

依瞳人工智能平台模型开发快速指引

文档版本

V1.3.0

发布日期

2020-12-01



版权所有 © 华为技术有限公司 2020。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： <https://www.huawei.com>

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 模型开发指导	1
1.1 TensorFlow 模型训练指导	1
1.2 MindSpore 模型训练指导	4
1.2.1 单卡训练	4
1.2.2 多卡训练	5
2 参考	7
2.1 NPU 环境变量配置信息	7
2.2 创建开发环境	8
2.3 上传代码/数据集	11
2.3.1 Web 界面上传	11
2.3.2 SSH 方式上传	13
2.4 模型开发环境选择	15
2.4.1 SSH 开发模式	15
2.4.2 Jupyter Notebook 开发模式	15
2.4.3 VS Code 开发模式	16

1 模型开发指导

本章节介绍使用依瞳人工智能平台进行 TensorFlow、MindSpore 模型训练，TensorFlow 框架版本为 1.15，MindSpore 框架版本为 1.0.1，模型示例代码可从 ModelZoo 上直接下载（<https://www.huaweicloud.com/ascend/resources/modelzoo>）。

1.1 TensorFlow 模型训练指导

1.2 MindSpore 模型训练指导

1.1 TensorFlow 模型训练指导

本教程以一个已迁移好的适配 TensorFlow 框架的 Resnet50 模型代码为例，介绍如何进行模型训练。

步骤 1 从 ModelZoo 上下载 [ModelZoo_Resnet50_HC.zip](#)。创建开发环境，请参考 2.2 创建开发环境。

步骤 2 待任务运行后，将代码上传至开发环境中，请参考 2.3 上传代码/数据集。

步骤 3 打开 Jupyter，请参考 2.4.2 Jupyter Notebook 开发模式，启动一个 Terminal，解压文件进入代码路径。

```
cd ~/code
unzip ModelZoo_Resnet50_HC.zip
```

步骤 4 编辑 res50_256bs_1p.py 配置文件，修改数据集路径(data_url)，如图 1-1。

```
vim ModelZoo_Resnet50_HC/00-access/src/configs/res50_256bs_1p.py
```

图1-1

图1-2

图1-3 修改数据集路径(/home/{username}/dataset/tiny_imagenet)



```

Terminal 2
import tensorflow as tf

import os
log_dir = './results/'+os.path.basename(__file__).split('.')[0]

#256
config = {
    # ===== for testing =====
    'accelerator': '1980', # 'gpu', '1980'
    'shuffle_enable': 'yes',
    'shuffle_buffer_size': 10000,
    'rank_size': 1,
    'shard': False,

    # ===== basic config ===== #
    'mode': 'train', # "train", "evaluate"
    'epochs_between_evals': 4, #use
    'stop_threshold': 80.0, #use
    'data_dir': '/opt/npu/resnet_data_new',
    'data_url': '/home/admin/dataset/tiny_imagenet',
    'data_type': 'TFRECORD',
    'model_name': 'resnet50', # 更换为上传后的数据集路径
    'num_classes': 1001,
    'num_epochs': None,
    'height': 224,
    'width': 224,
    'dtype': tf.float32,
    'data_format': 'channels_last',
    'use_nesterov': True,
    'eval_interval': 1,
    'loss_scale': 1024, #could be
    #If
    #Mus

    'use_lars': False,
    'label_smoothing': 0.1, #If
}

```

本次活动分配基本为每个用户只有一张 NPU 卡，在编辑完示例代码后直接执行如下命令即可拉起模型训练。

```

python ~/code/ModelZoo_Resnet50_HC/00-access/src/mains/res50.py --
config_file=res50_256bs_1p --max_train_steps=1000 --iterations_per_loop=100 --
debug=True --eval=False --model_dir=d_solution/ckpt${DEVICE_ID}

```

步骤 5 训练结束后，进入模型输出路径(model_dir 配置的输出路径)，即可查看对应模型文件。

```
cd d_solution
ls -alt
```

图1-4 模型输出路径

```
test@061d2995-3190-4c3f-a26e-7c4bf56da934:~/code/ModelZoo_Resnet50_HC/00-access/scripts/d_solution$ cd ~/code/ModelZoo_Resnet50_HC/00-access/scripts/d_solution$ ls
ckpt2 ckpt3
test@061d2995-3190-4c3f-a26e-7c4bf56da934:~/code/ModelZoo_Resnet50_HC/00-access/scripts/d_solution$ cd ckpt2
test@061d2995-3190-4c3f-a26e-7c4bf56da934:~/code/ModelZoo_Resnet50_HC/00-access/scripts/d_solution/ckpt2$ ls
checkpoint events.out tf_events_1606826712.1b410b09-32fa-4277-8c1f-03d9781339a-0 model.ckpt-0 data-00000-of-00002 model.ckpt-0.meta
test@061d2995-3190-4c3f-a26e-7c4bf56da934:~/code/ModelZoo_Resnet50_HC/00-access/scripts/d_solution/ckpt2$ █
```

说明

示例命令中 admin 为当前登录用户，用户可根据当前登录用户名进行调整。

- data_url：为数据集路径，data_url 填写的为用户数据集保存的路径
- config_file：为模型配置文件
- max_train_steps：为模型训练最大步长
- model_dir：为模型输出路径
- 在调试过程中，驱动错误会存储在/var/log/npulog/slog/下，通常系统报错放在 host-0 下，对应卡的报错在 device-以及 device-os-*下，对应调用的是哪一张卡，就去哪一个目录去找对应的日志，如图 1-3 所示。（可参考[图 2-1 查看当前使用的 device 设备 id 号](#)）。
- 第一次训练执行完成后，如再次启动训练任务程序会自动加载之前的权重文件，建议把输出目录清空后再次进行训练。

