seed":"None", #CMA-ES 的随机数种子。
- "consider_pruned_trials":["False","True"] #该参数为 True 时，则考虑对 PRUNED 试验进行抽样。
- "restart_strategy":"None", #收敛到局部最小值时重启 CMA-ES 优化策略，设置为 None 时，不重启 CMA-ES 优化策略;如果设置为 ipop，将随着增加的种群大小重启 CMA-ES 优化策略.
- "popsize":"None", #CMA-ES 的种群大小，当 restart_strategy 设置为 ipop 时生效。
- "inc_popsize":2, #每次重新启动前种群增加的倍数，当 restart_strategy 设置为 ipop 时生效。
- "use_separable_cma":["False","True"]#该参数为 True 时，协方差矩阵被限制为对角线。
- "source_trials":"None" #此选项适用于 Warm Starting CMA-ES，这是一种通过 CMA-ES 的初始化来转移类似 HPO 任务的先验知识的方法。该方法从 ``source_trials`` 估计一个有希望的分布并生成多元高斯分布的参数。 请注意，禁止同时使用“x0”、“sigma0”或“use_separable_cma”参数。

10. BruteForceSampler：使用蛮力的采样器。此采样器对定义的搜索空间执行详尽搜索。

- "seed":"None" #随机数种子。

（二）剪枝算法

1. NopPruner：不修剪 trial 的 pruner

2. PercentilePruner：保留指定百分位 trial 的 pruner，如果在同一步骤的 trial 中，最佳中间值位于最低百分位数，则进行剪枝。

- "percentile":25, #百分位数必须介于 0 到 100 之间（例如，如果给定 25.0，保留第 25 个百分位数 trial 的顶部）

- "n_startup_trials":5, #剪枝将被禁用，直到在同一 study 中完成给定的 trial 次数为止。
- "n_warmup_steps":0, #在 trial 超过给定步骤数之前，将禁用剪枝功能。
- "interval_steps":1 #不同剪枝检查之间的间隔步骤，预热步骤不算在其中。 如果在剪枝检查时未报告任何值，则该特定检查将被推迟，直到报告了一个值。该值最小为 1.

3. HyperbandPruner：使用 hyperband 的 pruner。

- "min_resource":1, #用于指定分配给单个 trial 的最小资源 (在 原文中 该参数是 r).一个更小的 r 会更快返回结果，但是 一个更大的 r 能保证更佳的不同配置之间的判定。
- "max_resource":"auto", # 用于指定分配给一个 trial 的最大资源参数。原文中的 R 对应着 max_resource / min_resource. 该值代表着且应符合最大迭代步数（也就是一个神经网络的 epoch 数）。当该参数是 "auto" 时，最大资源是根据已经完成的 trial 来预估的。 .. note::"设置成 "auto" 时，最大资源将是头一个（如果在并行优化的情况下，就是头几个中的一个）已完成的 trial 的 report() 报告的最大步数。如果每个 trial 的最后中间值的步骤不同的话，请 手动设定最大的可能步骤 max_resource。
- "reduction_factor":3, # 用于设置可提升 trial 的缩减因子参数。
- "bootstrap_count":0 #指定在可以提升任何试验之前梯级中所需试验次数的参数。

4. MedianPruner：使用中值停止规则 的 pruner。如果该试验的最佳中间结果比同一步骤中先前试验的中间结果的中值差，则进行剪枝。

- "n_startup_trials":5, #剪枝将被禁用，直到在同一 study 中完成给定的 trial 次数为止。
- "n_warmup_steps":0, #在 trial 超过给定步骤数之前，将禁用剪枝功能。

- "interval_steps":1, #不同剪枝检查之间的间隔步骤，预热步骤不算在其中。 如果在剪枝检查时未报告任何值，则该特定检查将被推迟，直到报告了一个值。
- "n_min_trials":1 #最小的判断是否兼职的上报试验结果数量。

5. PatientPruner：利用公差包装另外一个剪枝器生成的新剪枝算法。

- "pruner":"MedianPruner", #当参数为``~optuna.pruners.PatientPruner`类时，允许对试验进行修剪，包装修剪器执行修剪。 如果‘None’，这个 pruner 就相当于 early-stopping 在 individual trial 中取了中间值。
- "patience":1 #修剪被禁用，直到目标没有因“patience”连续步骤而改善。

