注意：本文档的文字给出的代码是正确的，而图示的结果未更新。

# ModelArts训练文档

输入：训练数据

最终结果：onnx模型

## 创建OBS

登录华为云，网址是<https://auth.huaweicloud.com/authui/login.html?service=https://console.huaweicloud.com/modelarts/#/login>，根据下图的指引进入创建OBS界面。

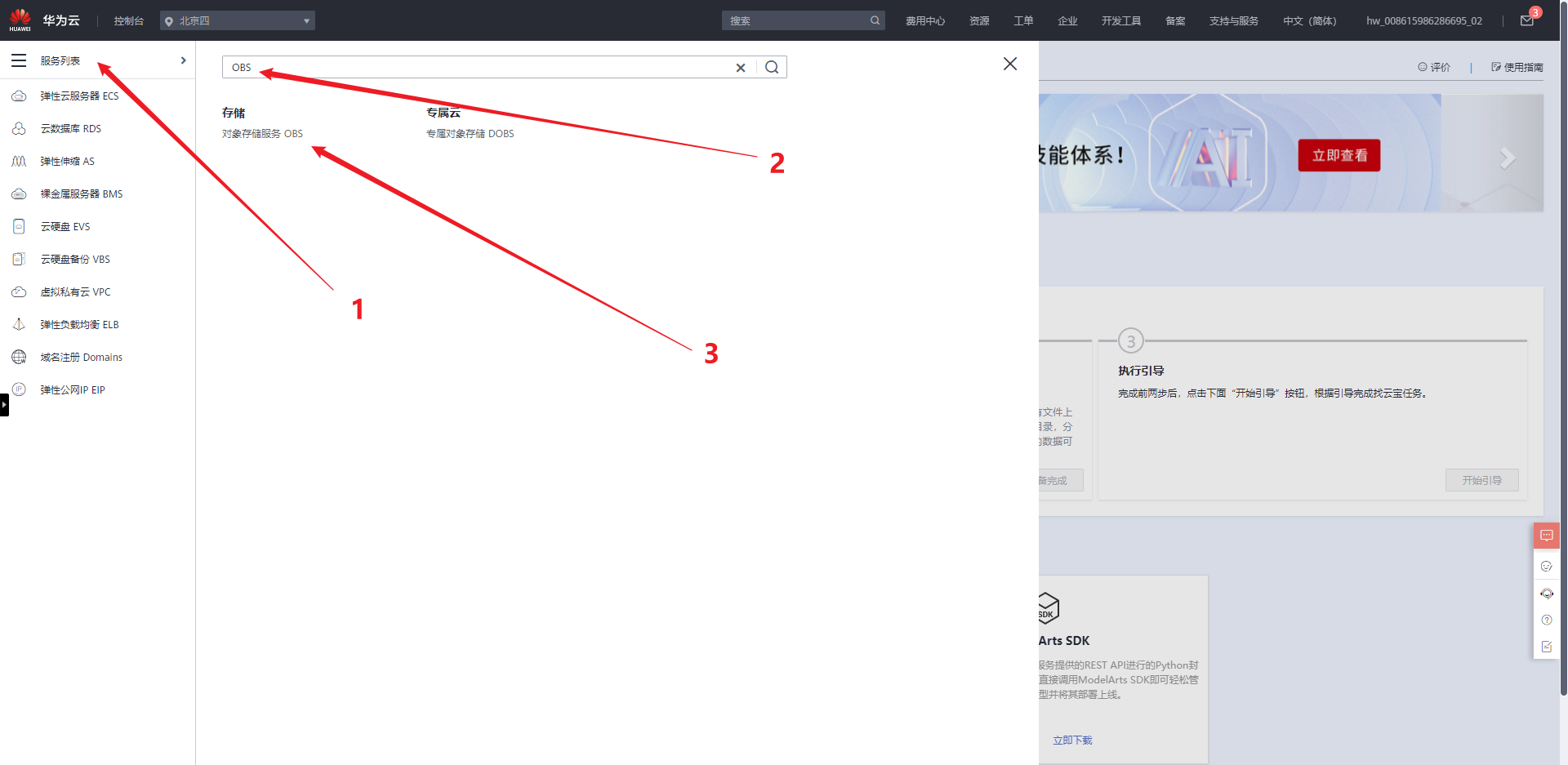


图 1 进入创建OBS界面

进入之后点击创建桶。

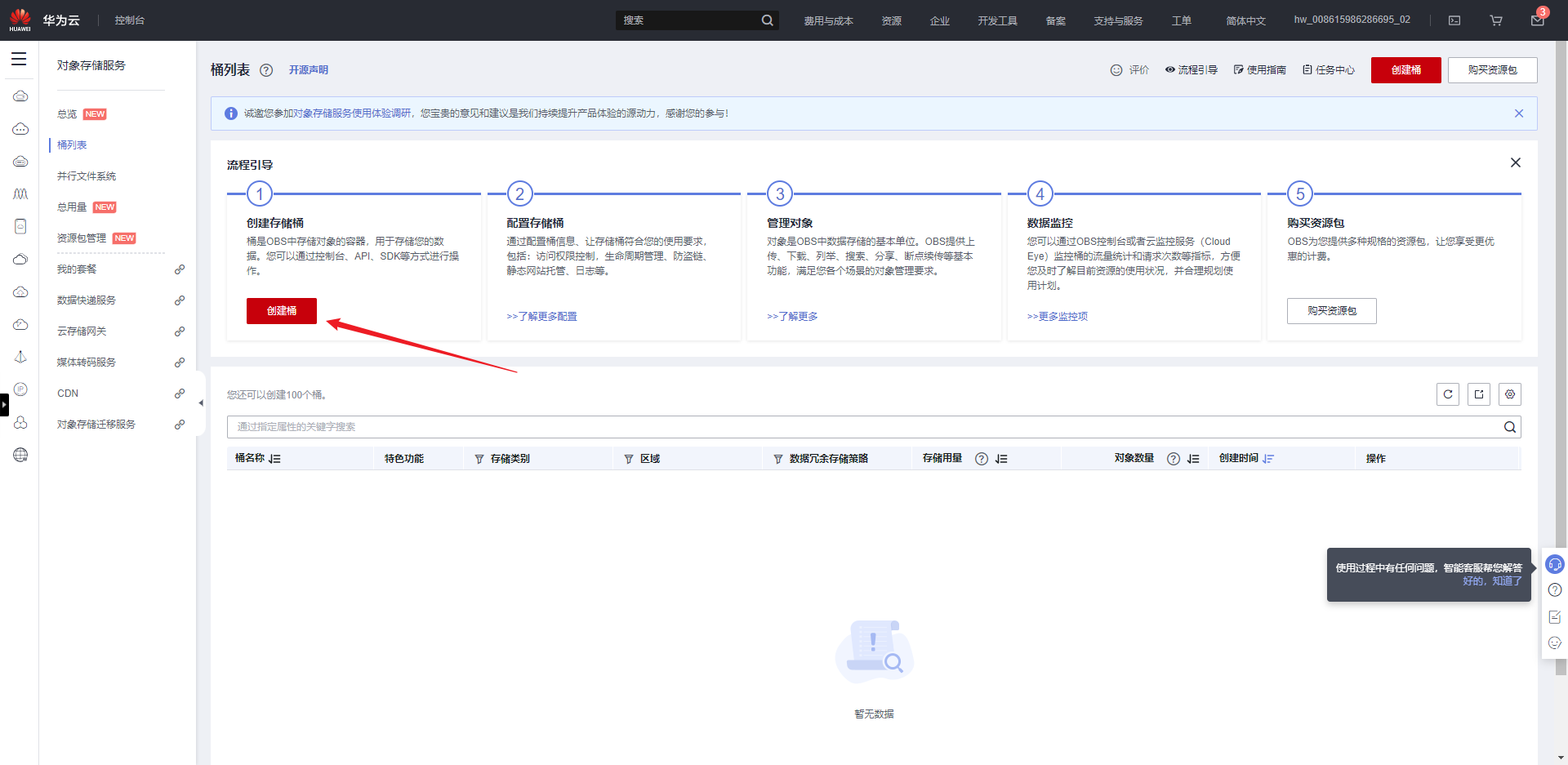


图 2 创建桶

访问<https://storage.huaweicloud.com/obs/?locale=zh-cn&region=cn-north-4#/obs/create>网址，创建OBS桶，OBS桶的配置信息如下图所示，桶名称可随意设置。在弹出的窗口点击设置即可。

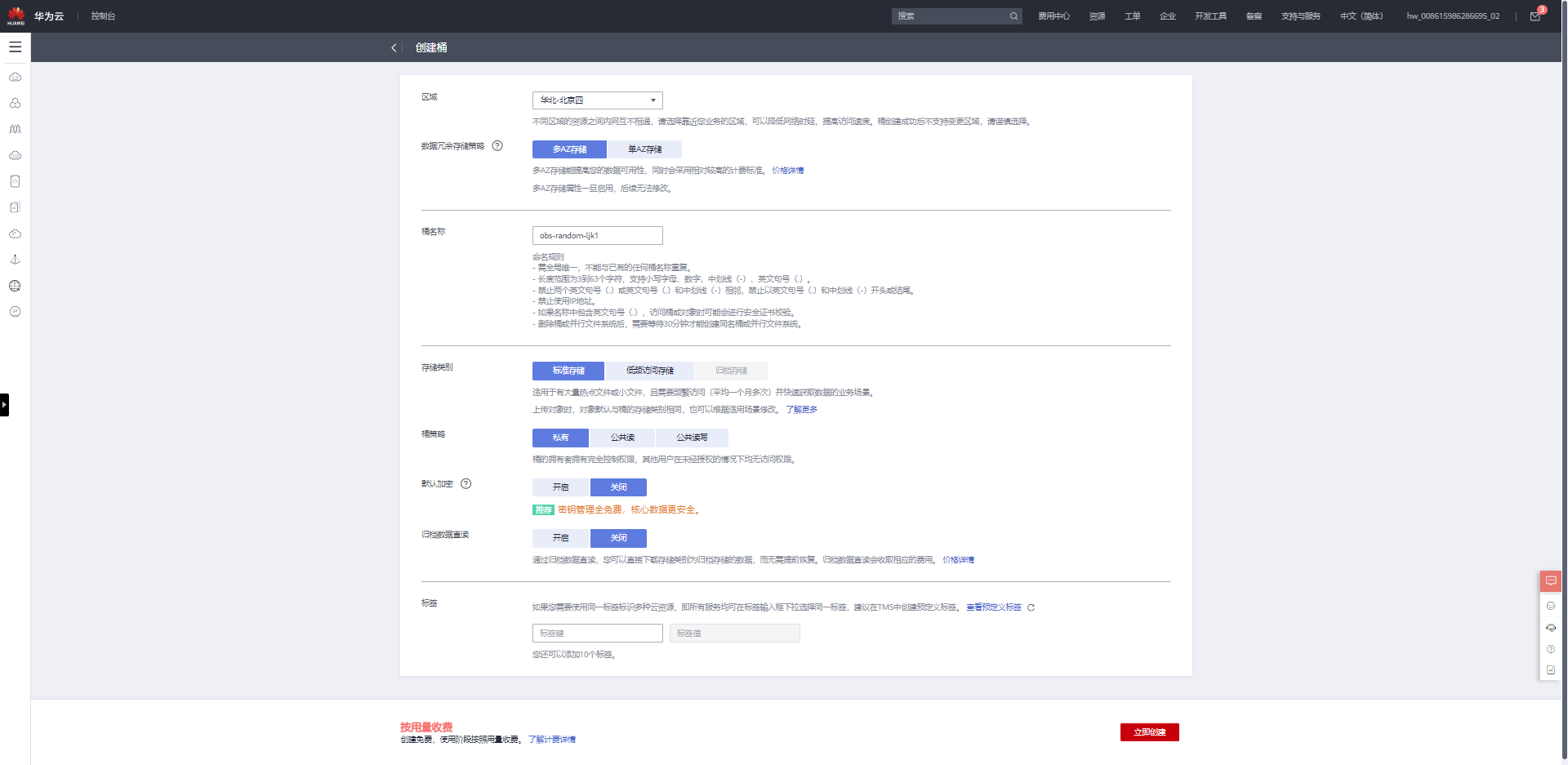


图 3 桶的配置信息

## ModelArts环境

点击网址<https://console.huaweicloud.com/modelarts/?region=cn-north-4#/notebook>，进入创建Notebook的界面，如下图所示。