图1-5 查看 npu 日志

```
admin@3d566869-7793-477a-827c-a980c43aee26:/var/log/npulog/slog$ ls -l
total 60
drwxrwxrwx 199 HwHiAiUser HwHiAiUser 12288 Dec  2 15:58 container
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 12:22 device-0
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-1
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-2
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-3
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-4
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-5
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-6
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-7
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-os-0
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 11:03 device-os-4
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Dec  1 18:03 host-0
drwxrwxrwx  2 HwHiAiUser HwHiAiUser  4096 Nov 28 10:47 slogd
admin@3d566869-7793-477a-827c-a980c43aee26:/var/log/npulog/slog$
```

1.2 MindSpore 模型训练指导

📖 说明

- 流程与 1.1 TensorFlow 模型训练指导类似，此处只针对差异进行介绍。

1.2.1 单卡训练

步骤 1 MindSpore 模型训练指导同样以 Resnet 为指引，可从码云上下载对应示例源码（https://gitee.com/mindspore/mindspore/tree/master/model_zoo/official/cv/resnet）

步骤 2 下载模型示例对应的数据集到某个路径（如“~/dataset/cifar10/”），参考代码 readme 中的参数，即可启动训练，此处以 cifar10 数据集为例。

```
cd ~/code/mindspore/model_zoo/official/cv/resnet/scripts
bash run_standalone_train.sh resnet50 cifar10 ~/dataset/cifar10/

#本示例训练脚本 run_standalone_train.sh 中将标准输出存入 train/log 文件中。
#如果需要输出到终端上，需要进入 run_standalone_train.sh 中去掉>重定向命令。
```

如果需要对储存模型文件的路径进行修改，则需要进入项目 src 文件夹下，修改对应的 config.py 文件，训练结束后可到对应路径下载对应输出。

📖 说明

- 用户在调试 MindSpore 模型示例时，如果前一次训练任务异常退出，需要清理环境相关进程，再次拉起训练脚本

```
ps -aux |grep python
```

```
#本示例需要清理如下两个相关的进程
python /usr/local/lib/python3.7/site-
packages/mindspore/_extends/remote/kernel_build_server_ascend.py

python /home/admin/code/ms-resnet/train.py --net=resnet50 --dataset=cifar10 --
run_distribute=True --device_num=2 --
dataset_path=/data/dataset/storage/cifar10/cifar10-bin/

#清除训练脚本进程，执行 kill 进程 id 清除相关进程
ps -aux |grep train |awk '{print $2}' |xargs -n 1 kill -9

#清除 mindspore 进程
ps -aux |grep mindspore |awk '{print $2}' |xargs -n 1 kill -9

停止训练任务，手动执行 kill 命令关闭训练进程
```

1.2.2 多卡训练

如果要使用多卡训练，需要使用特定的 shell 脚本来拉起训练，平台会自动生成一份对应模板在(`~/.npu/$DLWS_JOB_ID/train.sh`)，模板如下

```
cd ~/.npu/$DLWS_JOB_ID/
vim train.sh

IFS=',' read -ra VISIBLE_IDS <<< "${VISIBLE_IDS}"
echo "NPU---${VISIBLE_IDS[@]}"

# setting main path
MAIN_PATH=$(dirname $(readlink -f $0))

export JOB_ID=$RANDOM
export SOC_VERSION=Ascend910
export HCCL_CONNECT_TIMEOUT=200

# local variable
export RANK_SIZE=${#VISIBLE_IDS[@]}
export RANK_TABLE_FILE=/home/$DLWS_USER_NAME/.npu/$DLWS_JOB_ID/hccl_ms.json
SAVE_PATH=$MAIN_PATH/training

# training stage

for device_phy_id in $(seq 0 ${RANK_SIZE-1})
do
    export DEVICE_ID=$device_phy_id
    export RANK_ID=$device_phy_id
    echo "start training for rank $RANK_ID, device $DEVICE_ID"
    TMP_PATH=$SAVE_PATH/D$RANK
    mkdir -p $TMP_PATH
    cd $TMP_PATH
    # {{start command}}
    cd -
done
```



```
wait
```

在这个模板中，只需要将上述 {start command} 的位置，替换为如下 python 指令（通过原启动 shell 脚本可以确定 train.py 的参数指令）。

```
python ~/code/mindspore/model_zoo/official/cv/resnet/train.py --net=resnet50 --
dataset=cifar10 --run_distribute=True --device_num=$RANK_SIZE --
dataset_path=/home/admin/dataset/cifar10/ &
```

替换完成即可执行命令启动训练

```
bash train.sh
```

说明

- 启动脚本中，建议填写 python 启动文件绝对路径
- 粘贴到 shell 模板中要保证启动指令最后有 & 符号。因为多 npu 的使用需要每张卡启动一个进程，互相同步通信，阻塞启动会造成启动第一个进程卡住等待同步，而其他卡的进程需要等待第一个进程执行完退出才会拉起，这样就会导致训练同步失败。

