

华为云微认证系列

# MindSpore模型快速调优攻略

## 实验指导手册

版本:1.0



华为技术有限公司

**版权所有 © 华为技术有限公司 2021。 保留一切权利。**

未经本公司书面许可，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**



HUAWEI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暗示的声明或保证。

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

# **华为技术有限公司**

地址：                    深圳市龙岗区坂田华为总部办公楼                    邮编：518129

网址：                    <http://edu.huaweicloud.com>

# 目录

<b>背景介绍 .....</b>	<b>1</b>
<b>实验目的 .....</b>	<b>1</b>
<b>注意事项 .....</b>	<b>1</b>
<b>1 使用 MindConverter 完成 BERT 模型迁移.....</b>	<b>2</b>
1.1 实验介绍 .....	2
1.2 实验流程 .....	2
1.3 资源准备 .....	2
1.3.1 安装 MindSpore 及 MindInsight.....	2
1.3.2 下载预训练 BERT 模型至 Linux 环境 .....	3
1.4 使用 MindConverter 迁移 BERT 模型 .....	3
<b>2 基于 MindSpore 完成情感分类模型训练.....</b>	<b>5</b>
2.1 实验介绍 .....	5
2.2 实验流程 .....	5
2.3 资源准备 .....	5
2.3.1 创建华为云账号访问密钥.....	6
2.3.2 创建 OBS 数据桶 .....	7
2.3.3 安装 MindSpore IDE 插件并配置华为云访问密钥 .....	8
2.4 完成情感分类下游任务开发.....	9
2.5 结果验证 .....	17
<b>3 资源释放 .....</b>	<b>19</b>
3.1 实验说明 .....	19

## 背景介绍

近年来，深度学习技术在语音识别、自然语言处理、计算机视觉、信息检索等任务上取得了突破性进展。然而，随着深度学习模型的复杂度与规模日益扩张，导致模型的调试调优成为了困扰算法工程师的一大难题。

MindSpore 是由华为自研的深度学习框架，最佳匹配昇腾 AI 处理器算力，为数据科学家和算法工程师提供设计友好、运行高效的开发体验。在广泛吸纳行业用户诉求后，MindSpore 推出 MindInsight 工具包，涵盖生态迁移、调试调优，以解决用户关键诉求。

实验通过基于 BERT 情感分类模型的调试调优，展示 MindSpore 的快速调试调优、端云协同的能力。

## 实验目的

本实验指导包含两个实验任务：使用 MindConverter 完成 BERT 模型迁移，使用调试调优工具完成情感分类模型训练。通过本实验，您将能够：

- 了解MindSpore调试调优工具包MindInsight的使用
- 掌握MindSpore IDE的基本使用
- 利用华为云的ModelArts与OBS服务完成MindSpore模型训练

## 注意事项

实验资源一旦购买就开始计费，请合理安排时间进行实验，并注意以下几点：

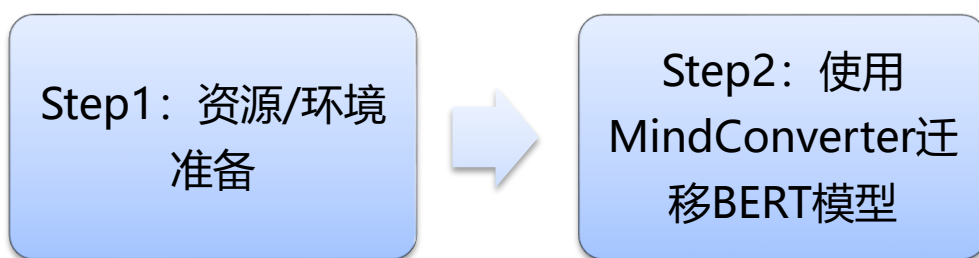
- 本实验预计 1 小时完成（使用 ModelArts 训练的单次训练时长小于 10 分钟），实验结束后请停止 MindInsight 可视化作业，避免继续计费；
- 若实验中途离开或中断，建议释放实验资源，否则将会按照购买的资源继续计费；

# 1 使用 MindConverter 完成 BERT 模型迁移

## 1.1 实验介绍

使用 MindSpore 生态迁移工具 MindConverter 迁移中文 BERT 预训练语言模型至 MindSpore，通过迁移报告验证迁移是否成功。

## 1.2 实验流程



## 1.3 资源准备

进行实验前，需提前配置好以下资源，实验中涉及的模型转换需要在 Ubuntu Python 3.7.5 环境下进行（MindInsight 当前仅支持 Linux 操作系统，后续逐步支持 macOS、Windows 操作系统）：