6. ThresholdPruner：用于检测 trial 的无关度量的 pruner。如果一个度量超过了上限，低于下限或者变成了 nan, 就剪枝。

- "lower":"None", #一个确定 pruner 是否需要剪枝的最小值。如果某个中间值小于此值，就剪枝。
- "upper":"None", #一个确定 pruner 是否需要剪枝的最大值。如果某个中间值大于此值，就剪枝。
- "n_warmup_steps":0, #在 trial 超过给定步骤数之前，将禁用剪枝功能。
- "interval_steps":1 #不同剪枝检查之间的间隔步骤，预热步骤不算在其中。 如果在剪枝检查时未报告任何值，则该特定检查将被推迟，直到报告了一个值。该值最小为 1。

7. SuccessiveHalvingPruner：使用异步连续减半算法的 pruner。Successive Halving 是一种基于 bandit 的算法，用于识别多种配置中的最佳配置。

- "min_resource":"atuo", #用于指定分配给单个 trial 的最小资源。默认情况下，该参数是 'auto'，此时，其值是由一个启发式算法设定的，该算法会观察完成第一个 trial 所需要的步数。

- "reduction_factor":4, # 用于设置可提升 trial 的缩减因子参数。在每一级的完成点，大约有 1/reduction_factor 的 trial 会被提升。
- "min_early_stopping_rate":0, #用于确定最小提前终止率的参数。
- "bootstrap_count":0 #指定在可以提升任何试验之前梯级中所需试验次数的参数。

（三）超参数设置

1. int：整型搜索方式

- "low":1, #推荐值的下限，下限需小于上限，数值类型为整型
- "high":10, #推荐值的上限，下限需大于上限，数值类型为整型
- "step":1, #步长，数值类型为整型
- "log":["False","True"] #对日志域中的值是否进行采样

2. float：浮点型搜索方式

- "low":1, #推荐值的下限，下限需小于上限，数值类型为浮点型
- "high":10, #推荐值的上限，下限需大于上限，数值类型为浮点型
- "step":1, #步长，数值类型为浮点型
- "log":["False","True"] #对日志域中的值是否进行采样

3. uniform：连续均匀采样搜索方式

- "low":1, #推荐值的下限，下限需小于上限，数值类型为浮点型
- "high":10 #推荐值的上限，下限需大于上限，数值类型为浮点型

4. loguniform：对数均匀采样搜索方式

- "low":1, #推荐值的下限，下限需小于上限，数值类型为浮点型
- "high":10 #推荐值的下限，下限需小于上限，数值类型为浮点型

5. discrete_uniform：离散均匀采样搜索方式

- "low":1,#推荐值的下限，下限需小于上限，数值类型为浮点型
- "high":10,#推荐值的下限，下限需小于上限，数值类型为浮点型
- "q":1 #离散化步长，数值类型为浮点型

6. categorical：枚举型搜索方式

- "choices":[1,2,3]#搜索范围，输入列表，列表元素可以为字符串或者数值

（四）回调函数

1. KerasPruningCallback(monitor: str, interval: int = 1) keras 剪枝回调

- monitor：剪枝的回调函数，如“val_loss”和“val_accuracy”；
- interval:每隔 interval 个 epoch 剪枝,当 interval=1,表示每个 epoch 都检查是否剪枝;

2. MXNetPruningCallback(eval_metric:str)mxnet 剪枝回调

- eval_metric：剪枝的回调函数，如“val_loss”和“val_accuracy”；

3. PaddlePruningCallback(monitor: str, interval: int = 1)paddle 剪枝回调

- monitor：剪枝的回调函数，如“val_loss”和“val_accuracy”；
- interval:每隔 interval 个 epoch 剪枝,当 interval=1,表示每个 epoch 都检查是否剪枝;