修改后的 train.sh 脚本如图所示：

图1-6 修改后 train.sh 脚本

```
echo "NPU--${VISIBLE_IDS[@]}"
# setting main path
MAIN_PATH=$(dirname $(readlink -f $0))
export JOB_ID=$RANDOM
export SOC_VERSION=Ascend910
export HCCL_CONNECT_TIMEOUT=200
export GLOBAL_LOG_LEVEL=3
export TF_CPP_MIN_LOG_LEVEL=3
export SLOG_PRINT_TO_STDOUT=0

# local variable
export RANK_SIZE=${#VISIBLE_IDS[@]} //训练所拉起的卡的数量
export RANK_TABLE_FILE=/home/$DLWS_USER_NAME/.npu/$DLWS_JOB_ID/hccl_ms.json
SAVE_PATH=$MAIN_PATH/training

# training stage
for device_phy_id in $(seq 0 ${RANK_SIZE-1})
do
    export DEVICE_ID=$device_phy_id
    export RANK_ID=$device_phy_id
    echo "start training for rank $RANK_ID, device $DEVICE_ID"
    TMP_PATH=$SAVE_PATH/$RANK_ID
    mkdir -p $TMP_PATH
    cd $TMP_PATH
    python $MAIN_PATH/train.py --net=resnet50 --dataset=cifar10 --run_distribute=True --device_num=$RANK_SIZE --
dataset_path=/home/admin/dataset/cifar10/ &
    cd -
done
wait
```

替换为python启动命令

2 参考

2.1 NPU 环境变量配置信息

2.2 创建开发环境

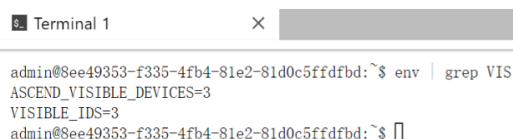
2.3 上传代码/数据集

2.4 模型开发环境选择

2.1 NPU 环境变量配置信息

- 查看当前使用的 device 设备 id 号，如图 2-1 所示 device_id 为 3。
- **env | grep VIS:** 查看当前可使用设备号，默认由平台分配

图2-1 查看当前任务使用的 device id 号



```
admin@8ee49353-f335-4fb4-81e2-81d0c5ffdfbd:~$ env | grep VIS
ASCEND_VISIBLE_DEVICES=3
VISIBLE_IDS=3
admin@8ee49353-f335-4fb4-81e2-81d0c5ffdfbd:~$
```

- 设置 HCCL 通信时间 (“~/npu/\$DLWS_JOB_ID/train.sh” 中配置)。

```
export HCCL_CONNECT_TIMEOUT=200
```

- NPU 驱动日志输出路径为 /var/log/npulog/slog,

host-0 代表该节点所有 NPU 的驱动报错日志;

device-n 代表第 n 张 NPU 卡的驱动报错日志;

device-os-0 代表第 0-3 张 NPU 卡的驱动报错日志;

device-os-4 代表第 4-7 张 NPU 卡的驱动报错日志。

- NPU 驱动日志打印到标准输出 (“~/npu/\$DLWS_JOB_ID/train.sh” 中配置)。

```
export SLOG_PRINT_TO_STDOUT=1
#设置此变量会默认以 debug 模式输出日志，NPU 驱动不会再写入 log 日志到/var/log/npulog/slog
下
```

- 设置全局日志级别。
 - Tensorflow 的日志控制环境变量（“~/npu/\$DLWS_JOB_ID/train.sh”中配置）

```
export GLOBAL_LOG_LEVEL=3
export TF_CPP_MIN_LOG_LEVEL=3
export SLOG_PRINT_TO_STDOUT=0

GLOBAL_LOG_LEVEL 取值范围说明如下：
0：对应 DEBUG 级别。
1：对应 INFO 级别。
2：对应 WARNING 级别。
3：对应 ERROR 级别。
4：对应 NULL 级别。
```

- Mindspore 的日志控制环境变量（“/var/log/npu/conf/slog/slog.conf”中配置）

```
#note, 0:debug, 1:info, 2:warning, 3:error, 4:null(no output log), default(3)
global_level=3
# Event Type Log Flag, 0:disable, 1:enable, default(1)
enableEvent=0
```

- 日志文档等级可介绍：https://support.huaweicloud.com/logr-Inference-cann/atlaslog_24_0007.html

2.2 创建开发环境

须知

由于环境资源有限，活动参与者较多，请开发者注意如下事项：

- 创建开发环境后，请尽快使用，本次使用完请“停止”释放，下次使用可再重新创建申请，建议每次使用不超过 3 小时。
- 如果出现环境多人排队情况，管理员将强制停止空闲任务、超时任务。
- 建议首次创建“开发环境”时先设置设备资源为 0，不占用 NPU 资源，先上传数据集和脚本，避免排队。

步骤 1 开发者使用活动主办方提供的账户登录平台（<http://122.224.218.50:8888/>），依次选择“代码开发 -> 创建开发环境”。如图 2-2 所示。

图2-2 代码开发



说明

代码开发主要提供模型开发所依赖的 docker 镜像环境。

步骤 2 系统进入代码开发任务配置页面，配置任务信息如图 2-3 所示，各配置项参考如表 2-1 所示。单击“立即创建”。

图2-3 创建开发环境

返回代码开发

* 开发环境名称:

tf_Resnet50

开发环境描述:

tf_Resnet50

* 代码存储路径:

/home/admin/ code/resnet50/

选择引擎来源:

☒ 预置引擎 ☐ 已保存引擎 ☐ 使用自定义引擎

* 引擎类型:

tensorflow tensorflow-npu:1.15-20.1.RC1-ar...