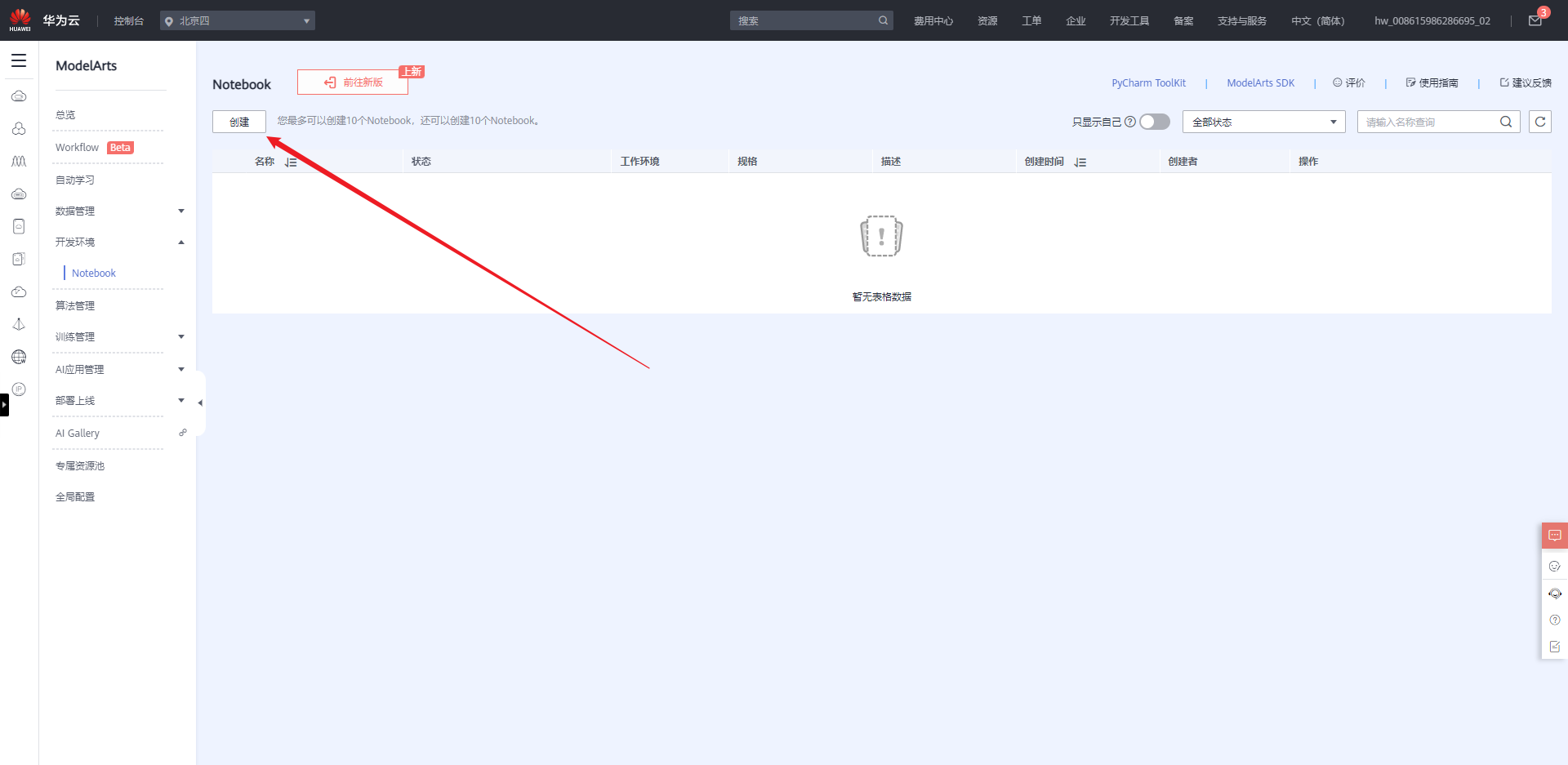


图 4 创建Notebook

Notebook的配置信息如下所示，需要选择mindspore1.7.0-py3.7-ubuntu18.04的公共镜像，然后选择存储位置。

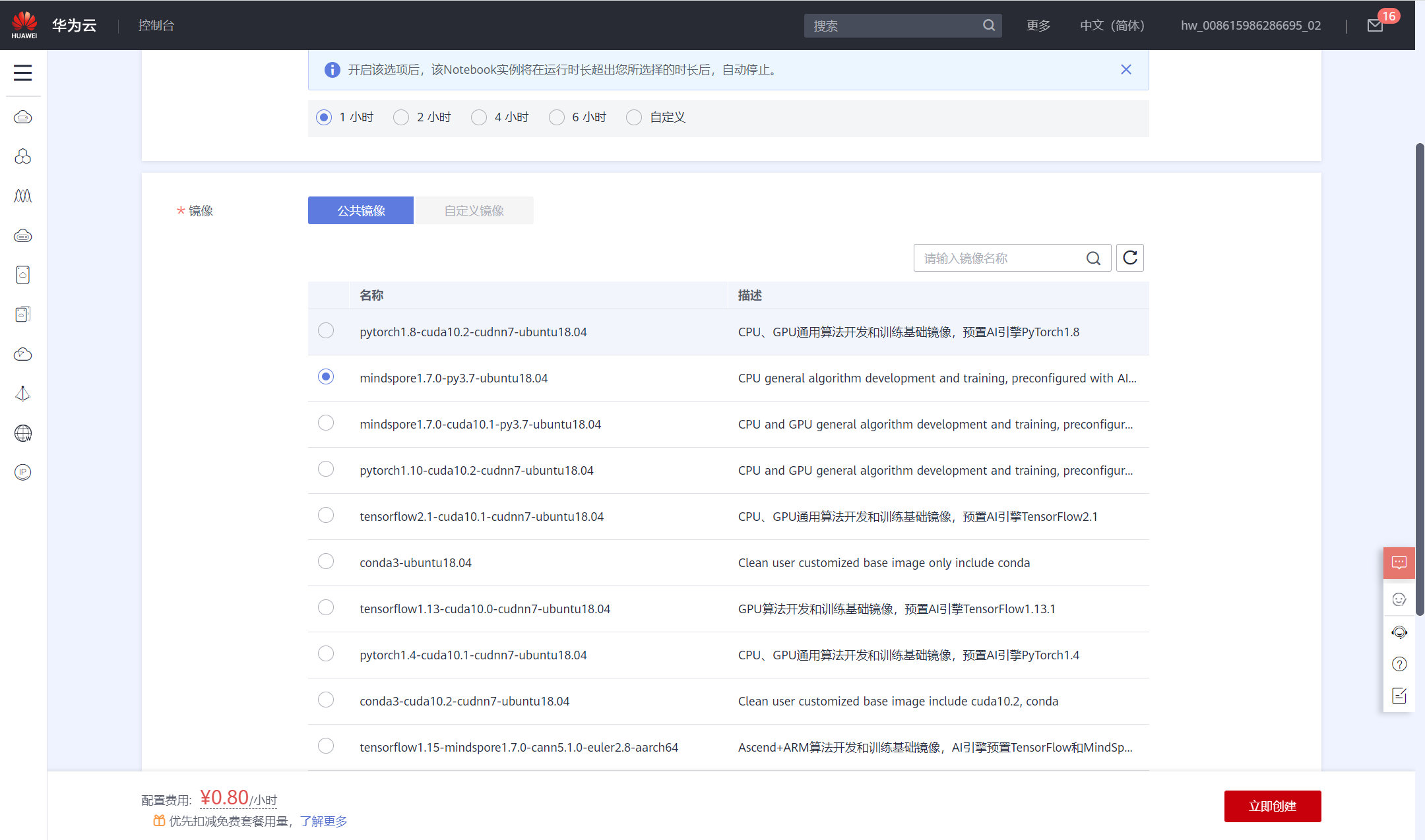


图 5 Notebook的配置信息

配置存储位置如下图所示，需要先选择OBS桶，然后创建一个文件夹。



图 6 配置存储位置

提交即创建完成（下图所示的环境未更新，注意MindSpore的版本是1.7.0）。

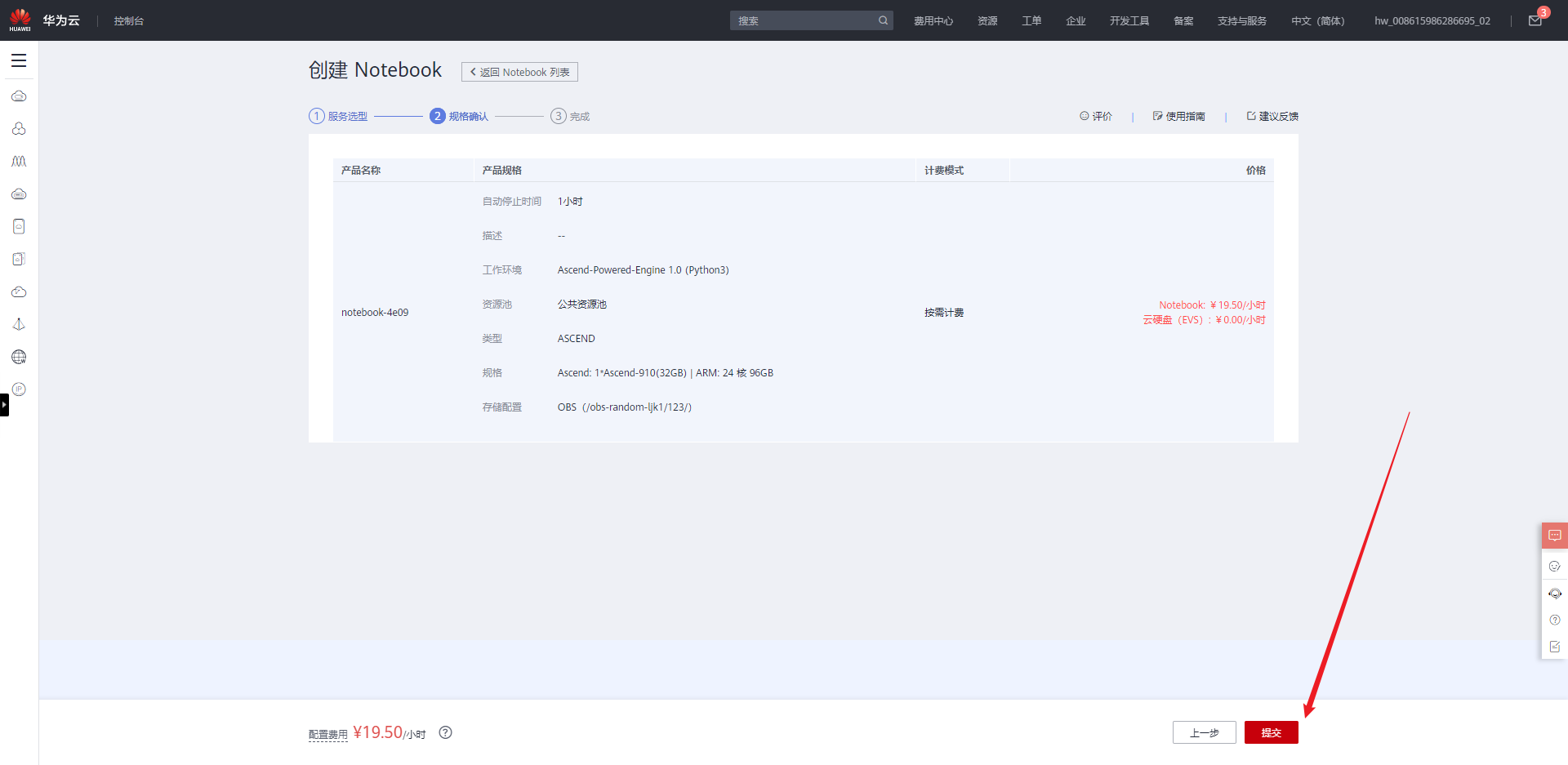


图 7 提交并创建NoteBook

成功后点击“返回NoteBook列表”。再次回到OBS界面，进入之前创建的OBS桶，比如obs-random-ljk1。

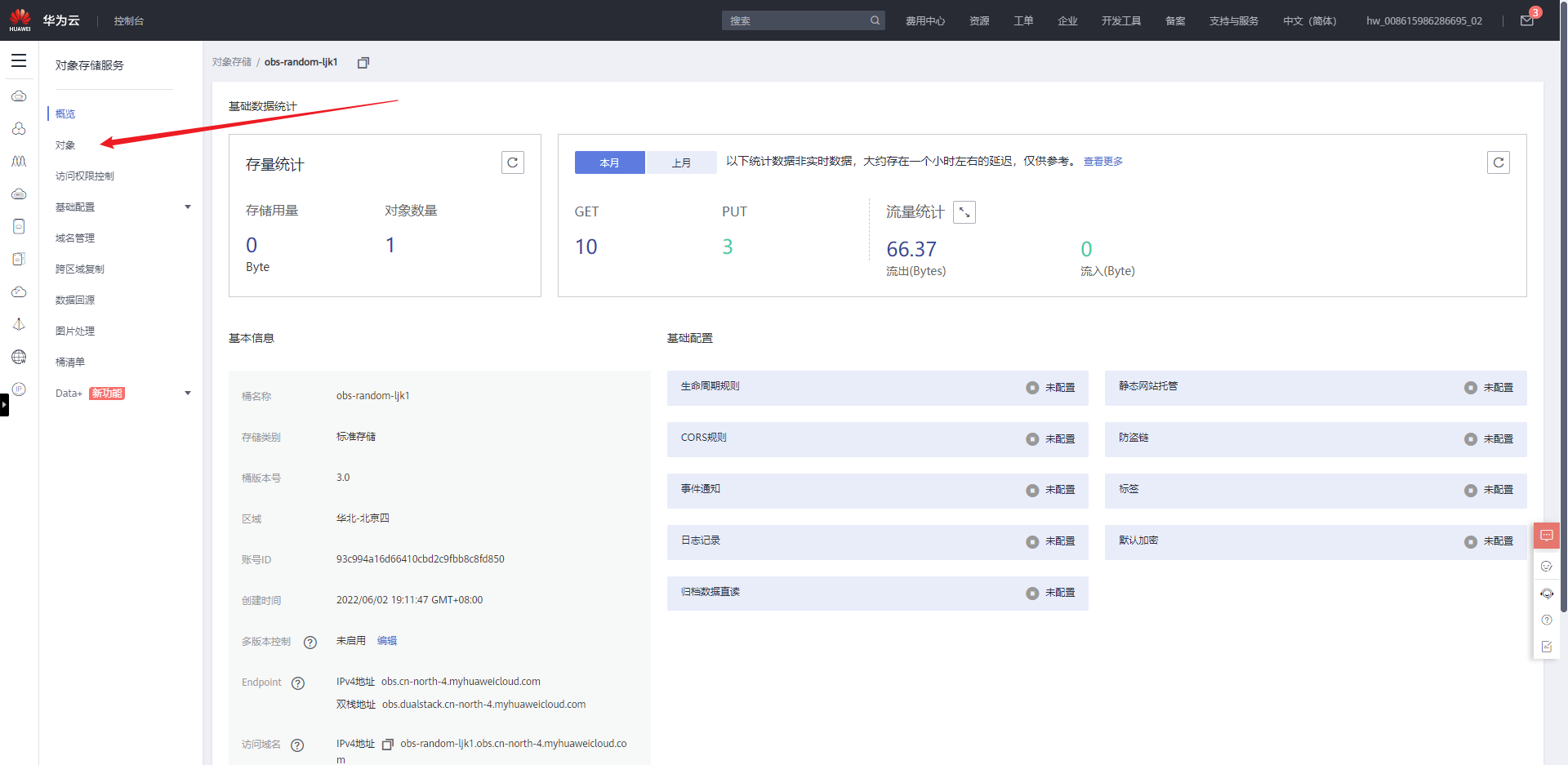


图 8进入OBS桶

上传mobilenetv2\_garbage.zip文件到OBS桶的/name/123路径下。

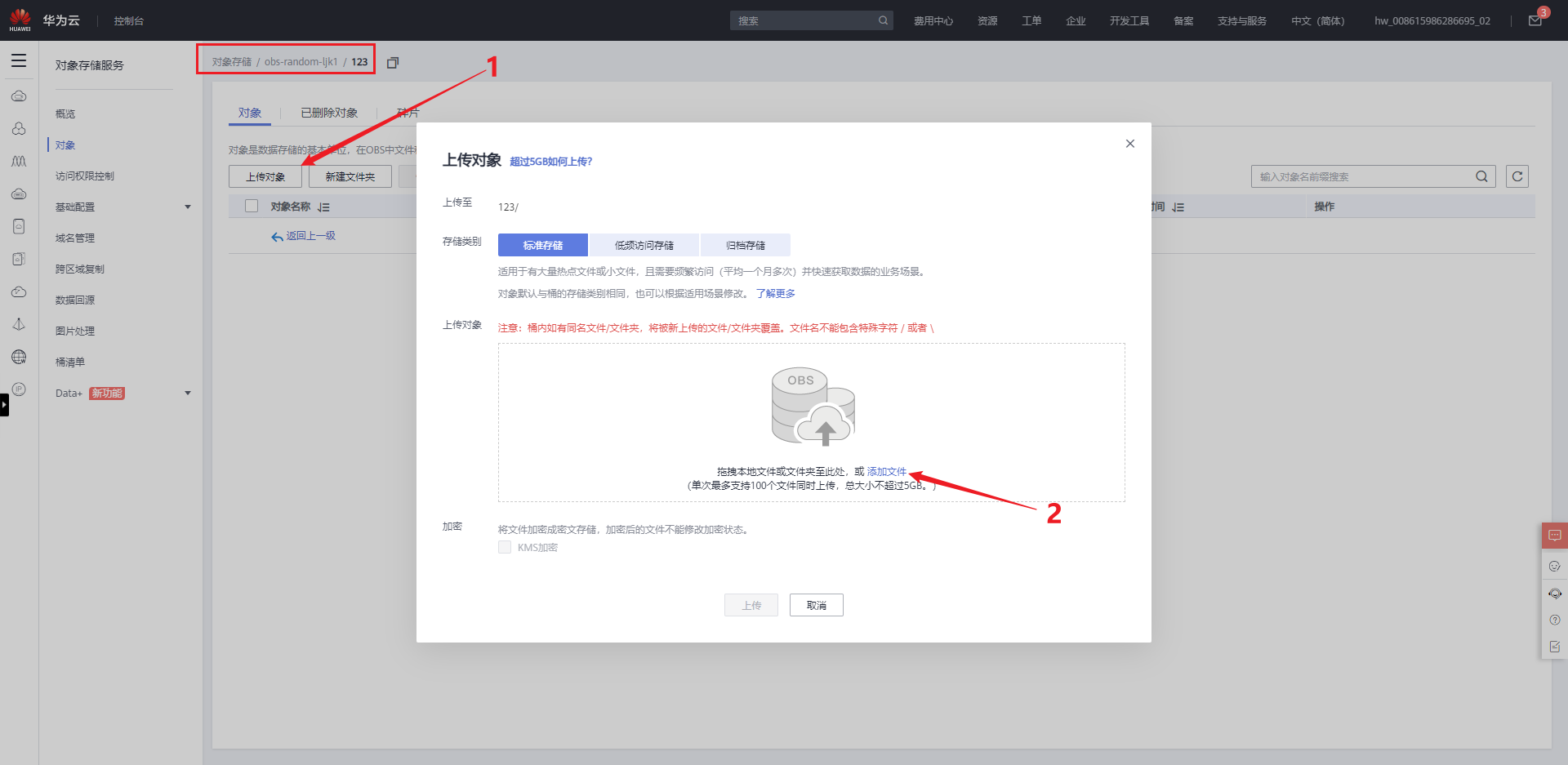


图 9 上传数据集

上传完成后，可在/obs-random-ljk1/123路径下查看到数据集。

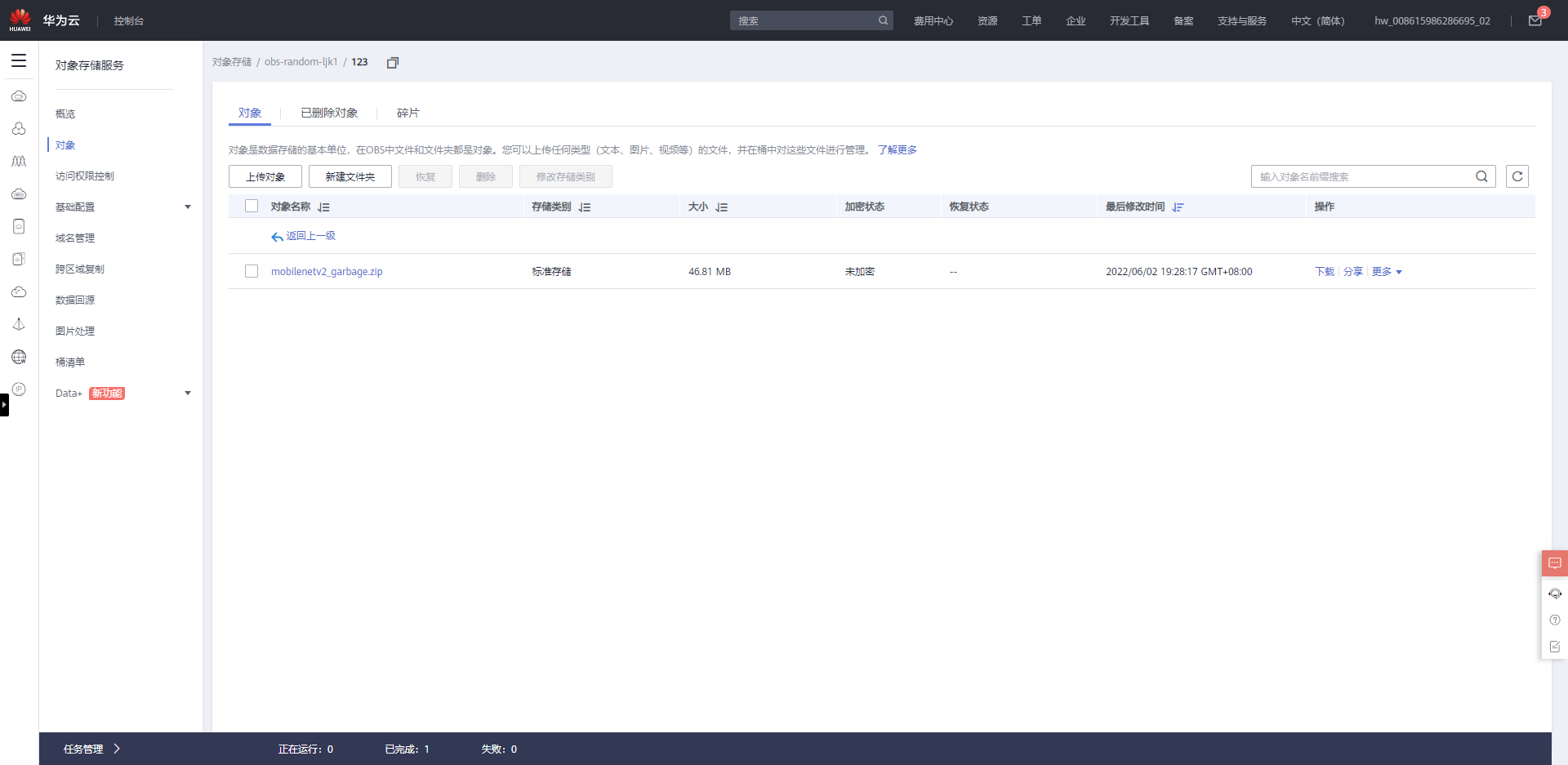


图 10上传数据集成功示意图