- 安装 MindSpore 1.2.0 以及 MindInsight Master。
- 安装 MindConverter 所需的 Python 三方依赖。
- 下载原始 BERT 模型（模型为 ONNX 格式）至 Linux 环境。

### 1.3.1 安装 MindSpore 及 MindInsight

步骤 1 通过以下命令安装 MindSpore（为加速安装包下载，使用华为云安装资源）：

```
pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.2.0/MindSpore/cpu/ubuntu_x86/mindspore-1.2.0-cp37-cp37m-linux_x86_64.whl --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com -i https://pypi.tuna.tsinghua.edu.cn/simple.
```

步骤 2 通过以下命令安装 MindInsight (为加速安装包下载, 使用华为云安装资源):

```
pip install https://lau-hdc-demo.obs.cn-north-4.myhuaweicloud.com/experiment%20resources/mindinsight-1.2.0.master-cp37-cp37m-linux_x86_64.whl --trusted-host lau-hdc-demo.obs.cn-north-4.myhuaweicloud.com -i https://pypi.tuna.tsinghua.edu.cn/simple.
```

步骤 3 安装 MindConverter 所需 Python 三方依赖库:

```
pip install "onnx>=1.8.0" "onnxoptimizer>=0.1.2" "onnxruntime>=1.4.0" "tensorflow" -i https://pypi.tuna.tsinghua.edu.cn/simple.
```

(TensorFlow 作为 onnxruntime 执行引擎被使用, 其版本无要求, 若用户当前环境已安装 TensorFlow, 可不重复安装)

步骤 4 通过如下命令验证所需 Python 依赖是否安装:

```
pip list | grep "onnx\|tensorflow\|mindinsight\|mindspore",
```

预期结果如下图所示。(命令执行后出现 mindspore、mindspore-gpu、mindspore-ascend 均可)

```
(lau) root@ubuntu:demo# pip list | grep "onnx\|tensorflow\|mindinsight\|mindspore"
mindinsight          1.2.0
mindspore-gpu        1.2.0
onnx                  1.8.1
onnxoptimizer         0.2.5
onnxruntime           1.7.0
tensorflow-addons     0.12.0
tensorflow-datasets   3.0.0
tensorflow-estimator  1.15.1
tensorflow-gpu        1.15.3
tensorflow-hub        0.7.0
tensorflow-metadata   0.24.0
```

## 1.3.2 下载预训练 BERT 模型至 Linux 环境

步骤 1 点击[模型下载地址](https://weirenzheng.obs.cn-north-1.myhuaweicloud.com/MindSpore%E6%A8%A1%E5%9E%8B%E5%BF%AB%E9%80%9F%E8%B0%83%E4%BC%98%E6%94%BB%E7%95%A5%E8%BD%AF%E4%BB%B6/bert_zh.onnx)通过浏览器下载模型文件, 或通过命令

```
wget -c 'https://weirenzheng.obs.cn-north-1.myhuaweicloud.com/MindSpore%E6%A8%A1%E5%9E%8B%E5%BF%AB%E9%80%9F%E8%B0%83%E4%BC%98%E6%94%BB%E7%95%A5%E8%BD%AF%E4%BB%B6/bert_zh.onnx'
```

将模型下载至 Linux 环境 (模型名称为 bert\_zh.onnx), 预期结果如下图所示。

```
(lau) root@ubuntu:demo# ll
total 399960
drwxr-xr-x  2 root root    4096 Apr  7 19:17 ./
drwxr-xr-x 42 root root    4096 Apr  7 18:22 ../
-rw-r--r--  1 root root 409141380 Apr  7 19:11 bert_zh.onnx
```

## 1.4 使用 MindConverter 迁移 BERT 模型

步骤 1 通过 cd 命令进入下载好的模型所在目录, 使用图示命令进行模型迁移。命令执行后, 若打印 "[INFO] MINDCONVERTER: MindConverter: conversion is completed." 信息, 则说明迁移完成, 迁移结果位于当前目录的 output 目录中。

```
(lau) root@ubuntu:demo# mindconverter --model_file bert_zh.onnx --shape 1,512 1,512 1,512 \
> --input_nodes input_ids attention_mask token_type_ids \
> --output_nodes output_0 output_1
WARNING: 'Pack' is deprecated from version 1.1 and will be removed in a future version, use 'Stack' instead.
[INFO] MINDCONVERTER: MindConverter: conversion is completed.
```