* 任务类型:

☒ 常规任务 ☐ 分布式任务

* 设备类型:

huawei_npu_arm64

* 设备数量:

1

立即创建

表2-1 配置项说明（创建开发环境）

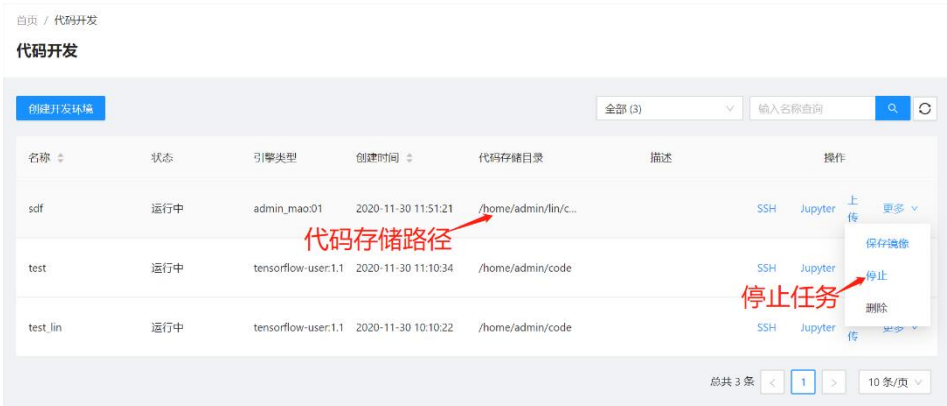
选项	配置说明
开发环境名称	任务名称
开发环境描述	选填，任务信息描述
代码存储路径	任务代码存放路径，其中“/home/用户名/”为固定路径，用户名为当前登录用户的名称，如图示例为 admin 用户

选项	配置说明
选择引擎来源	选择开发镜像环境。在该示例中，选择默认的“预置引擎”即可。 <ul style="list-style-type: none">预置引擎:为平台预置的镜像（本次活动会预置相关镜像）已保存引擎:为用户保存的镜像使用自定义引擎:使用外部镜像源
引擎类型	模型训练依赖的镜像环境。 <ul style="list-style-type: none">第一个选项为深度学习框架，在该示例中，可选择“tensorflow”或“mindspore”。第二个选项为具体的镜像名称，在该示例中，可选择“tensorflow-npu:1.15-20.1.RC1-arm”或“mindspore-npu:1.0.1-20.1.RC1-arm”。
任务类型	本次训练任务的类型。选择默认的“常规任务”即可。 <ul style="list-style-type: none">常规任务：单机任务分布式任务：多节点分布式任务
设备类型	该示例中，请选择“huawei_npu_arm64”，代表昇腾 910 训练设备。
设备数量	模型训练所需的昇腾 910 数量，该示例中，建议选 1。

⚠ 注意

如暂时不使用环境资源，请将任务停止，避免资源的占用，如图 2-4 所示，可选择停止代码开发任务。任务停止后，用户数据仍然会保存在平台中，用户下次再登录平台创建代码开发任务时(需重新配置任务信息)，只需在“代码存储路径”选择之前的存储路径，即可关联此前已上传的代码及数据。

图2-4 开发环境状态



----结束

2.3 上传代码/数据集

代码开发任务创建后，等待任务调度至“运行中”表示该任务已调度至运行状态，之后是代码工程及数据集的上传。平台支持 Web 界面和 SSH 两种上传方式。

图2-5 代码开发任务列表

代码开发

创建开发环境

全部 (5)

输入名称查询

🔍

🔄

名称	状态	引擎类型	创建时间	代码存储目录	描述	操作
test1	队列中	tensorflow-npu:1.1	2020-12-01 11:33:...	/home/Turing2/te...	SSH Jupyter	上传 更多
mnist1	运行中	tensorflow-npu:1.1	2020-12-01 11:16:...	/home/Turing2/d...	SSH Jupyter	上传 更多

2.3.1 Web 界面上传

上传代码

步骤 1 用户可点击代码开发任务中的“上传”按钮，如图 2-6 所示。

图2-6 Web 界面上传

代码开发

创建开发环境

全部 (5)

输入名称查询

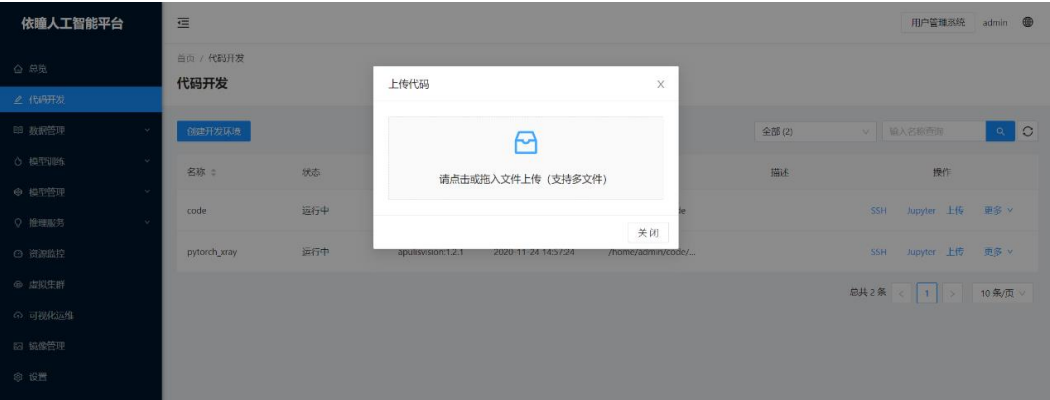
🔍

🔄

名称	状态	引擎类型	创建时间	代码存储目录	描述	操作
test1	队列中	tensorflow-npu:1.1	2020-12-01 11:33:...	/home/Turing2/te...	SSH Jupyter	<div>上传</div> <div>更多</div>
mnist1	运行中	tensorflow-npu:1.1	2020-12-01 11:16:...	/home/Turing2/d...	SSH Jupyter	<div>上传</div> <div>更多</div>