返回NoteBook创建列表，在刚刚创建的notebook那一行点击“打开JupyterLab”。

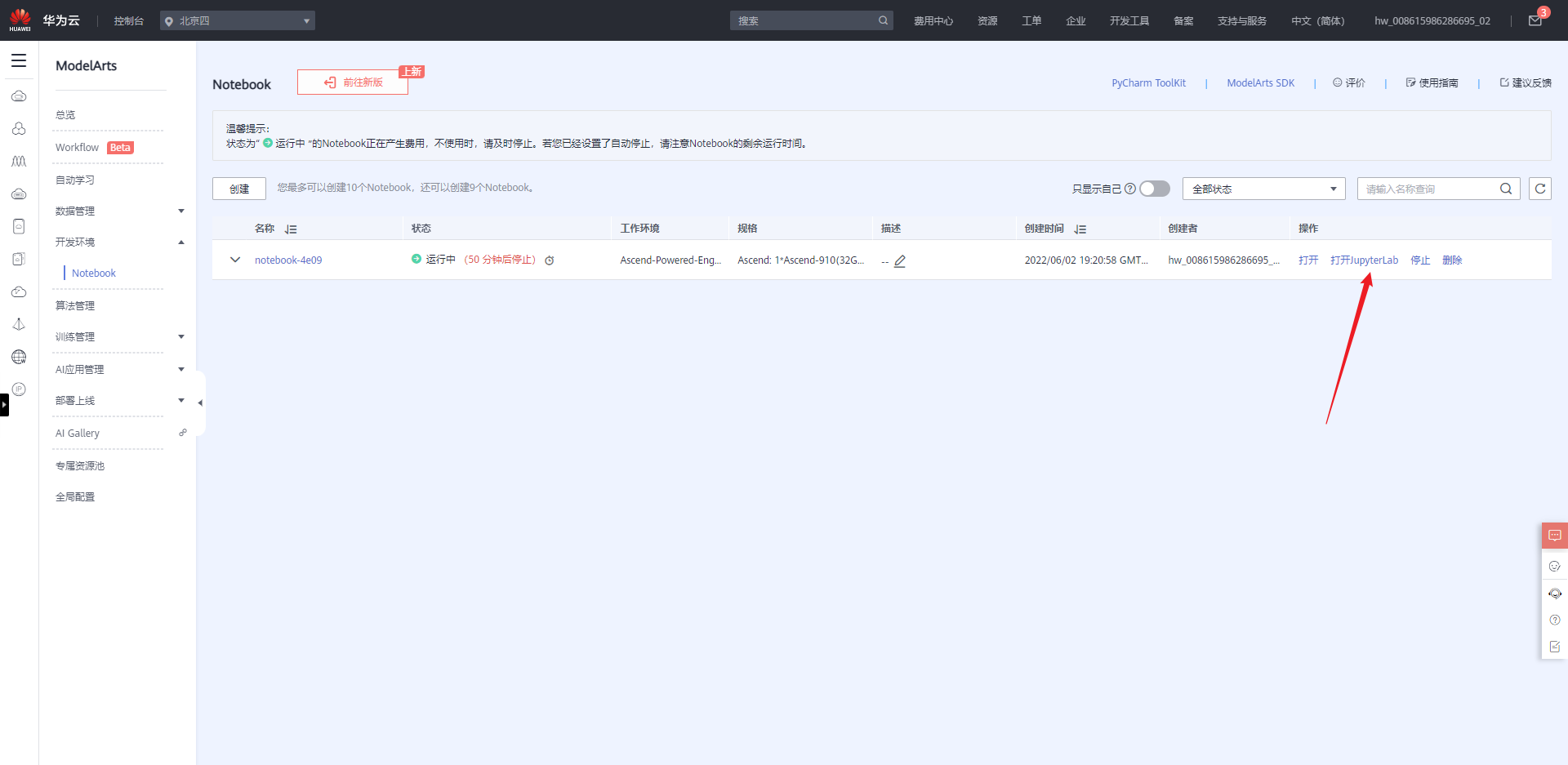


图 11 打开JupyterLab

将上传的数据集同步到训练主机上，如下图所示。

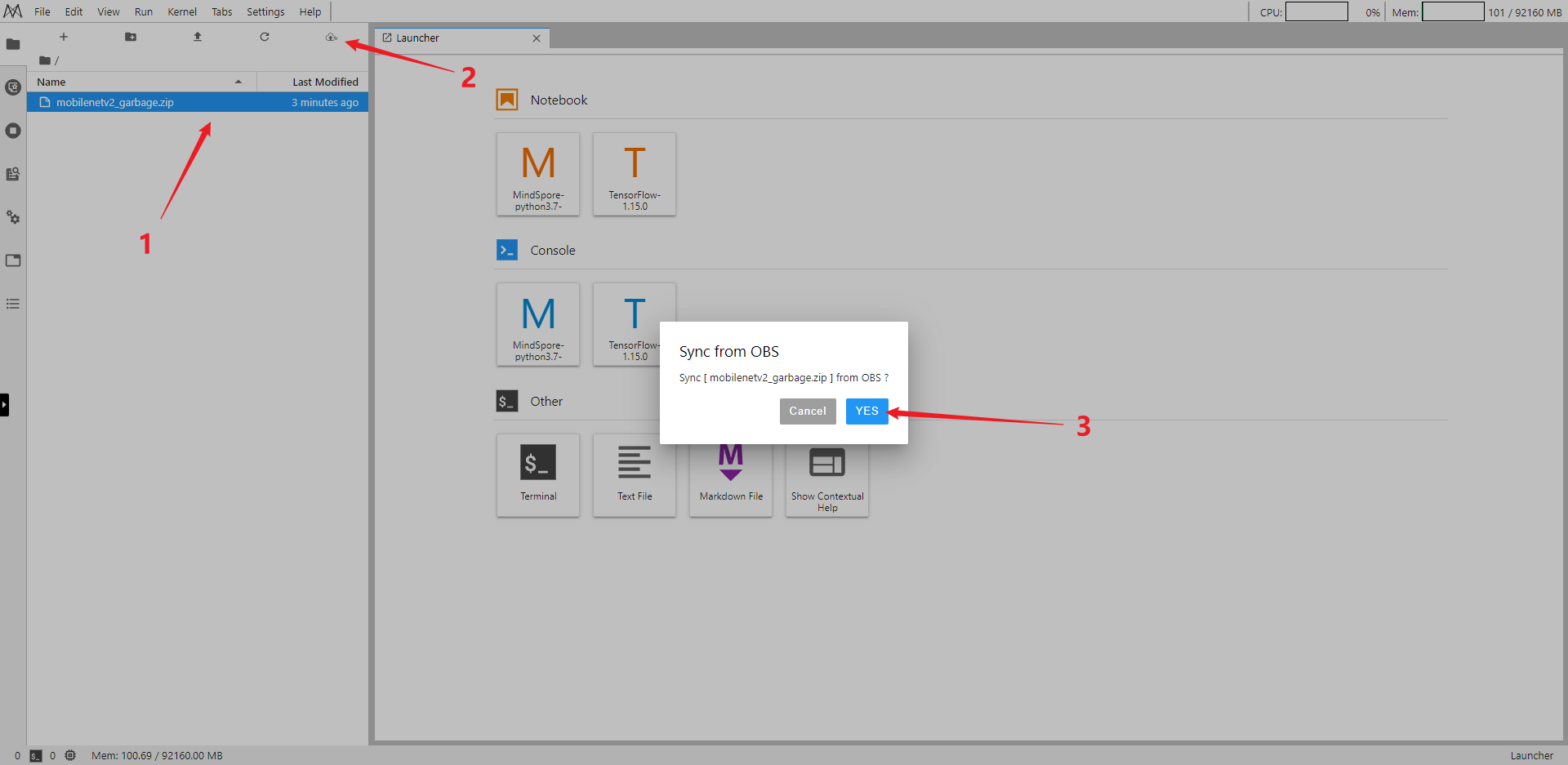


图 12 同步数据集

进入Terminal环境。在控制台输入cd work命令，切换到work目录下。再输出unzip mobilenetv2\_garbage.zip命令解压数据集。

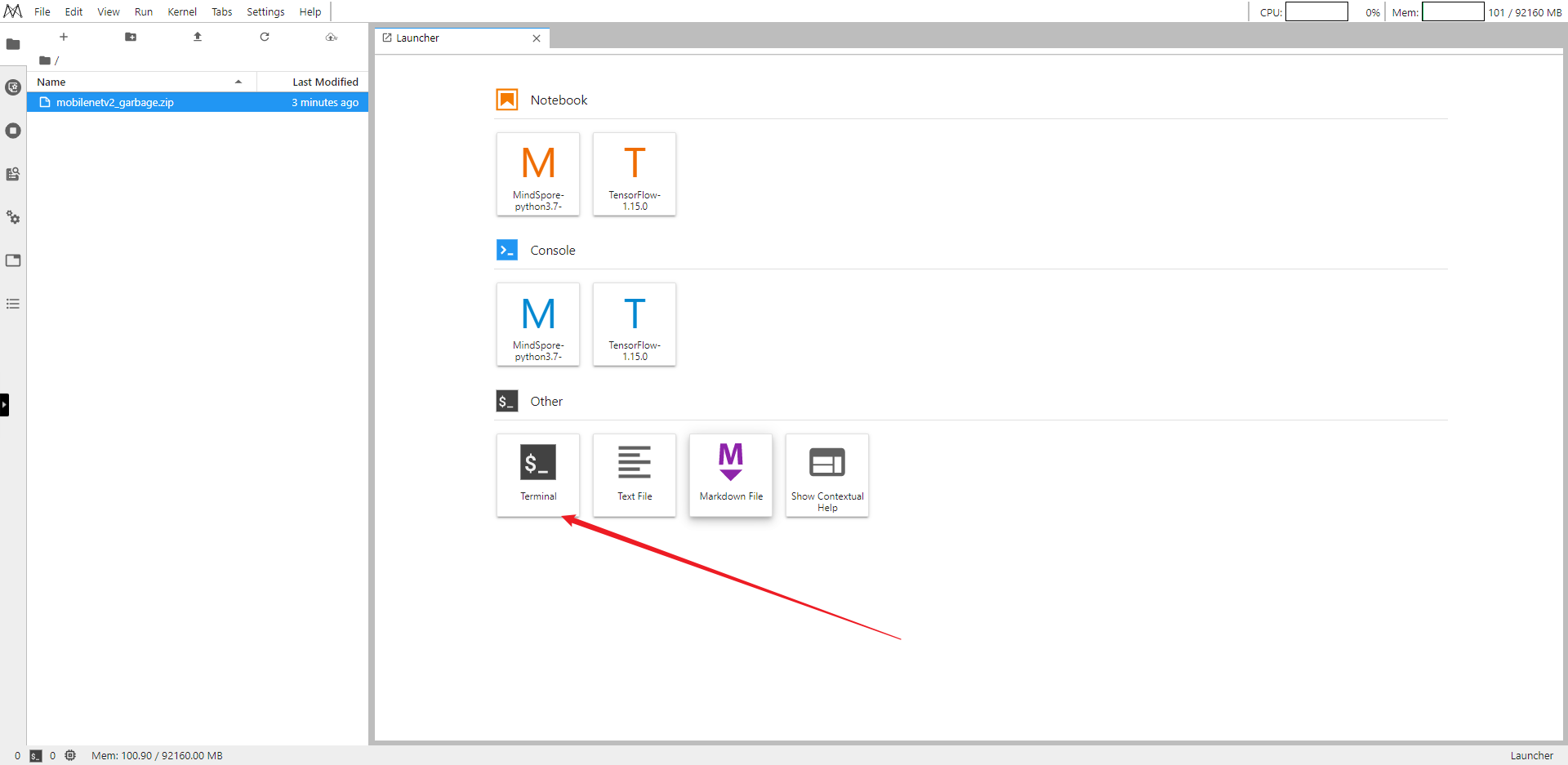


图 13 点击进入Terminal环境

解压后的结果如下图所示。

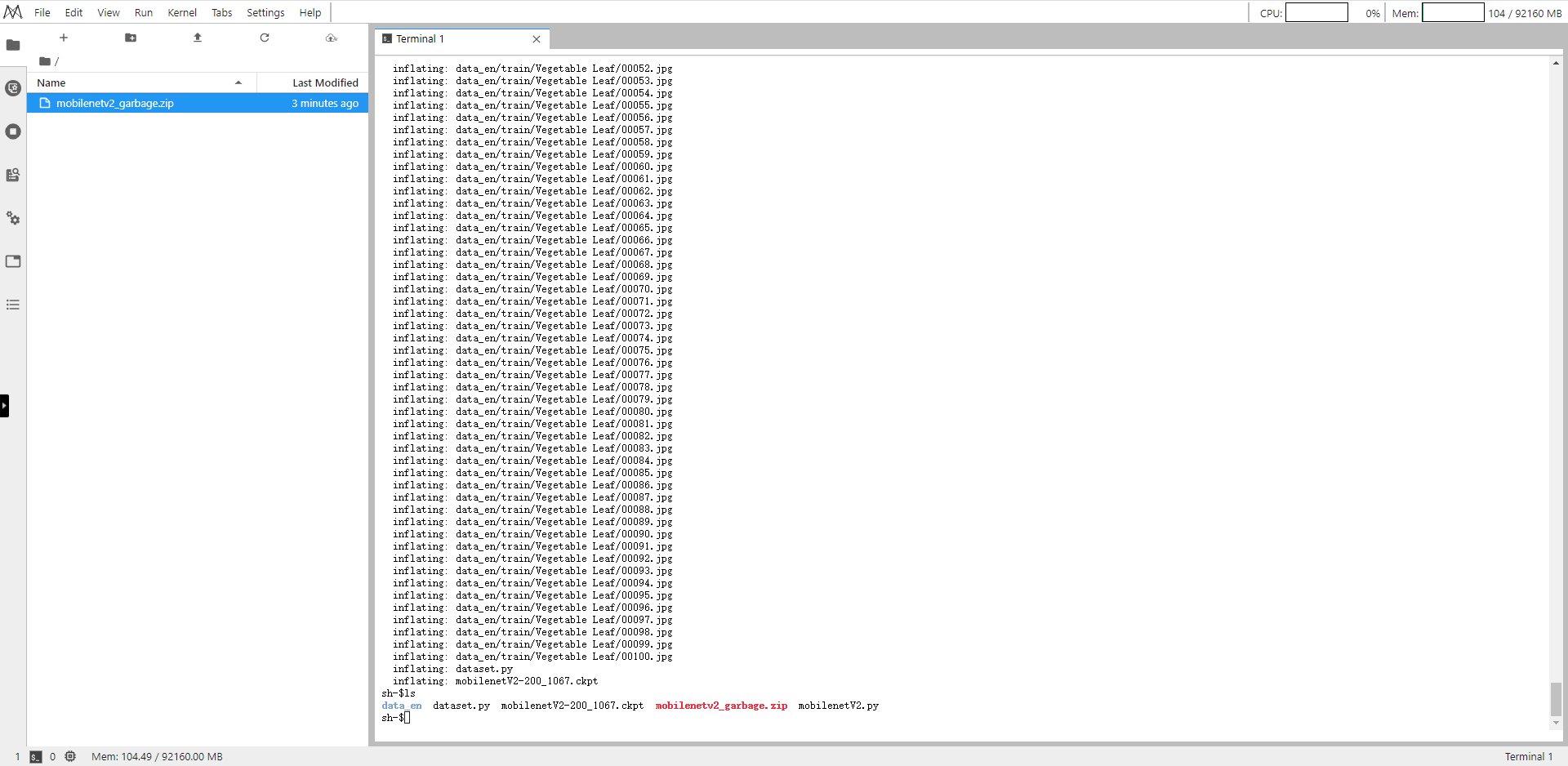


图 14数据集解压结果

创建新的Launcher，并点击MindSpore-python3.7-。

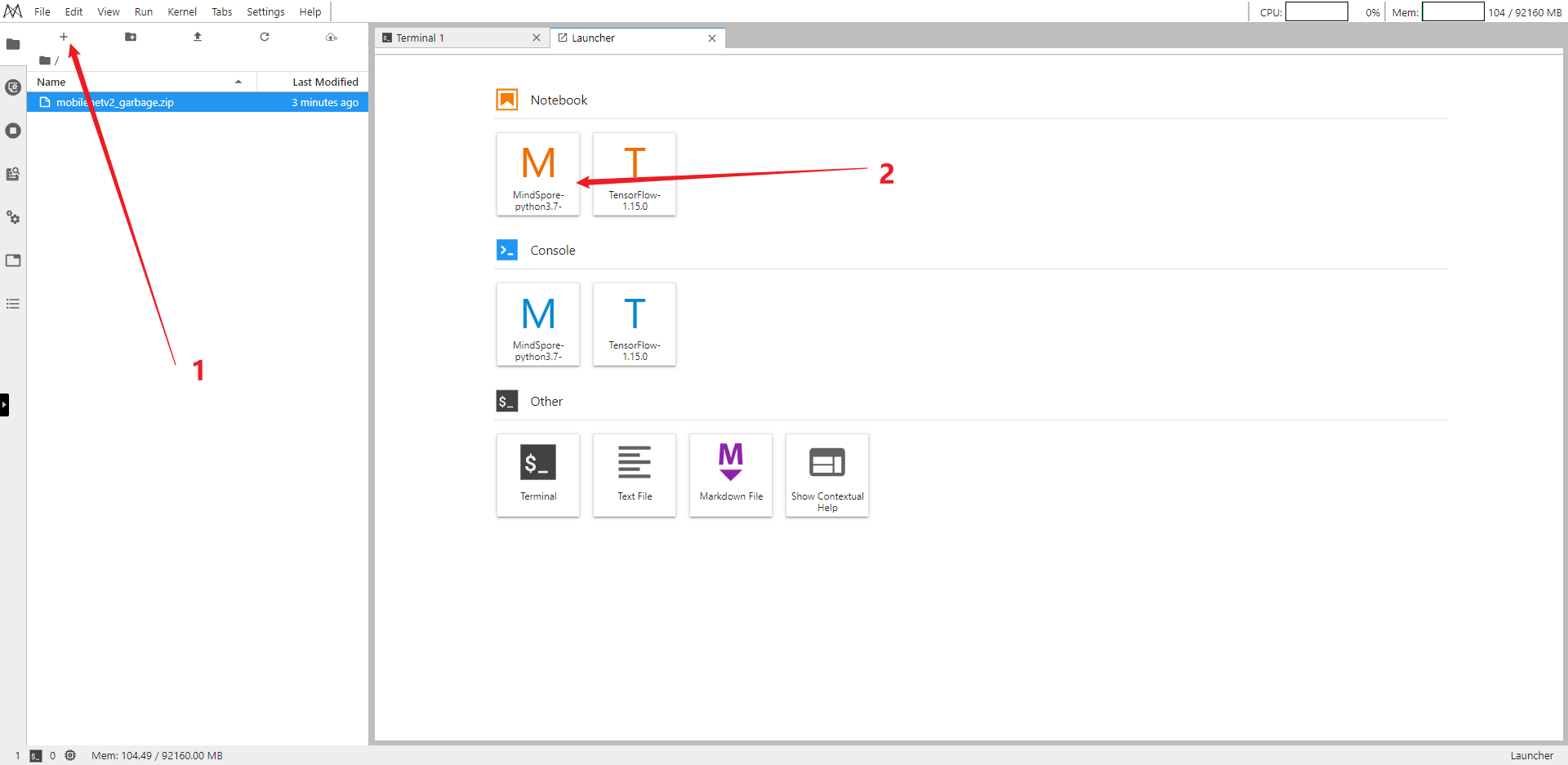


图 15 进入Pyhon编辑界面

## MobileNetV2训练

复制下面的代码到cell中，并按下alt + enter组合键，运行该cell并创建一个新的cell。

import math

import numpy as np

import os

import random

import shutil

import time

from matplotlib import pyplot as plt

from easydict import EasyDict

from PIL import Image

import mindspore as ms

from mindspore import context

from mindspore import nn

from mindspore import Tensor

from mindspore.train.model import Model

from mindspore.train.serialization import load\_checkpoint, save\_checkpoint, export

from mindspore.train.callback import Callback, LossMonitor, ModelCheckpoint, CheckpointConfig

from dataset import create\_dataset # 数据处理脚本

from mobilenetV2 import MobileNetV2Backbone, MobileNetV2Head, mobilenet\_v2 # 模型定义脚本

os.environ['GLOG\_v'] = '3' # Log level includes 3(ERROR), 2(WARNING), 1(INFO), 0(DEBUG).