步骤 2 通过 cat 命令查看 output 目录中的迁移报告 (report\_of\_bert\_zh.txt)，迁移率为 100%，即说明迁移成功，预期结果如下图所示。

```
(lau) root@ubuntu:demo# cat output/report_of_bert_zh.txt
[Start Convert]
[Convert Over]
Converted Rate: 100.00%.
```

步骤 3 若无法获取 Linux 环境，可下载转换后结果，完成后续实验步骤，[下载链接](#)。

步骤 4 通过任意文本编辑器修改迁移后的 bert\_zh.py 中定义的模型规格，使其可接受任意批大小 (Batch size) 的训练、推理数据；模型输入数据尺寸通常受 Reshape 算子影响，因此，我们找到 Reshape 算子对应的操作数，将其第一个维度的值由 “1” 改为 “-1” 即可，修改后代码见下图；至此，**实验一 BERT 模型迁移**已完成。

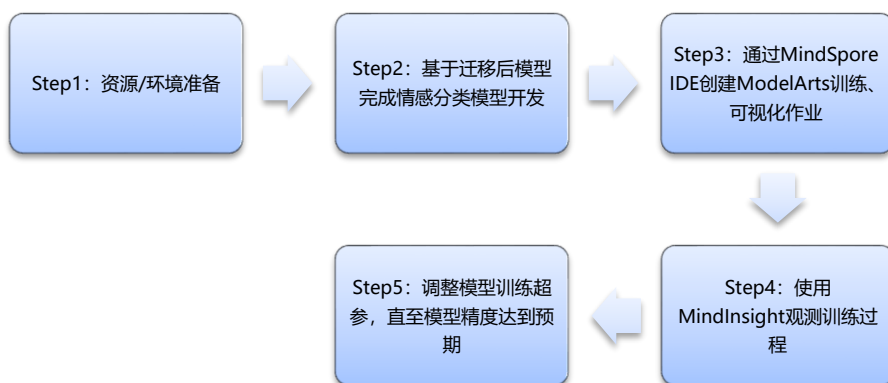
```
class MultiHeadAttn(nn.Cell):
    def __init__(self):
        super(MultiHeadAttn, self).__init__()
        self.matmul_0 = nn.MatMul()
        self.matmul_0_w = Parameter(Tensor(np.random.uniform(0, 1, (768, 768)).astype(np.float32)), name=None)
        self.matmul_1 = nn.MatMul()
        self.matmul_1_w = Parameter(Tensor(np.random.uniform(0, 1, (768, 768)).astype(np.float32)), name=None)
        self.matmul_2 = nn.MatMul()
        self.matmul_2_w = Parameter(Tensor(np.random.uniform(0, 1, (768, 768)).astype(np.float32)), name=None)
        self.add_3 = P.Add()
        self.add_3_bias = Parameter(Tensor(np.random.uniform(0, 1, (768,)).astype(np.float32)), name=None)
        self.add_4 = P.Add()
        self.add_4_bias = Parameter(Tensor(np.random.uniform(0, 1, (768,)).astype(np.float32)), name=None)
        self.add_5 = P.Add()
        self.add_5_bias = Parameter(Tensor(np.random.uniform(0, 1, (768,)).astype(np.float32)), name=None)
        self.reshape_6 = P.Reshape()
        self.reshape_6_shape = tuple([-1, 512, 12, 64])
        self.reshape_7 = P.Reshape()
        self.reshape_7_shape = tuple([-1, 512, 12, 64])
        self.reshape_8 = P.Reshape()
        self.reshape_8_shape = tuple([-1, 512, 12, 64])
        self.transpose_9 = P.Transpose()
        self.transpose_10 = P.Transpose()
        self.transpose_11 = P.Transpose()
        self.matmul_12 = nn.MatMul()
        self.div_13 = P.Div()
        self.div_13_w = 8.0
        self.add_14 = P.Add()
        self.softmax_15 = nn.Softmax(axis=3)
        self.matmul_16 = nn.MatMul()
        self.transpose_17 = P.Transpose()
        self.reshape_18 = P.Reshape()
        self.reshape_18_shape = tuple([-1, 512, 768])
        self.matmul_19 = nn.MatMul()
        self.matmul_19_w = Parameter(Tensor(np.random.uniform(0, 1, (768, 768)).astype(np.float32)), name=None)
        self.add_20 = P.Add()
        self.add_20_bias = Parameter(Tensor(np.random.uniform(0, 1, (768,)).astype(np.float32)), name=None)
```