步骤 2 系统弹出如图 2-7 所示页面，将代码工程相关文件上传至开发环境中。

图2-7 上传代码



上传后用户可在此前创建代码开发环境的代码路径中查看（如 /home/admin/code/Resnet50_HC，admin 为登录用户名）。

----结束

上传数据集

对于数据集上传，用户可依次点击“数据管理 -> 数据集管理 ->新增数据集”，进行 Web 界面上传数据集，数据集支持 zip、tar、tar.gz 等几个格式文件进行上传，新增数据集配置信息可参考表 2-2 进行填写。

图2-8 数据集上传



表2-2 数据集上传配置说明

选项	配置说明
数据集名称	数据集名称
简介	数据集描述信息
是否已标注	数据是否已标注
数据权限	数据权限配置，为了保障数据安全，如通过 Web 界面上传，建议选择“私有”权限。 <ul style="list-style-type: none">私有：表示该数据集在平台上只给当前用户使用。公有：表示该数据集在平台上可公开使用。
数据源	数据的来源 <ul style="list-style-type: none">网页上传数据源：可选择本地数据集进行上传，支持 zip、tar、tar.gz 等压缩格式的数据集进行上传。使用以其它方式上传的数据源：如用户通过 ssh 的方式上传数据集，可通过该方式对数据集进行关联，方便平台其它模块使用。
上传文件	选择上传的数据集压缩文件

上传完成后，可以在“数据集管理”页面，单击数据集名称，查看数据集的存放路径。如图 2-9 所示，cyxx 为当前登录用户名。

图2-9 数据集存储路径



⚠ 注意

如需上传较大的代码工程或数据集，建议 SSH 方式进行上传。

2.3.2 SSH 方式上传

步骤 1 在代码开发任务列表中点击对应任务的“SSH”按钮，如图 2-10 所示。

图2-10 获取代码开发 docker 环境 SSH 登录信息

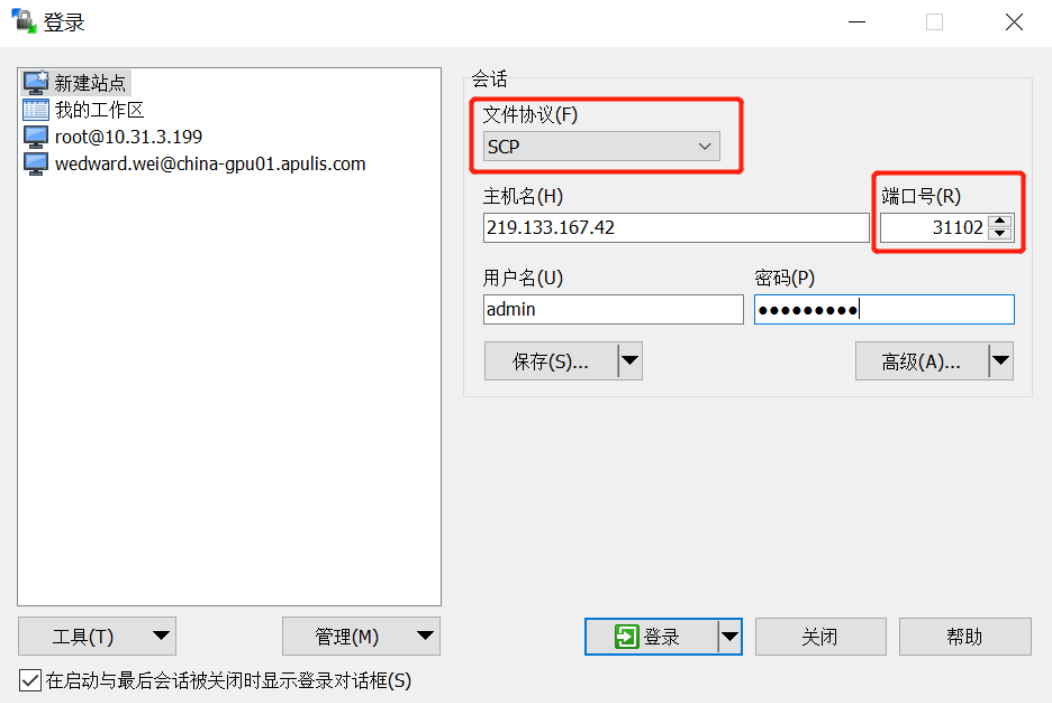


系统会弹出 SSH 登录信息，左键点击弹出信息即可复制 SSH 信息。

步骤 2 根据获取的 SSH 信息，使用 WinSCP 将代码工程或数据集上传至代码开发镜像环境内（如“/home/用户名/”路径下，用户名对应当前登录的用户名），如图 2-11 所示配置 WinSCP 登录信息。