# context.set\_context(mode=context.GRAPH\_MODE, device\_target="Ascend", device\_id=0) # 设置采用图模式执行，设备为Ascend#

context.set\_context(mode=context.GRAPH\_MODE, device\_target="CPU") # 设置采用图模式执行，设备为CPU#

结果如下图所示，图示的代码未更新，按照上面的来就行。

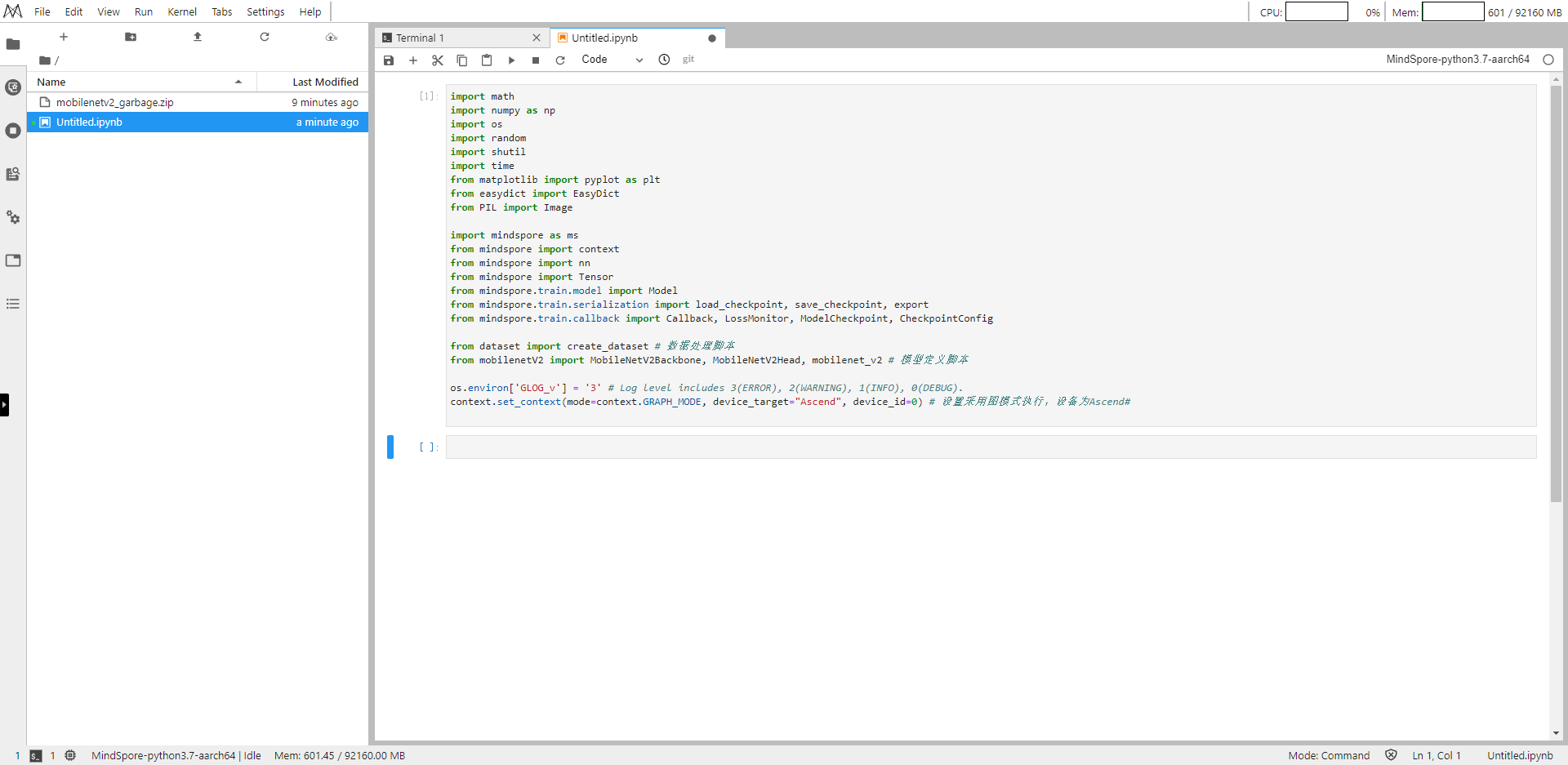


图 16 导入实验所需模块结果

复制下面的代码到刚刚创建的cell中，并按下alt+enter组合键，运行cell和创建新的cell。

# 垃圾分类数据集标签，以及用于标签映射的字典。

garbage\_classes = {

'干垃圾': ['贝壳', '打火机', '旧镜子', '扫把', '陶瓷碗', '牙刷', '一次性筷子', '脏污衣服'],

'可回收物': ['报纸', '玻璃制品', '篮球', '塑料瓶', '硬纸板', '玻璃瓶', '金属制品', '帽子', '易拉罐', '纸张'],

'湿垃圾': ['菜叶', '橙皮', '蛋壳', '香蕉皮'],

'有害垃圾': ['电池', '药片胶囊', '荧光灯', '油漆桶']

}

class\_cn = ['贝壳', '打火机', '旧镜子', '扫把', '陶瓷碗', '牙刷', '一次性筷子', '脏污衣服',

'报纸', '玻璃制品', '篮球', '塑料瓶', '硬纸板', '玻璃瓶', '金属制品', '帽子', '易拉罐', '纸张',

'菜叶', '橙皮', '蛋壳', '香蕉皮',

'电池', '药片胶囊', '荧光灯', '油漆桶']

class\_en = ['Seashell', 'Lighter','Old Mirror', 'Broom','Ceramic Bowl', 'Toothbrush','Disposable Chopsticks','Dirty Cloth',

'Newspaper', 'Glassware', 'Basketball', 'Plastic Bottle', 'Cardboard','Glass Bottle', 'Metalware', 'Hats', 'Cans', 'Paper',

'Vegetable Leaf','Orange Peel', 'Eggshell','Banana Peel',

'Battery', 'Tablet capsules','Fluorescent lamp', 'Paint bucket']

index\_en = {'Seashell': 0, 'Lighter': 1, 'Old Mirror': 2, 'Broom': 3, 'Ceramic Bowl': 4, 'Toothbrush': 5, 'Disposable Chopsticks': 6, 'Dirty Cloth': 7,

'Newspaper': 8, 'Glassware': 9, 'Basketball': 10, 'Plastic Bottle': 11, 'Cardboard': 12, 'Glass Bottle': 13, 'Metalware': 14, 'Hats': 15, 'Cans': 16, 'Paper': 17,

'Vegetable Leaf': 18, 'Orange Peel': 19, 'Eggshell': 20, 'Banana Peel': 21,

'Battery': 22, 'Tablet capsules': 23, 'Fluorescent lamp': 24, 'Paint bucket': 25}

# 训练超参

config = EasyDict({

"num\_classes": 26,

"image\_height": 224,

"image\_width": 224,

#"data\_split": [0.9, 0.1],

"backbone\_out\_channels":1280,

"batch\_size": 64,

"eval\_batch\_size": 8,

"epochs": 10,

"lr\_max": 0.05,

"momentum": 0.9,

"weight\_decay": 1e-4,

"save\_ckpt\_epochs": 1,

"save\_ckpt\_path": "./ckpt",

"dataset\_path": "./data\_en",

"class\_index": index\_en,

"pretrained\_ckpt": "./mobilenetV2-200\_1067.ckpt" # mobilenetV2-200\_1067.ckpt mobilenetv2\_ascend.ckpt

})

运行结果如下图所示。

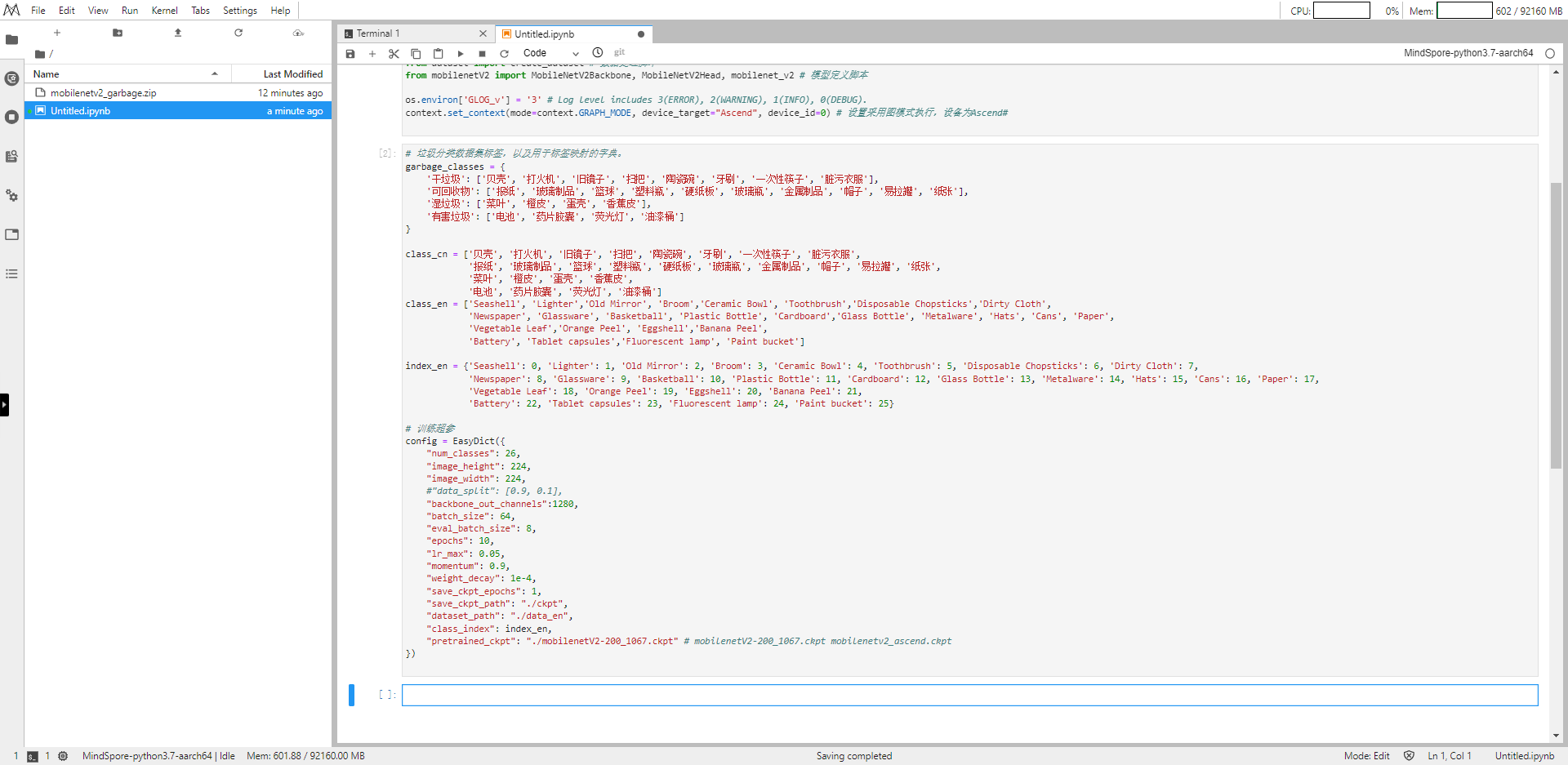


图 17 配置参数的运行结果

下面的代码用于展示部分处理后的数据，复制下面的代码到新的cell中，并按下alt+enter组合键，运行并创建新的cell。

ds = create\_dataset(dataset\_path=config.dataset\_path, config=config, training=False)

print(ds.get\_dataset\_size())

data = ds.create\_dict\_iterator(output\_numpy=True).\_get\_next()

images = data['image']

labels = data['label']

for i in range(1, 5):

plt.subplot(2, 2, i)

plt.imshow(np.transpose(images[i], (1,2,0)))

plt.title('label: %s' % class\_en[labels[i]])

plt.xticks([])

plt.show()