# 2 基于 MindSpore 完成情感分类模型训练

## 2.1 实验介绍

基于迁移后 BERT 模型，在 MindSpore 下开发情感分类下游任务，借助 MindSpore 调试调优工具完成模型训练，使其精度达到 90%以上。

## 2.2 实验流程



## 2.3 资源准备

进行实验前，需提前配置好以下资源，实验中涉及的模型训练、训练过程可视均在“**华北-北京四**”区域下进行：

- 已成功注册华为云账号并通过实名认证。
- 创建华为云账号的访问密钥（AK、SK）。
- 访问华为云 OBS 服务，创建 OBS 数据桶。
- 访问华为云 ModelArts 服务，允许其访问 OBS 资源。
- 基于 PyCharm 安装 MindSpore IDE 插件（请确保 PyCharm 版本高于 2020.3）。
- 基于 MindSpore IDE 插件配置华为云访问密钥（AK、SK）。



## 2.3.1 创建华为云账号访问密钥

步骤 1 访问华为云官网，注册账户、并完成实名认证。



步骤 2 进入“账户中心”，点击“管理我的凭证”，进入凭证管理页面，切换至“访问密钥”标签，点击“新增访问密钥”即可完成创建，创建完成后将密钥文件（.csv）下载至本地（密钥 csv 文件仅可下载一次，请妥善保管，2.3.3 步骤三中会使用）。



### 基本信息



#### 基本信息

帐号名	<div></div>	
帐号类型	<div></div>	
注册时间	2021/03/25 09:50:29 GMT+08:00	
企业名称	<div></div>	
姓名		<a href="#">修改</a>
职位	未选择职位	<a href="#">修改</a>
手机号码	尚未绑定	<a href="#">修改</a>
注册邮箱	<div></div>	<a href="#">修改</a>
密码	*****	<a href="#">修改</a>
认证信息	<div></div>	<a href="#">查看</a>
安全凭证		<a href="#">管理我的凭证</a>

#### 联系信息

联系信息更新后，华为云将会通过以下信息联系您

联系地址	尚未设置联系地址	<a href="#">设置联系地址</a>
联系方式	尚未设置联系方式	<a href="#">设置联系方式</a>



我的凭证

---

API凭证

[访问密钥](#)