4. TorchDistributedTrial(group)PyTorch 分布式训练剪枝回调

- group:`torch.distributed.ProcessGroup`组，默认为 None

5. PyTorchLightningPruningCallback(monitor: str) PyTorch Lightning 剪枝回调

- monitor：剪枝的回调函数，如“val_loss”和“val_accuracy”；

6. OptunaSearchCV sklearn 剪枝会掉

7. TensorFlowPruningHook(metric: str,run_every_steps: int) tensorflow 框架剪枝回调

- metric：剪枝的回调函数，如“val_loss”和“val_accuracy”；
- run_every_steps：每隔 run_every_steps 个 epoch 剪枝，当 run_every_steps=1，表示每个 epoch 都检查是否剪枝;

8. TFKerasPruningCallback() tensorflow keras 剪枝回调

- monitor：剪枝的回调函数，如“val_loss”和“val_accuracy”；

9. 附录七：Web 终端类型支持 SSH 连接的镜像说明

新建 Web 终端类型的开发环境，并且支持 SSH 连接功能，需要在镜像中安装 openssh，并修改配置。

示例如下：

```
# Dockerfile
FROM ubuntu  #基础镜像

RUN apt update -y && apt install openssh-server -y && \
    sed -i
'/^#PasswordAuthentication/s/#PermitRootLogin.*/PermitRootLogin
yes/g' /etc/ssh/sshd_config && \
    sed -i 's/#PermitRootLogin.*/PermitRootLogin yes/g'
/etc/ssh/sshd_config   #安装 openssh 相关文件

CMD ["bash"]
```

10. 附录八：常见问题及解决办法

(1) 如何在容器服务中使用用户自己的数据。

解决方案：

容器服务启动时会默认挂载用户家目录、作业数据区、共享数据区，可以在启动的容器服务中直接访问用户家目录、作业数据区、共享数据区下的数据文件，目录绝对路径和宿主机保持一致。

(2) 如何保存在容器服务中创建的文件。

解决方案：

容器服务启动时会默认挂载用户家目录、作业数据区和共享数据区，可以将需要保存的文件拷贝到用户家目录、作业数据区或者共享数据区下。

(3) 如何在离线内网环境安装额外的 pip 或 conda 依赖。

解决方案：

可以从外网下载 pip、 conda 的离线包，导入内网安装。或者购买景行锐创离线 pip、conda 源服务在内网搭建源，方便安装依赖。

(4) 如何在非 root 启动的镜像中安装依赖包。

解决方案：

使用 sudo 执行 yum 或 apt-get 命令。

(5) 模型训练任务异常退出，显示 out of memory

解决方案：

调整训练程序代码或数据大小。

(6) 容器服务启动失败报错，output.txt 中提示：“service start failed,current status is: rejected, reason is: No such image: ahaha/aaa:v_testa1”

原因：

创建容器服务时，选择/输入的镜像名是不存在的，或者是浏览器页面缓存的镜像名实际上已经不存在。

解决方案：

清除浏览器缓存，输入/选择真正存在的镜像。

(7) 挂起 TensorFlow、Pytorch 等 AI 作业。作业状态显示挂起，但后台进程仍然运行。

原因：

作业容器启动前无法被挂起。

(8) 数值数据集列名中包含符号.的数据集，预览无法显示数据。

原因：

不支持列名中包含符号.的数据集。

解决方案：

不要在列名中包含符号。

(9) 创建一个新的开发环境，此时开发环境一直启动不了，报错：“The agent was unable to contact any other agent located on a manager node。”

原因：

发现和虚拟机虚拟网卡有关系。

解决方案：

当出现这种情况时关闭网卡的 checksums，使用以下命令执行：

```
ethtool -K <interface> tx off
```

(10) 容器桌面中，执行 mount，报错：“mount: /ubuntuxwf/: mount failed: Operation not permitted.”