- 文件协议：选择 SCP
- 端口：为获取 SSH 信息中对应的端口
- 用户名：为当前登录用户名
- 密码：目前固定为 tryme2017

图2-11 WinSCP 配置 SSH 登录信息



----结束

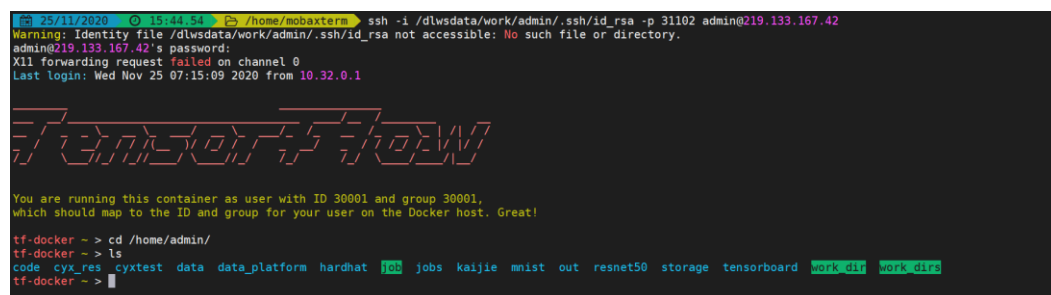
2.4 模型开发环境选择

用户将代码及数据集上传后，获取对应任务 SSH 登录信息后，平台支持用户通过 SSH 批处理模式、Jupyter Notebook/VS Code 交互式模式进行模型开发。

2.4.1 SSH 开发模式

如图 2-12 所示，输入 SSH 信息登录代码开发镜像，平台会对应当前登录用户的环境路径，用户可通过命令行方式进行代码调试、模型训练。

图2-12 SSH 登录



2.4.2 Jupyter Notebook 开发模式

用户可在代码开发列表中，单击“Jupyter”按钮进入 Notebook 环境进行模型开发。

图2-13 启动 Jupyter

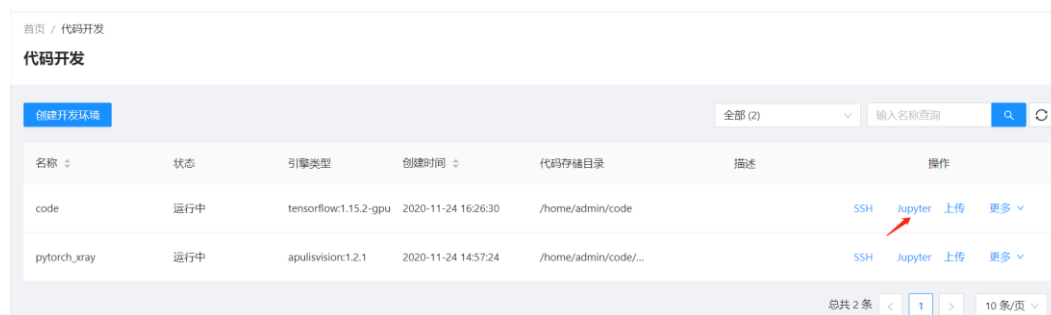
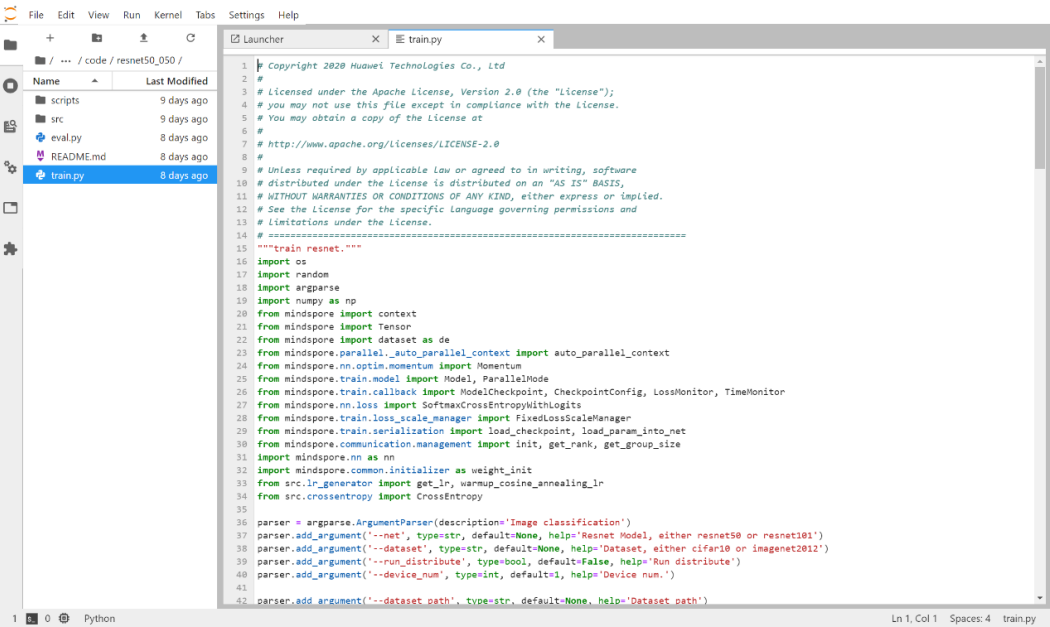


图2-14 Jupyter Notebook 交互式界面



2.4.3 VS Code 开发模式

用户获取 SSH 信息后，也可通过配置使用 VS Code 进行远程链接调试，VS Code 需安装 Remote 插件，如图 2-15~图 2-20 进行配置及登录对应环境。