### 访问密钥

如果访问密钥泄露，会带来数据泄露风险，且每个访问密钥仅能下载一次，为了帐号安全性，建议您定期更换并妥善保存访问密钥。

[新增访问密钥](#) 您还可以添加1个访问密钥。

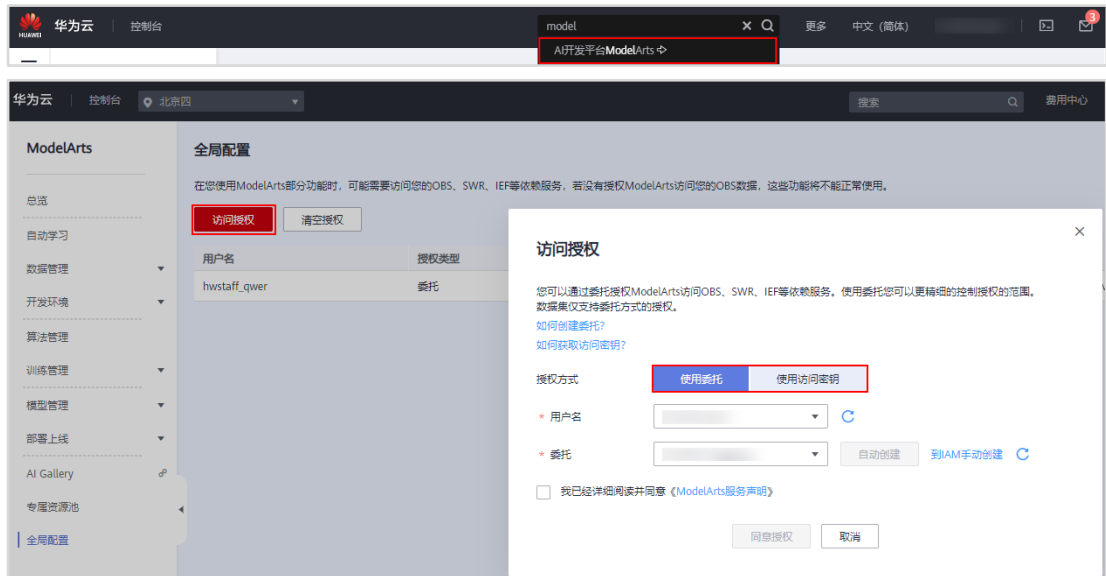
访问密钥ID	描述	创建时间
<div></div>		

## 2.3.2 创建 OBS 数据桶

步骤 1 进入华为云 OBS 控制台（通过华为云页面上方中间的搜索栏，输入“**OBS**”关键字，点击自动关联出的“**对象存储服务 OBS**”即可进入控制台），点击“**创建桶**”，创建出数据桶实例（数据桶需创建在“**华北-北京四**”区域，其余选项默认即可）。

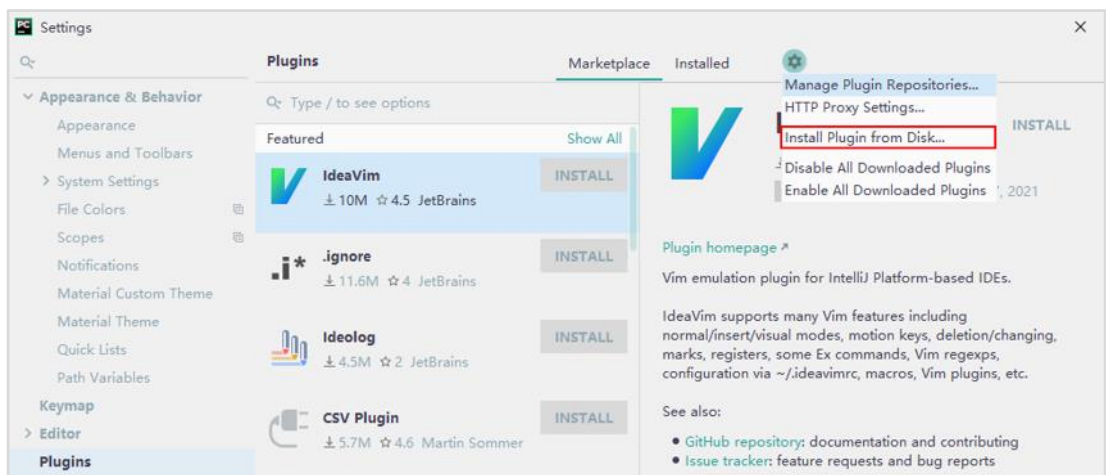


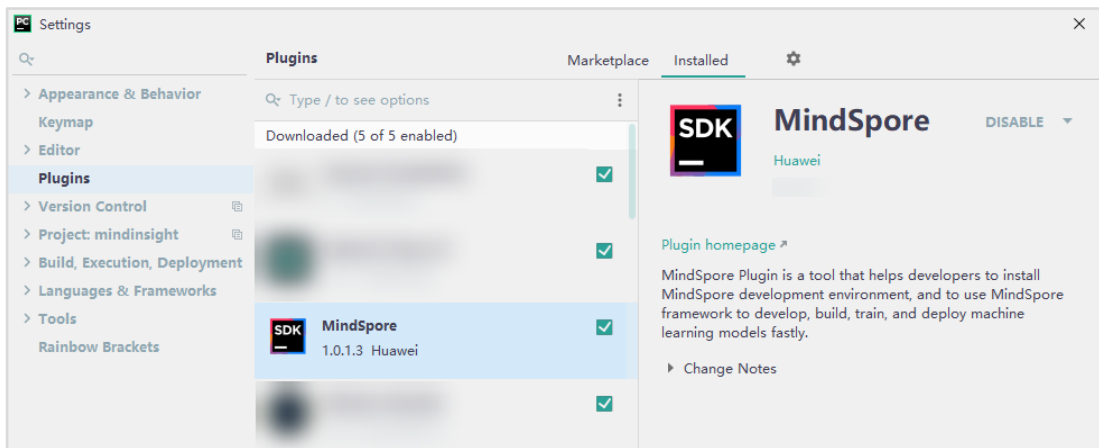
步骤 2 进入华为云 ModelArts 控制台（**注意**：需要通过网页左上角切换至**华北-北京四**区），在左侧导航栏单击“**全局配置**”，进入“**全局配置**”页面，单击“**访问授权**”后点击“**自动创建**”以创建委托，在勾选“**我已经详细阅读并同意《ModelArts 服务声明》**”后点击“**同意授权**”即可完成授权（建议使用委托的方式进行授权）。