原因：

容器桌面安全考虑，没有开放 privileged 权限。

解决方案：

把需要挂载的文件解压后放到容器中使用。

(11) tensorboard 有时打开后为空页面，没有加载出 tensorboard 主页面。

原因：

可能 tensorboard 需要的 json 文件没有加载完。

解决方案：

关闭页面重新打开。

(12) 开发环境容器桌面中启动 vscode 闪退。

原因：

vscode 在容器桌面中启动需要使用非隔离模式。

解决方案：

启动参数增加--no-sandbox，例如：./code --no-sandbox

(13) 使用 libaiflow.log_figure 保存并记录图表 html 时，在实验管理界面中 html 无法显示。

原因：

plotly 生成的图表 html 文件中对于 plotly 库的引用使用的是 cdn 模式，需要能够联网加载 plotly.js 能正常显示图表。

解决方案：

建议保存和记录 plotly 时使用.png, .jpeg 等图片格式替代 html 格式。

(14) 桌面容器在 firefox 中获取剪切板内容失败，导致无法粘贴从本地复制的内容

原因：

Firefox 存在剪贴板访问安全限制。

解决方案 A：

可使用门户登录页推荐的浏览器。

解决方案 B：

步骤 1、打开火狐浏览器地址栏输入：about:config

步骤 2、点击接收风险并继续，搜索：

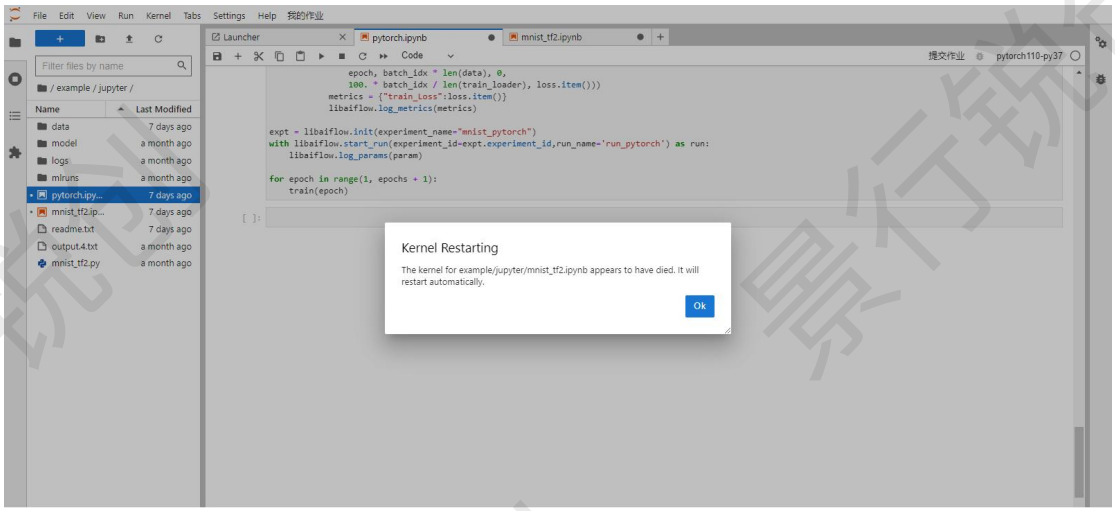
dom.events.testing.asyncClipboard、

dom.events.asyncClipboard.readText 将其值修改为 true；(注：如果 dom.events.asyncClipboard.readText 不存在可以不用修改)