结果如下图所示。

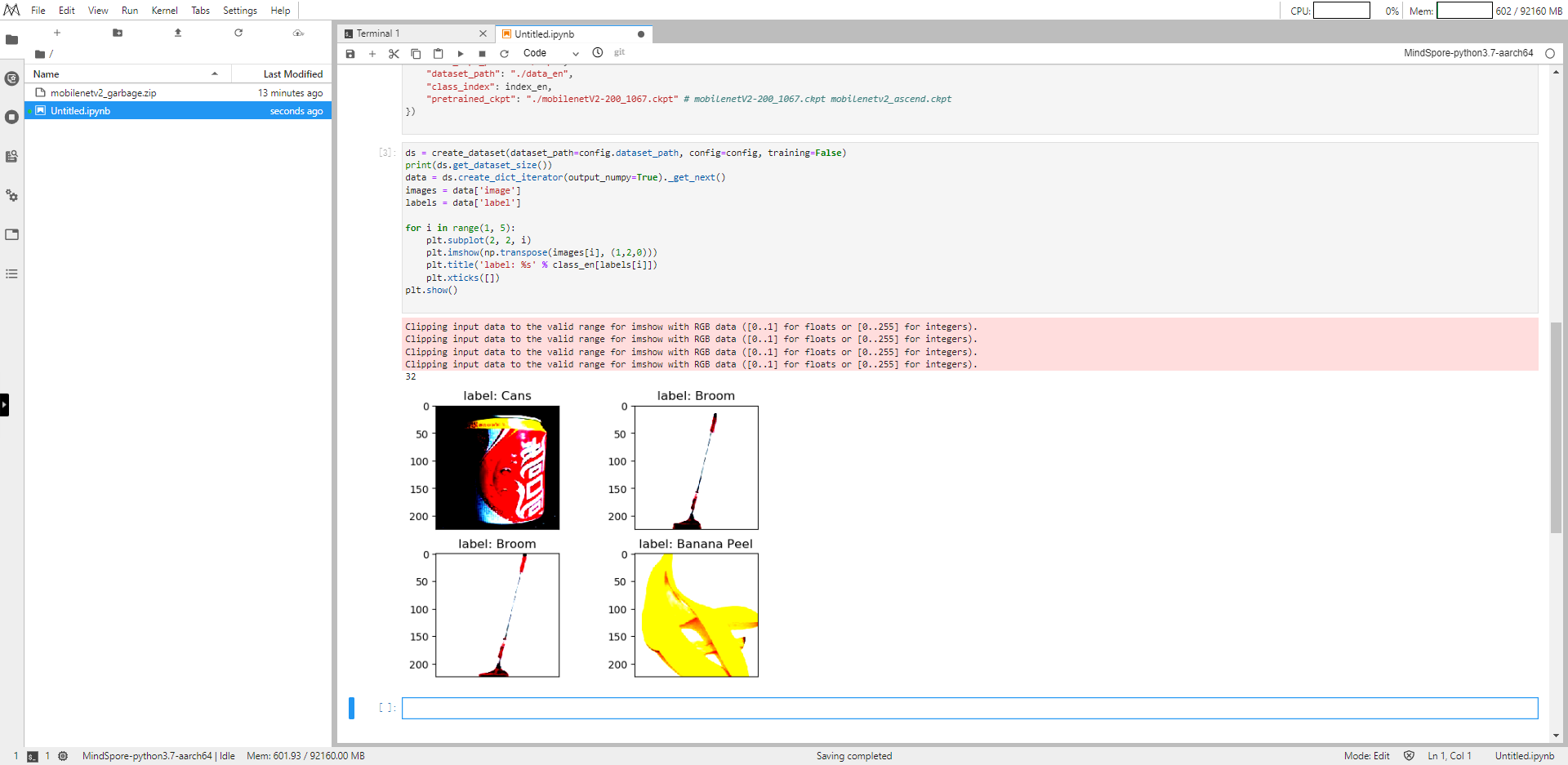


图 18 展示部分处理后的数据结果

复制下面的代码，并允许，按下alt+enter组合键，它用于指定网络的训练策略。（注意：return语句属于该函数，缩进应该是4个空格）

def cosine\_decay(total\_steps, lr\_init=0.0, lr\_end=0.0, lr\_max=0.1, warmup\_steps=0):

"""

Applies cosine decay to generate learning rate array.

Args:

total\_steps(int): all steps in training.

lr\_init(float): init learning rate.

lr\_end(float): end learning rate

lr\_max(float): max learning rate.

warmup\_steps(int): all steps in warmup epochs.

Returns:

list, learning rate array.

"""

lr\_init, lr\_end, lr\_max = float(lr\_init), float(lr\_end), float(lr\_max)

decay\_steps = total\_steps - warmup\_steps

lr\_all\_steps = []

inc\_per\_step = (lr\_max - lr\_init) / warmup\_steps if warmup\_steps else 0

for i in range(total\_steps):

if i < warmup\_steps:

lr = lr\_init + inc\_per\_step \* (i + 1)

else:

cosine\_decay = 0.5 \* (1 + math.cos(math.pi \* (i - warmup\_steps) / decay\_steps))

lr = (lr\_max - lr\_end) \* cosine\_decay + lr\_end

lr\_all\_steps.append(lr)

return lr\_all\_steps

结果如下所示。

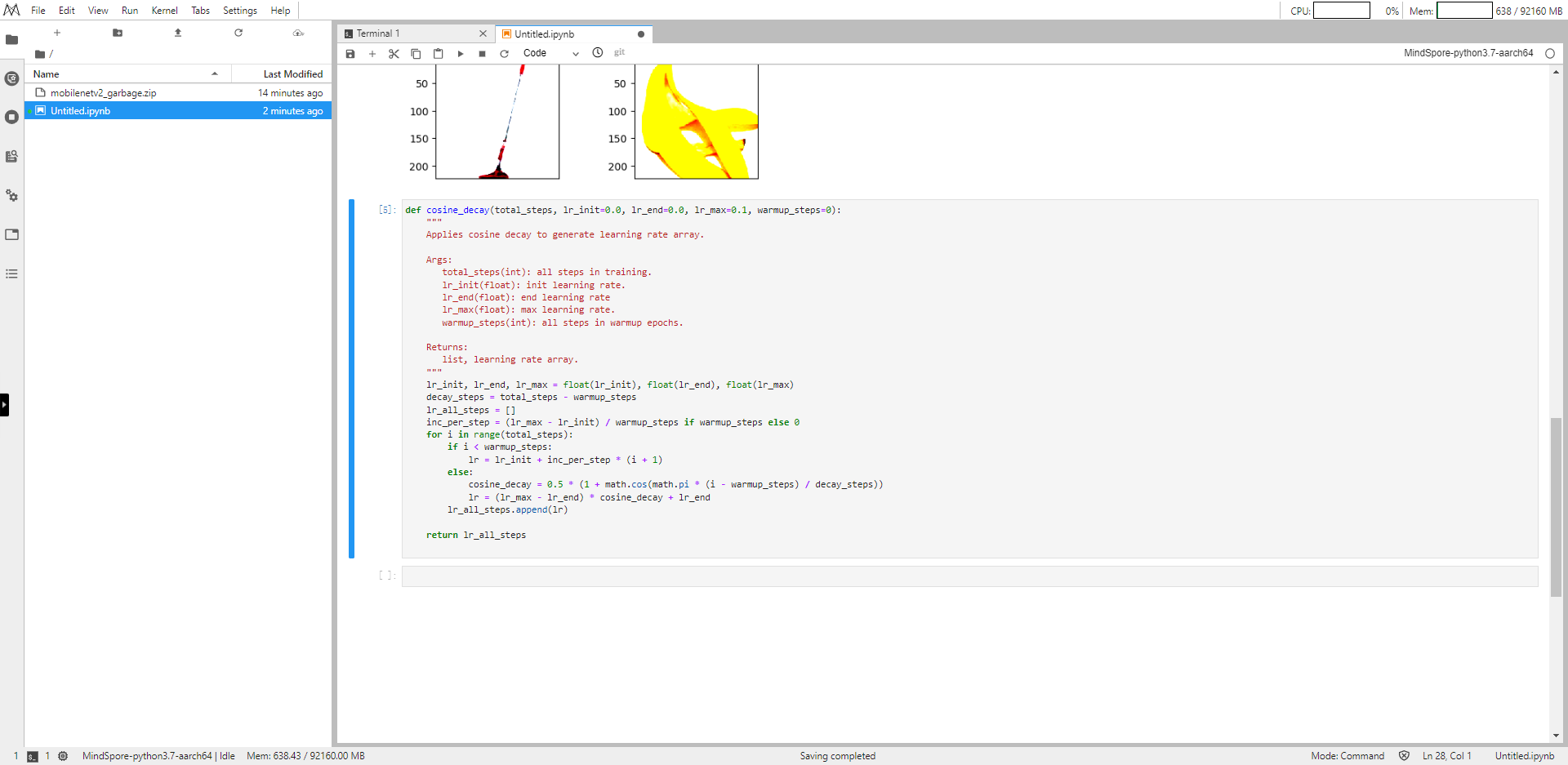


图 19 定义训练策略的代码运行结果

复制下面代码，并运行。

def switch\_precision(net, data\_type):

if context.get\_context('device\_target') == "Ascend":

net.to\_float(data\_type)

for \_, cell in net.cells\_and\_names():

if isinstance(cell, nn.Dense):

cell.to\_float(ms.float32)

运行结果。

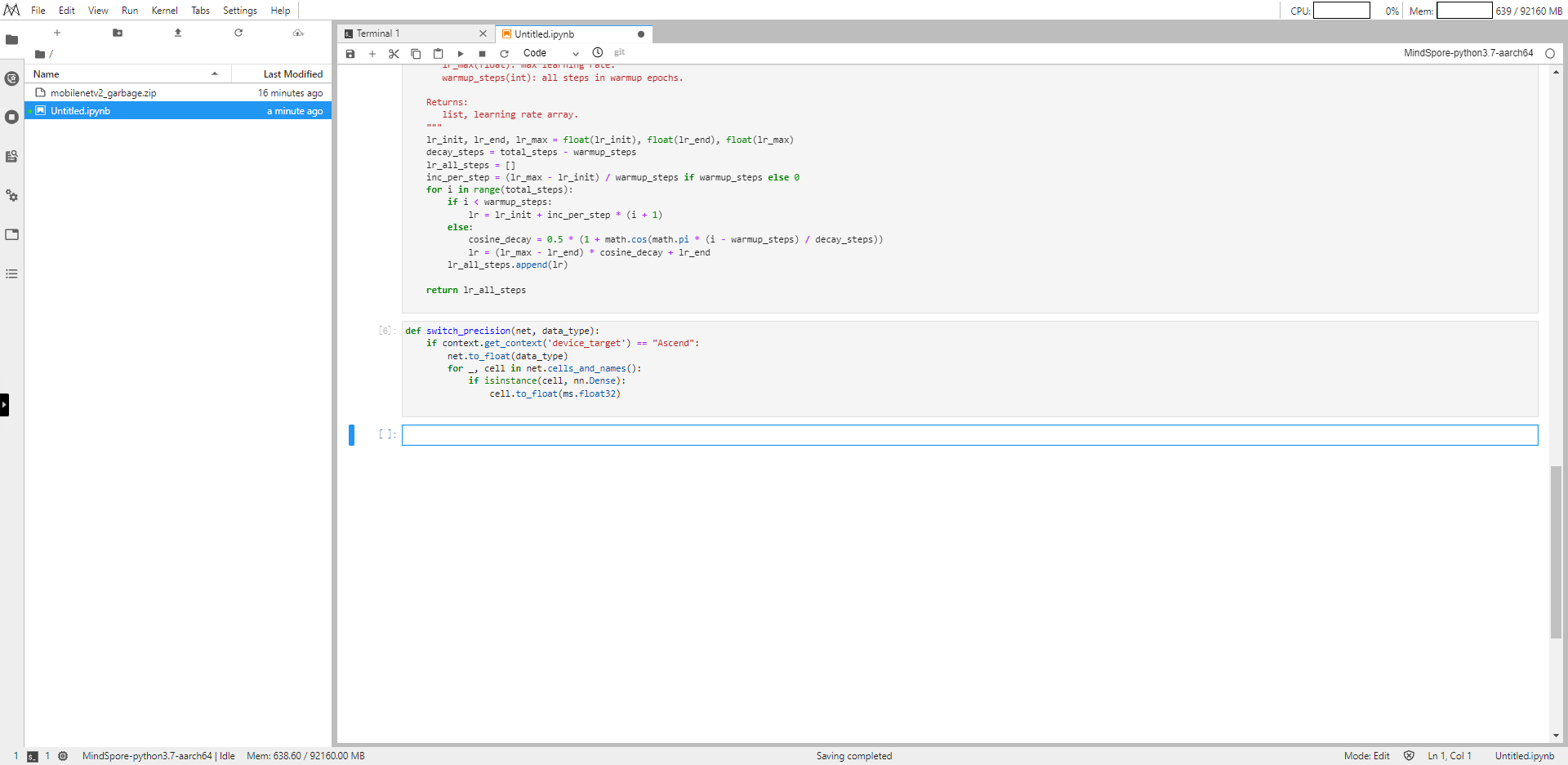


图 20转换模型精度

复制下面的代码，定义一个类。

class EvalCallback(Callback):

def \_\_init\_\_(self, model, eval\_dataset, history, eval\_epochs=1):

self.model = model

self.eval\_dataset = eval\_dataset

self.eval\_epochs = eval\_epochs

self.history = history

self.acc\_max = 0

# acc连续5次<=过程中的最大值，则停止训练

self.count\_max = 5

self.count = 0

def epoch\_begin(self, run\_context):

self.losses = []

self.startime = time.time()

def step\_end(self, run\_context):

cb\_param = run\_context.original\_args()

loss = cb\_param.net\_outputs

self.losses.append(loss.asnumpy())

def epoch\_end(self, run\_context):

cb\_param = run\_context.original\_args()

cur\_epoch = cb\_param.cur\_epoch\_num

train\_loss = np.mean(self.losses)

time\_cost = time.time() - self.startime

if cur\_epoch % self.eval\_epochs == 0:

metric = self.model.eval(self.eval\_dataset, dataset\_sink\_mode=False)

self.history["epoch"].append(cur\_epoch)

self.history["eval\_acc"].append(metric["acc"])

self.history["eval\_loss"].append(metric["loss"])

self.history["train\_loss"].append(train\_loss)

self.history["time\_cost"].append(time\_cost)

if self.acc\_max < metric["acc"]:

self.count = 0

self.acc\_max = metric["acc"]

else:

self.count += 1

if self.count == self.count\_max:

run\_context.request\_stop()

print("epoch: %d, train\_loss: %f, eval\_loss: %f, eval\_acc: %f, time\_cost: %f" %(cur\_epoch, train\_loss, metric["loss"], metric["acc"], time\_cost))