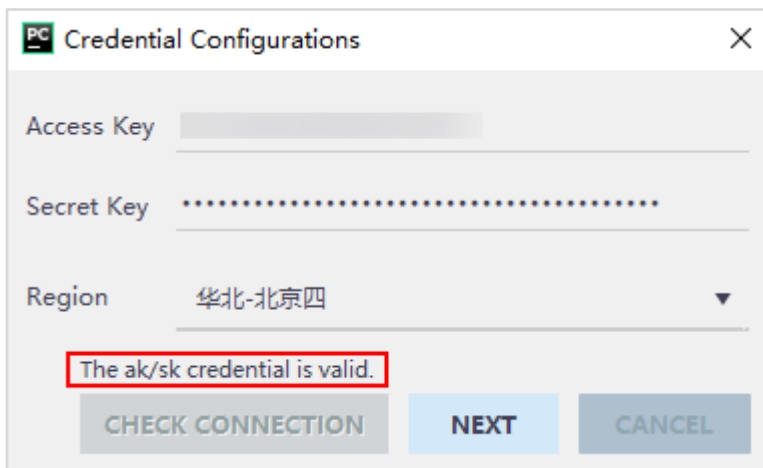
### 2.3.3 安装 MindSpore IDE 插件并配置华为云访问密钥

- 步骤 1 点击[下载地址](#)，将 MindSpore IDE 插件下载至本地（**无需解压插件的 zip 安装包**）。
- 步骤 2 打开 PyCharm，进入设置（Setting），找到插件（Plugins），点击从本地安装（Install Plugin from Disk），选择“步骤 1”中下载的文件，重启 PyCharm；再次进入 PyCharm 插件（Plugins）管理选项，显示插件已成功安装。



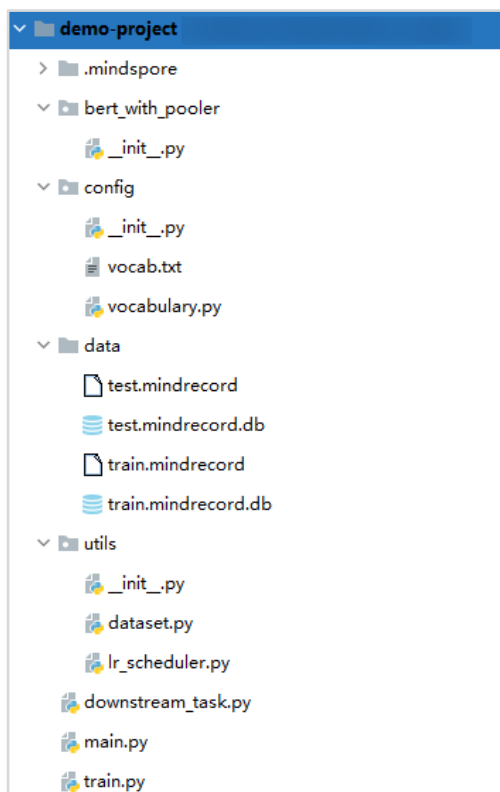


步骤 3 点击 Pycharm 任务栏的“**MindSpore**”菜单中“Cloud”的“**Config Credentials**”，将华为云的访问密钥填入，点击“**Check Connection**”，若出现下图所示提示，则说明密钥配置正确；若成功，点击“**CANCEL**”关闭弹窗即可。