步骤 3、重启浏览器即可成功。

(15) 开发环境的内存耗尽后可能会引起下面一系列问题。

1）Jupyter 中 kernel Restarting



原因：

可能是因为内存占满而导致 Kernel 重启。

解决方案：

可以尝试切换更高版本内存的硬件规格。

2）**Jupyter 报错内存已满或者“CUDA error: out of memory”**

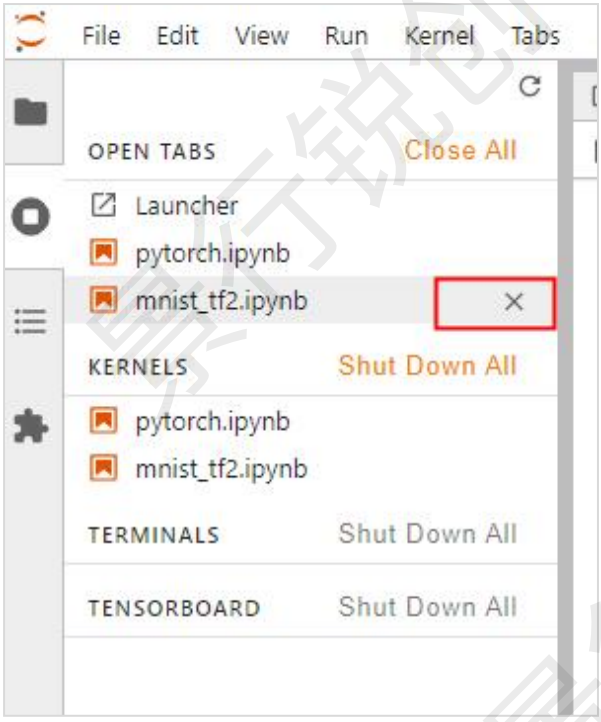
原因：

这类内存或显存满了的错误，原因是正在运行中的案例脚本太多。

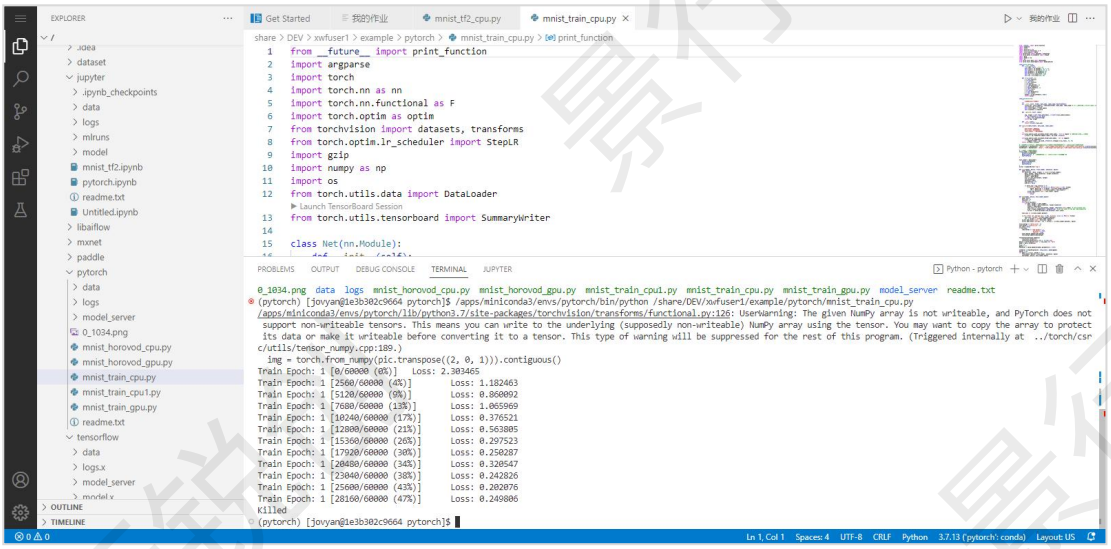
解决方案：

关掉一些正在运行的脚本即可释放内存或显存，操作方法如下：

进入如下页面，检查后台正在运行的脚本，点击“×”关闭不需要运行的脚本。



3）**问题：vscode 中运行的程序直接被 Killed。**



原因：

可能是内存占满导致。

解决方案：

检查开发环境内存是否用完，释放内存，或者可以尝试切换更高版本内存的硬件规格，或调整程序减少内存使用。

4）**问题：CentOS/Ubuntu 桌面开发环境连不上，提示错误：“会话登录超时！”。**

原因：

可能是内存占满导致。

解决方案：

检查 CentOS/Ubuntu 桌面开发环境内存是否用完，可以等待一段时间，CentOS/Ubuntu 桌面中使用的内存释放后就会恢复正常，或者可以尝试切换更高版本内存的硬件规格。

(16) MindSpore 任务结束自动标密失败。

原因：

如果 mindspore 训练代码中使用了 SummaryCollector 记录 summary 信息，保存的 summary 为只读权限，如果使用保存模型 api，保存的模型权限

为只读权限。

解决方案：

自动标密失败时，可以在训练结束后作业数据区手动执行标密操作。

(17) 在服务管理部署的 Triton Inference Server 类型服务，点击“访问服务”，跳转页面并没有显示服务信息，只显示一个空列表。

原因：

模型目录选择有误。

解决方案：

请确保选择的模型目录遵循以下规则：

每个模型目录下都至少包含一个模型版本目录和一个模型配置文件：

- 模型版本目录：包含模型文件，且必须以数字命名，作为模型版本号，数字越大版本越新。
- 模型配置文件：用于提供模型的基础信息，通常命名为 config.pbtxt。