图2-15 安装 Remote 插件

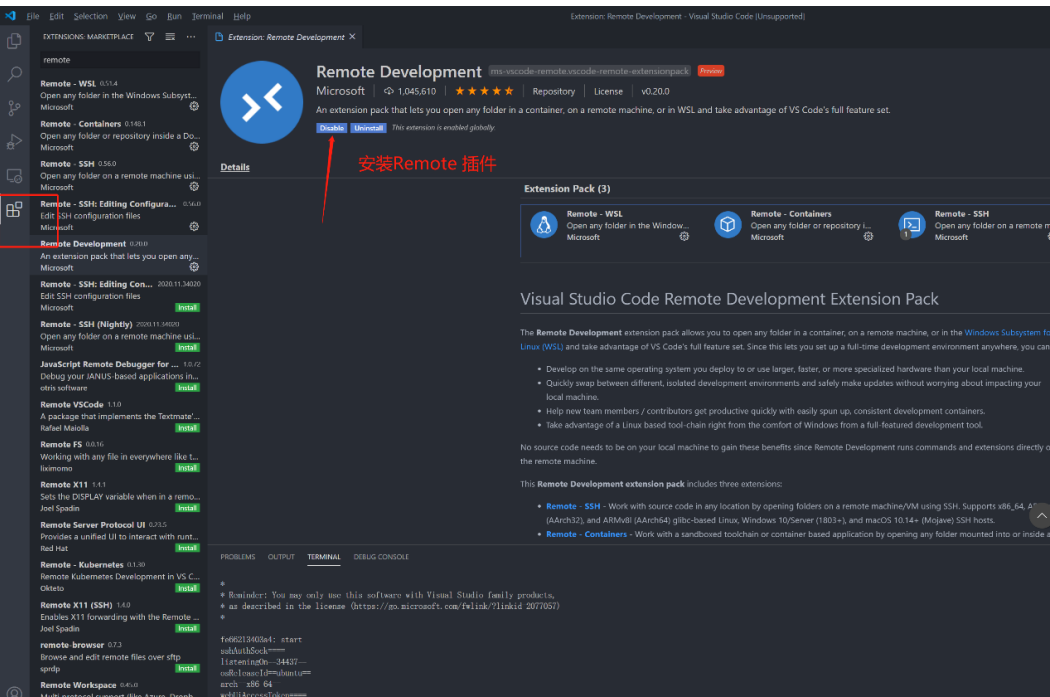


图2-16 添加 SSH 信息

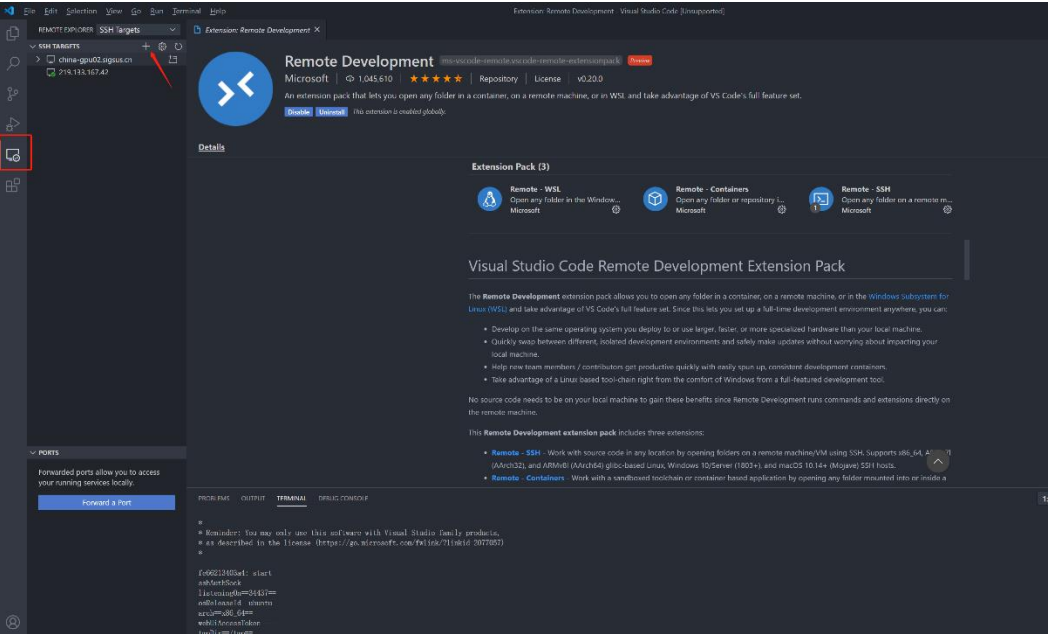


图2-17 输入 SSH 信息

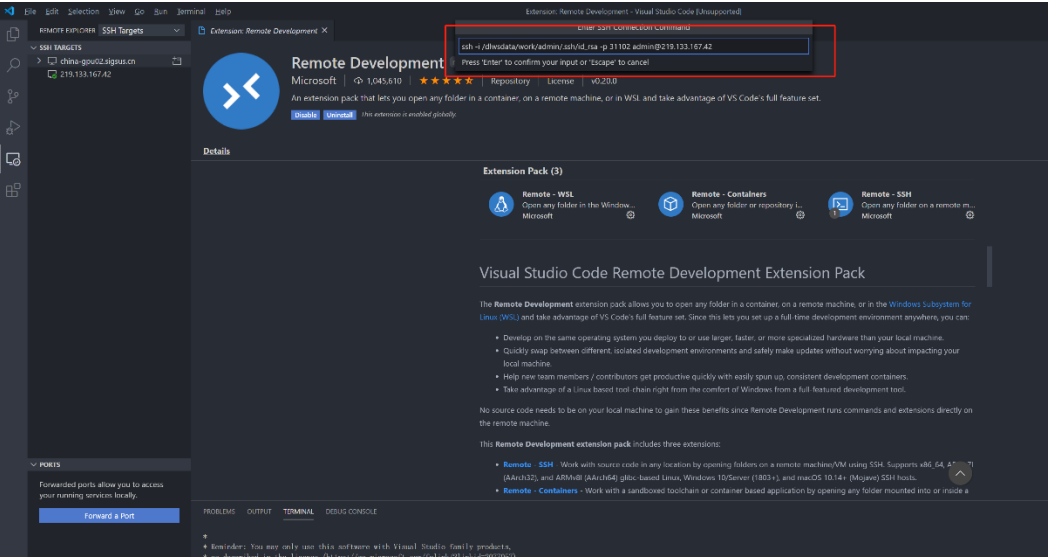