运行结果如下。

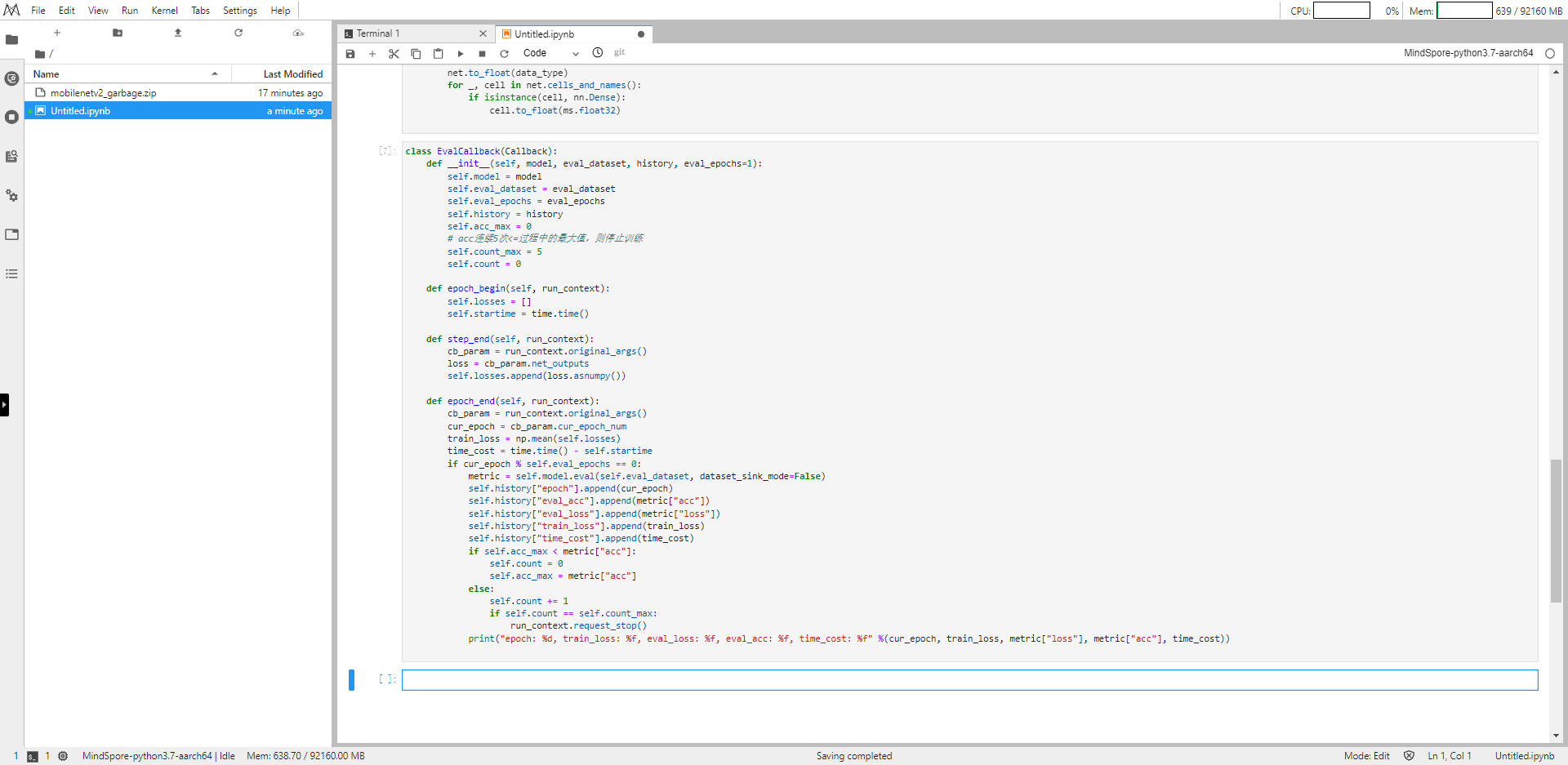


图 21 定义模型验证类

复制下面代码，并运行。它用于定义训练过程。（注意：return属于train函数，缩进是4个空格）

from mindspore.train.loss\_scale\_manager import FixedLossScaleManager

LOSS\_SCALE = 1024

def train():

train\_dataset = create\_dataset(dataset\_path=config.dataset\_path, config=config)

eval\_dataset = create\_dataset(dataset\_path=config.dataset\_path, config=config)

step\_size = train\_dataset.get\_dataset\_size()

backbone = MobileNetV2Backbone() #last\_channel=config.backbone\_out\_channels

# Freeze parameters of backbone. You can comment these two lines.

for param in backbone.get\_parameters():

param.requires\_grad = False

# load parameters from pretrained model

load\_checkpoint(config.pretrained\_ckpt, backbone)

# head = MobileNetV2Head(num\_classes=config.num\_classes, last\_channel=config.backbone\_out\_channels)

head = MobileNetV2Head(input\_channel=backbone.out\_channels, num\_classes=config.num\_classes)

network = mobilenet\_v2(backbone, head)

# define loss, optimizer, and model

loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')

loss\_scale = FixedLossScaleManager(LOSS\_SCALE, drop\_overflow\_update=False)

lrs = cosine\_decay(config.epochs \* step\_size, lr\_max=config.lr\_max)

opt = nn.Momentum(network.trainable\_params(), lrs, config.momentum, config.weight\_decay, loss\_scale=LOSS\_SCALE)

model = Model(network, loss, opt, loss\_scale\_manager=loss\_scale, metrics={'acc', 'loss'})

history = {'epoch': [], 'train\_loss': [], 'eval\_loss': [], 'eval\_acc': [], 'time\_cost':[]}

eval\_cb = EvalCallback(model, eval\_dataset, history)

cb = [eval\_cb]

ckpt\_cfg = CheckpointConfig(save\_checkpoint\_steps=config.save\_ckpt\_epochs \* step\_size, keep\_checkpoint\_max=config.epochs)

ckpt\_cb = ModelCheckpoint(prefix="mobilenetv2", directory=config.save\_ckpt\_path, config=ckpt\_cfg)

cb.append(ckpt\_cb)

model.train(50, train\_dataset, callbacks=cb, dataset\_sink\_mode=False)

return history

运行结果如下。

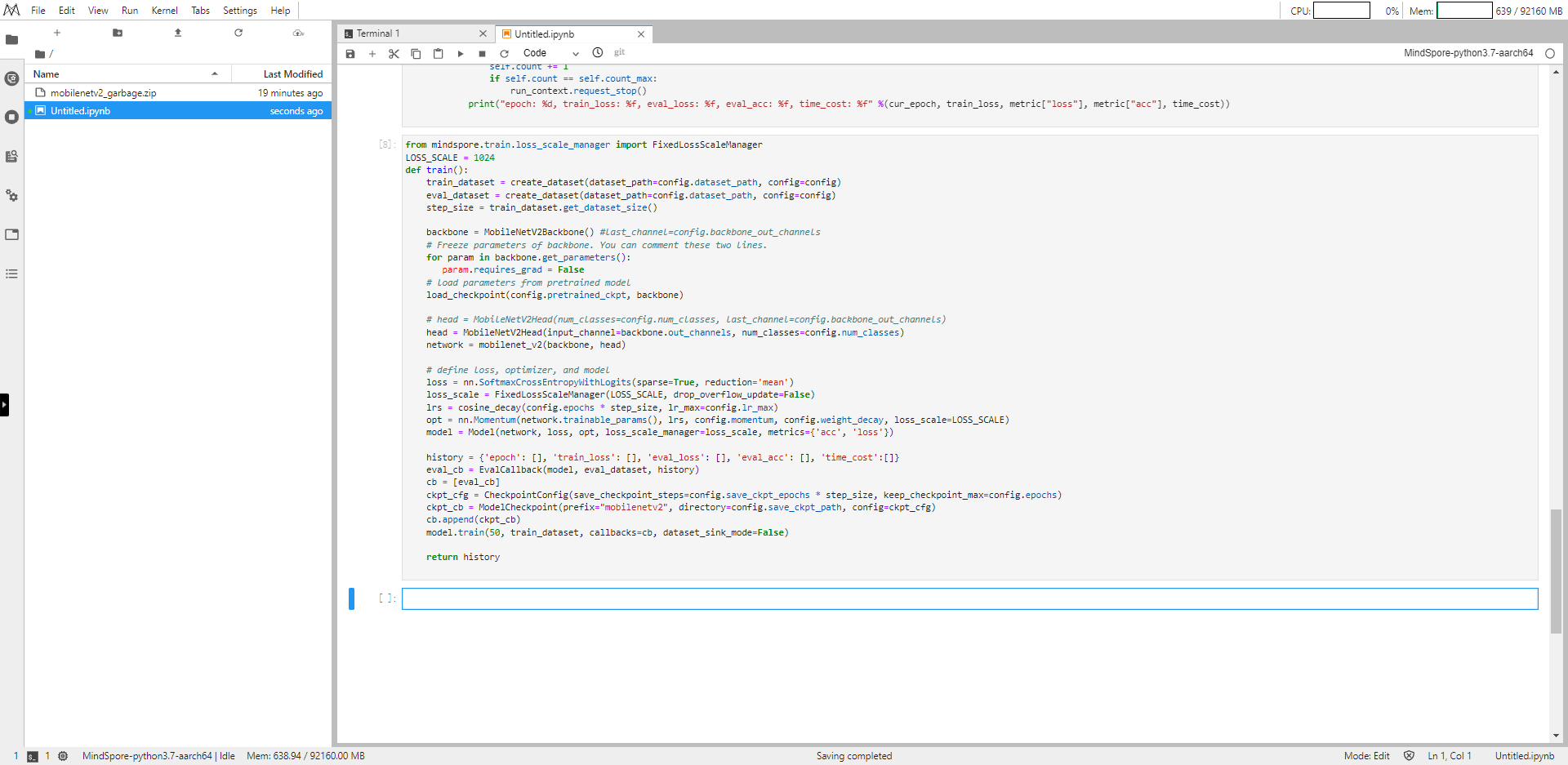


图 22 定义训练过程结果

下面的代码用于执行训练过程，并保存最优的模型，因为这个是训练过程，所以耗时会久一点。

if os.path.exists(config.save\_ckpt\_path):

shutil.rmtree(config.save\_ckpt\_path)

history = train()

plt.plot(history['epoch'], history['train\_loss'], label='train\_loss')

plt.plot(history['epoch'], history['eval\_loss'], 'r', label='val\_loss')

plt.legend()

plt.show()

plt.plot(history['epoch'], history['eval\_acc'], 'r', label = 'val\_acc')

plt.legend()

plt.show()

CKPT = 'mobilenetv2-%d\_40.ckpt' % (np.argmax(history['eval\_acc']) + 1) # 挑选最优模型Checkpoint，根据数据量和batch\_size修改81值

print("Chosen checkpoint is", CKPT)

运行结果如下所示。

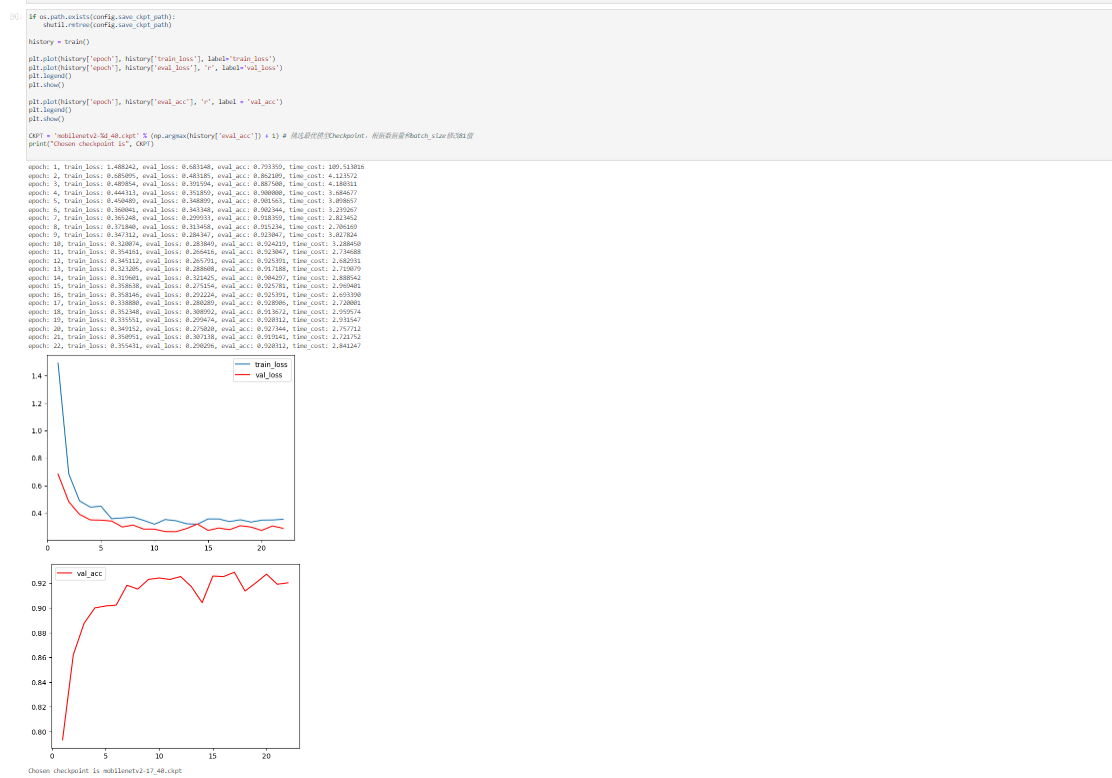


图 23 运行结果

然后定义CKPT变量为下面的值，其中17需要自己指定，因为上图输出的结果表示最优的效果出现在ecpoch为17的地方。

CKPT="mobilenetv2-17\_40.ckpt"

运行，并执行，alt+enter。

下面的代码定义了模型推理的过程。

def image\_process(image):

"""Precess one image per time.