假设模型存储目录在/examplebucket/models/triton/路径下，模型存储目录的格式如下：

```
triton
└──resnet50_pt
    ├── 1
    │   └── model.pt
    ├── 2
    │   └── model.pt
    ├── 3
    │   └── model.pt
    └── config.pbtxt
```

参考：

https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/model_repository.html

(18) 应用不同型号的 gpu 卡，提交 Pytorch Horovod 并行 gpu 作业，无法正常运行。

问题现象：

提交 Pytorch Horovod 并行 gpu 作业，日志报：“One or more tensors were submitted to be reduced, gathered or broadcasted by subset of ranks and are waiting for remainder of ranks for more than 60 seconds. This may indicate that different ranks are trying to submit different tensors or that only subset of ranks is submitting tensors, which will cause deadlock.”

问题原因：

应用不同型号的 gpu 卡，提交 Pytorch Horovod 并行 gpu 作业。

解决方案：

应用相同型号的 gpu 卡，提交 Pytorch Horovod 并行 gpu 作业，即可正常工作。

(19) 运行单机多卡程序，报：“RuntimeError: NCCL Error 2: unhandled system error (run with NCCL_DEBUG=INFO for details)”

问题原因：

容器中包含多张网卡，可能导致 NCCL 多卡通讯时失败。

解决方案：

运行程序前指定 NCCL_SOCKET_IFNAME 环境变量，指定 NCCL 使用的网卡。

开发环境中可以指定使用 eth1 网卡。

(20) tensorflow 模型转换和 tensorflow serving 模型部署，报 ModuleNotFoundError: No module named 'xxx.export_model'错误。

问题原因：

模型目录下存在非模型名称或者数字命名的文件夹，且文件夹下没有 export_model.py 文件，导致加载文件失败。

解决方案：

如果模型为权重文件，则需要在模型目录下创建模型加载文件，文件下包括 export_model.py（格式如下所示），文件加载模型文件并返回模型。如果不需要加载其他额外的文件，建议删除该文件。

```
#export_model.py
def export_model(model_file):
from FaceNet.nets.facenet import facenet
input_shape = [160, 160, 3]
model = facenet(input_shape, backbone='mobilenet', mode="predict")
model.load_weights(model_file, by_name=True)
return model
```

(21) MindSpore 任务输出中有报错信息：STDERR: setfattr: xxxxxx: Permission denied（xxxxxx 为文件路径）。

问题原因：

如果 mindspore 训练代码中使用了 SummaryCollector 则会在训练结束后把生成的相关文件修改为只读权限。

解决方案：

训练结束后在作业数据区手动执行标密操作。

(22) 提交 AI 作业（如：mindspore），发现作业正常 pull 镜像时，动态

输出尚未打印日志。

问题现象：

提交 AI 作业（如：mindspore），发现作业正常 pull 镜像时，动态输出尚未打印日志。

解决方案：

查看动态日志，通过浏览日志文件开启自动跟踪。

(23) 提交 AI 并行作业（如：提交 tensorflow horovod 并行作业）以及启动 AI 服务，发现动态输出尚未打印日志。

问题现象：

提交 AI 并行作业（如：提交 tensorflow horovod 并行作业）以及启动 AI 服务，发现动态输出尚未打印日志。

问题原因：

门户和调度获取日志的方式不同。

解决方案：

查看动态日志，通过浏览日志文件开启自动跟踪。

(24) 将“流体云仿真”模型实例，部署成 triton 服务失败。

问题现象：

将“流体云仿真”模型实例，部署成 triton 服务失败

问题原因：

triton 不支持流体云仿真模型中的一些算子。

解决方案：

使用 tensorflow serving 部署该模型。

(25) 节点 GPU 卡缺失，提交的 CPU 作业失败。

问题现象：

节点 GPU 人为拔掉，运行在该节点的 CPU 作业退出

问题原因：

某些系统 nvidia docker 不能自动切换为 CPU 模式。

解决方案：

修改/etc/docker/daemon.json 的 default-runtime 配置为 runc。

(26) Docker compose 类型的服务或其他服务作业，启动提示端口冲突。

问题现象：

存在 docker compose 服务时，其他服务启动日志提示端口冲突；或者启动 docker compose 服务时日志提示端口冲突。

问题原因：

存在 docker compose 服务时，docker compose 服务的端口未由调度分配，可能导致与调度分配的端口冲突；或者启动 docker compose 服务时，该服务的端口已被其他程序占用。