图2-18 打开 Remote SSH 环境

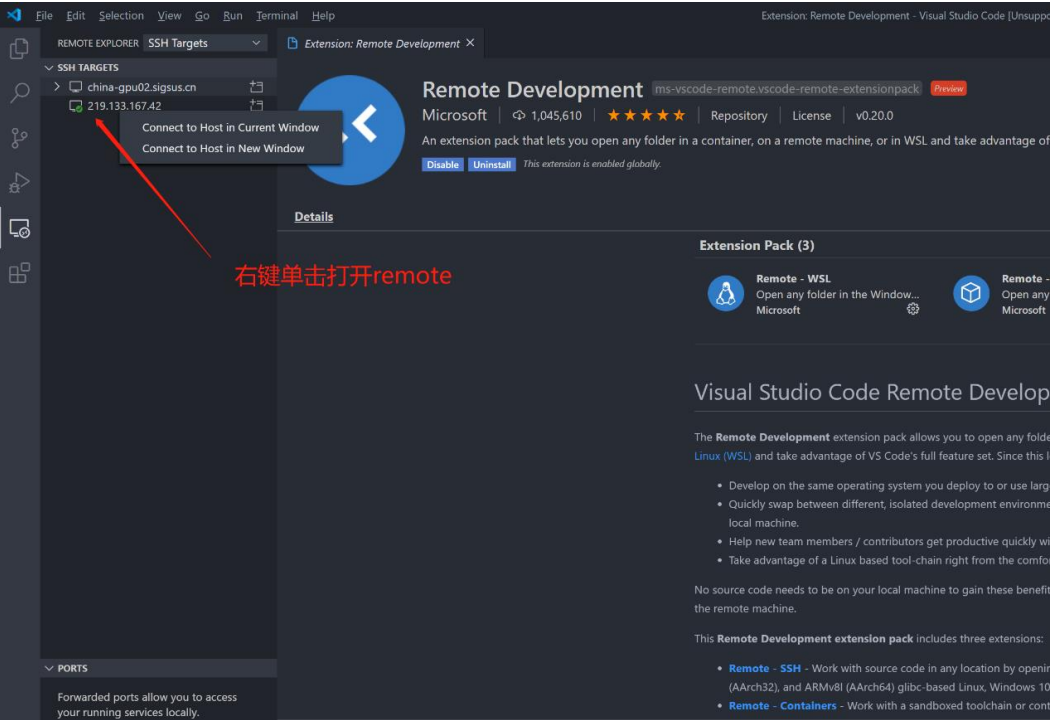


图2-19 填写对应 SSH 登录密码

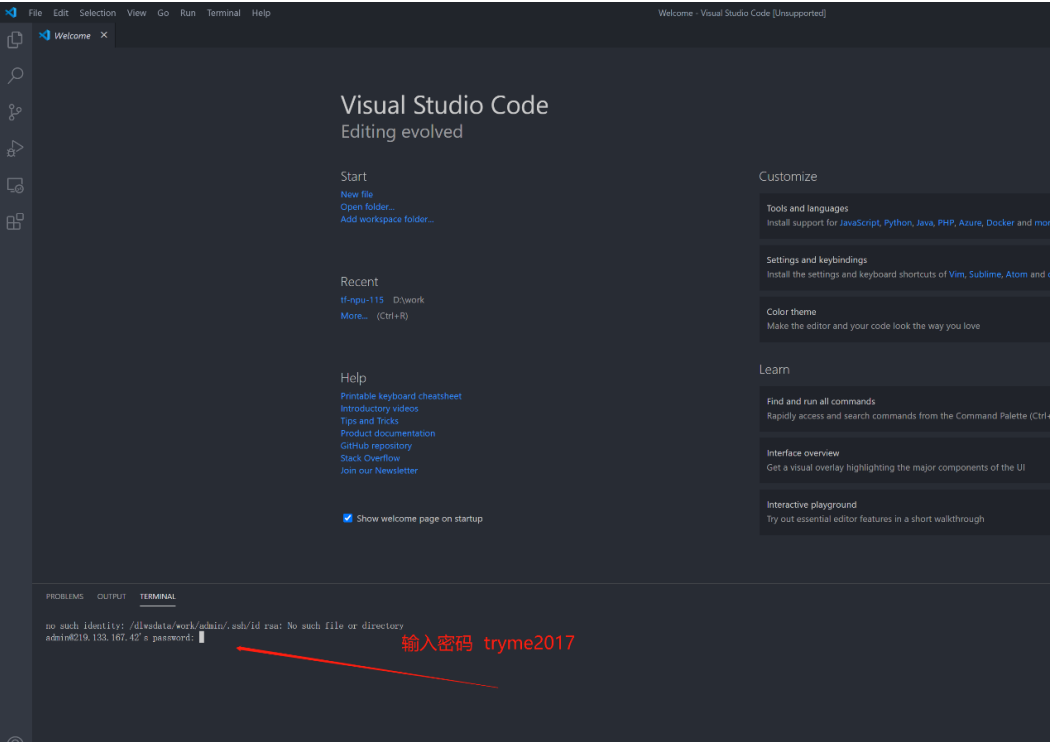


图2-20 选择开发目录