Args:

image: shape (H, W, C)

"""

mean=[0.485\*255, 0.456\*255, 0.406\*255]

std=[0.229\*255, 0.224\*255, 0.225\*255]

image = (np.array(image) - mean) / std

image = image.transpose((2,0,1))

img\_tensor = Tensor(np.array([image], np.float32))

return img\_tensor

def infer\_one(network, image\_path):

image = Image.open(image\_path).resize((config.image\_height, config.image\_width))

logits = network(image\_process(image))

pred = np.argmax(logits.asnumpy(), axis=1)[0]

print(image\_path, class\_en[pred])

def infer():

backbone = MobileNetV2Backbone(last\_channel=config.backbone\_out\_channels)

head = MobileNetV2Head(input\_channel=backbone.out\_channels, num\_classes=config.num\_classes)

network = mobilenet\_v2(backbone, head)

load\_checkpoint(os.path.join(config.save\_ckpt\_path, CKPT), network)

for i in range(91, 100):

infer\_one(network, f'data\_en/test/Cardboard/000{i}.jpg')

infer()

运行结果如下所示。

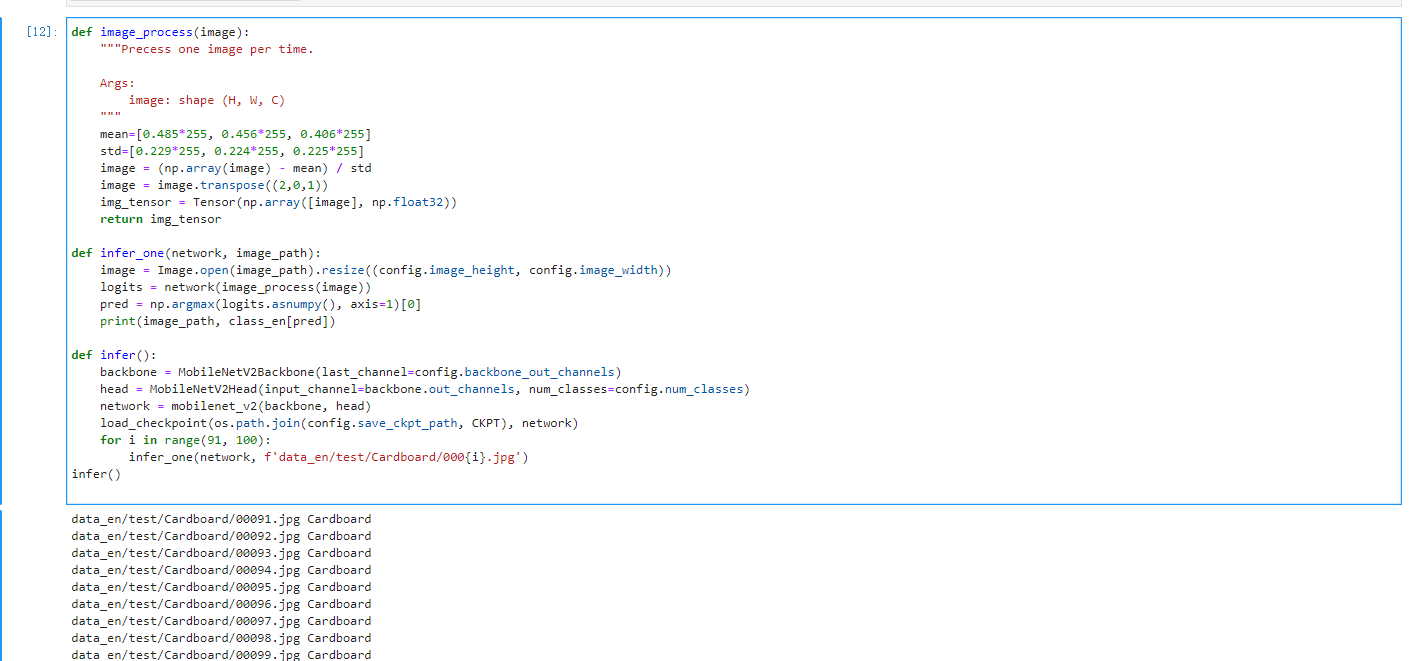


图 24 推理结果

下面的代码用于保存训练的模型，这里选择导出ONNX格式。

backbone = MobileNetV2Backbone(last\_channel=config.backbone\_out\_channels)

head = MobileNetV2Head(input\_channel=backbone.out\_channels, num\_classes=config.num\_classes)

network = mobilenet\_v2(backbone, head)

load\_checkpoint(os.path.join(config.save\_ckpt\_path, CKPT), network)

input = np.random.uniform(0.0, 1.0, size=[1, 3, 224, 224]).astype(np.float32)

# export(network, Tensor(input), file\_name='mobilenetv2.air', file\_format='AIR') # MindSpore 1.0

# export(network, Tensor(input), file\_name='mobilenetv2.pb', file\_format='GEIR') # MindSpore 0.5

export(network, Tensor(input), file\_name='mobilenetv2.onnx', file\_format='ONNX')

运行结果如下图所示。

## 导出AIR模型文件

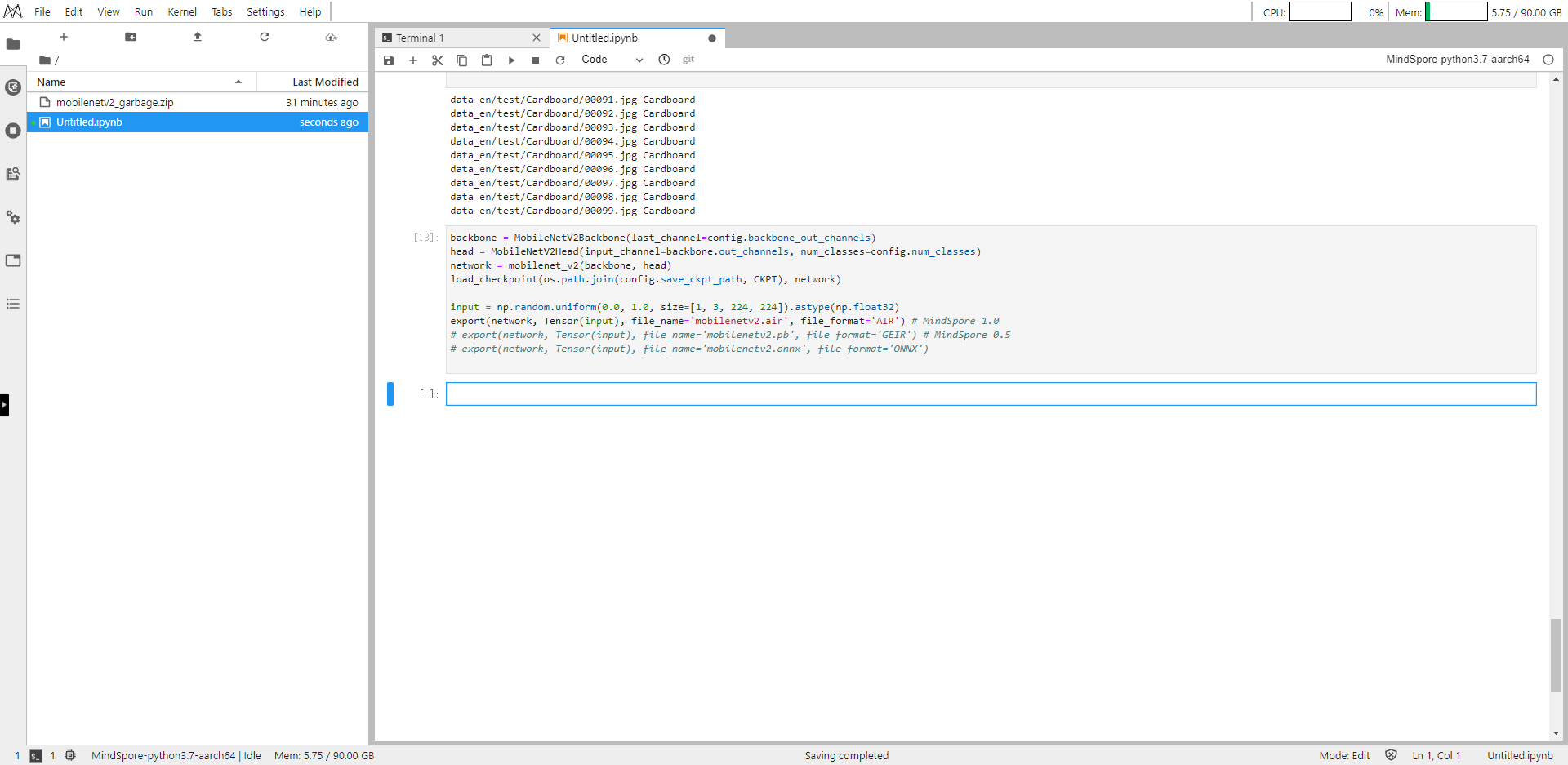


图 25 导出模型

下面的代码将生成的onnx文件存放到创建的obs文件夹中，下面的obs-random-jkl1/123需要根据实际情况修改。

import moxing as mox

mox.file.copy('/home/ma-user/work/mobilenetv2.onnx', 'obs://obs-random-ljk1/123/mobilenetv2.onnx')

运行结果如下所示。



图 26 将生成的air文件放入OBS桶内

在OBS桶界面可以查看到已有mobilenetv2.onnx文件。

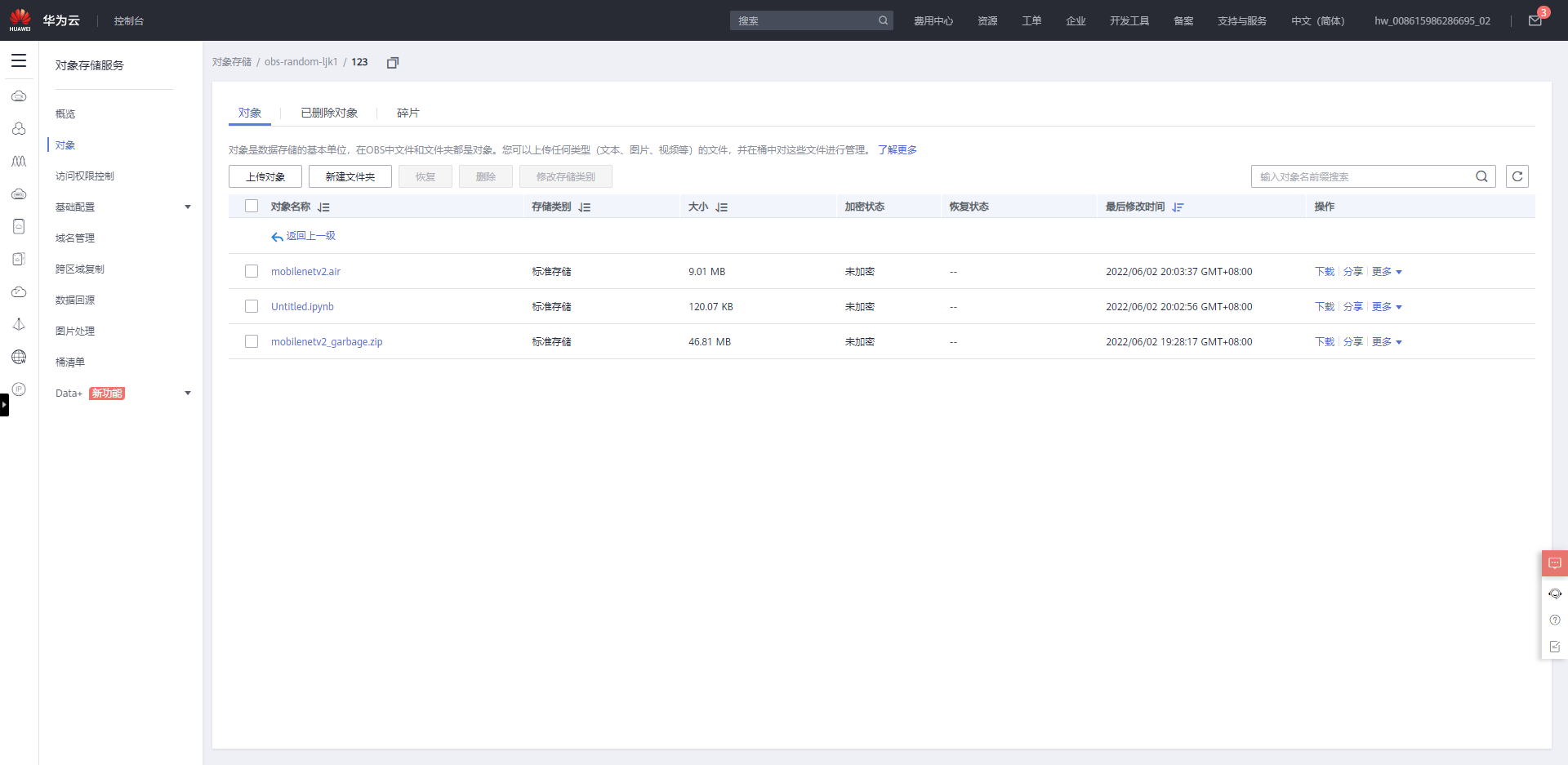


图 27 在OBS桶内查看导出的模型文件

可将该文件下载到本地，用以完成后续的步骤。