解决方案：

修改 docker compose 服务的占用端口，使用的端口时不要跟调度配置的端口范围重合，可以避免相互干扰。

(27) GPU 共享模式下，Tensorflow ps 和 Paddlepaddle ps 提交作业运行失败。

问题现象：

GPU 共享模式下，Tensorflow ps 和 Paddlepaddle ps 模式提交作业，

运行失败，报错：[Hint:’cudaErrorInvalidDevice’. This indicates that the device ordinal supplied by the user does not correspond to a valid CUD118[1] A device or that the actionrequested is invalid for the specified device.]

问题原因：

Tensorflow 和 paddlepaddle ps 并行运行在 GPU 共享模式下不支持。

(28) Edge 浏览器的 107 等版本，无法正常使用 vscode 插件。

问题现象：

Edge 浏览器打开 VSCode 开发环境时报错，无法正常使用 vscode 插件。

问题原因：

缺失证书 DigiCert Global Root G2.crt，导致 edge107 等版本打开 vscode 网页版插件时报错。

解决方案：

1）右键单击“DigiCert Global Root G2.crt ”证书文件，在弹出的右键菜单中，点击 “安装证书”。弹出“证书导入向导”窗口，选择“本地计算机”，点击下一步。

2）选择安装位置：选择“将所有的证书放入下列存储”，然后点击“浏览”，在列表中选择 “受信任的根证书颁发机构”，点击 “确定”。

3）完成证书安装：确认选择后，点击 “下一步”，会弹出安全性警告，提示安装证书可能会带来的风险，点击 “是”，最后点击 “完成”，完成证书导入。

4）验证证书安装：打开 edge 浏览器，在设置中搜索“证书”，在证书管理器中，展开 “受信任的根证书颁发机构”，点击 “证书”，在右侧列表中查找 “DigiCert Global Root G2”，确认其已存在。

5）重新打开 VSCode，此时 VSCode 插件应能正常使用。

(29) 使用谷歌浏览器需要导入景行应用门户证书，保证浏览器中安全访问景行应用门户，否则会导致景行人工智能/大模型部分功能不能正常打开。

问题现象：

启动 VSCode 服务，打开会出现报错信息：An unexpected error occurred ...(Error: WebSocket close with status code 1006)。

问题原因：

在谷歌浏览器中输入景行应用门户网址，此时页面显示访问不安全，在门户中使用景行人工智能/大模型功能，可能会导致 Web 终端、VSCode、环境日志等功能无法正常使用。需要导入景行应用门户证书，保证景行应用门户安全访问，景行人工智能/大模型功能才能正常使用。

解决方案：

1) 将${APPFORM_TOP}/traefik/cert 目录下的 ca.crt 文件复制到 windows 桌面。

2) 右键单击 ca.crt 证书文件，弹出 “证书导入向导” 窗口，点击 “下一步”。

3) 选择安装位置：选择 “将所有的证书放入下列存储”，然后点击 “浏览”，在列表中选择 “受信任的根证书颁发机构”，点击 “确定”。

4) 完成证书安装：确认选择后，点击 “下一步”，会弹出安全性警告，提示安装证书可能会带来的风险，点击 “是”，最后点击 “完成”，完成证书导入。

5) 验证证书安装：打开谷歌浏览器，在浏览器中输入景行应用门户 https 登录地址后，网址显示安全，说明证书生效。

（30） 景行应用门户中打开“AI 模型训练”页面，应用资源监控报错。

问题现象：

景行应用门户中打开“AI 模型训练”页面，应用资源监控报错：[No

found user token. Please use jlogin command create token first appform.tooltip.common.no.usable.host]

问题原因：

登录景行应用门户的用户名、密码不是真正的用户名、密码。

解决方案：

需要保证登录景行应用门户的用户名、密码，可以正常登录景行应用门户服务所在的机器。