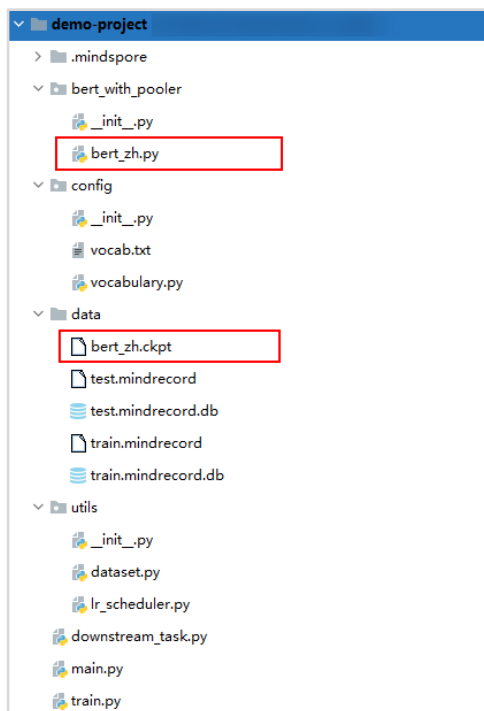


## 2.4 完成情感分类下游任务开发

步骤 1 下载[项目工程](#)至本地环境并解压，使用 PyCharm 打开该项目，目录结构如下图所示；其中 data 目录存放情感分类模型所需训练集、测试集，main.py 为训练启动脚本，train.py 为训练逻辑脚本，downstream\_task.py 为情感分类模型定义脚本。



步骤 2 将实验一中生成的 bert\_zh.py、bert\_zh.ckpt 分别放置于 bert\_with\_pooler、data 目录下（若未进行实验一，可仅将 1.4 步骤三中下载的 bert\_zh.ckpt 放置于 data 目录下即可），放置后项目结构如下图所示。



步骤 3 打开 downstream\_task.py, **注意**, EmotionClassifier 中的情感分类分类器需要用户自行实现, 情感分类模型对应数学表达为  $Scores = Softmax(X \cdot W + b)$ ; downstream\_task.py 中已给出情感分类模型示例代码, 用户可基于示例代码实现情感分类模型; 另外, 用户需基于 mindspore.ops.HistogramSummary 收集分类层的权重 (weight)、偏置 (bias) 分布直方图 (Histogram); 此处, 我们使用半精度 (float16) 作为 BERT 模型、情感分析分类器的运算精度以加速模型训练、推理性能; 最终模型定义代码见下图。

```
class EmotionClassifier(nn.Cell):
    def __init__(self, model):
        super(EmotionClassifier, self).__init__()
        self.pretrained_model = model.to_float(mstype.float16)
        # TODO: Please add classifier below this line.
        self.classifier = MatMul()
        self.weight = Parameter(initializer("normal", [768, 2]), name="weight")
        self.bias = Parameter(initializer("zeros", [2]), name="bias")
        self.softmax = nn.Softmax()
        self.cast = Cast()
        # TODO: Please activate the following commented line to record weights histogram.
        self.histogram_summary = HistogramSummary()

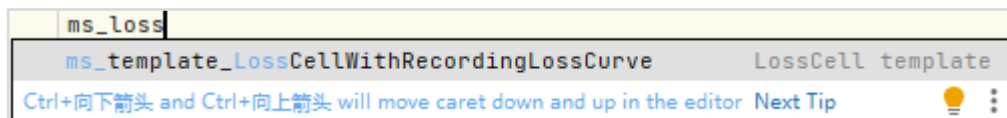
    def construct(self, input_ids, attention_mask, token_type_ids):
        _, pooled_output = self.pretrained_model(input_ids, attention_mask, token_type_ids)
        # (N, D) * (D, classes) -> (N, classes)
        logits = self.classifier(pooled_output, self.cast(self.weight, mstype.float16))
        logits += self.cast(self.bias, mstype.float16)
        # TODO: Please activate the following commented line to record weights histogram.
        self.histogram_summary("classifier_weight", self.weight)
        self.histogram_summary("classifier_bias", self.bias)
        logits = self.cast(logits, mstype.float32)
        scores = self.softmax(logits)
        return scores
```

分类器定义

分类器调用

分类器调用

步骤 4 仍然在 downstream\_task.py 文件中, 借助 MindSpore IDE 的代码模板功能, 创建自定义 LossCell, 以记录训练过程中的损失 (Loss) 曲线; 输入 “ms\_loss” 即可联想到该代码块, 点击键盘 “TAB” 键即可快速生成代码, 见下图所示; 同理, 输入 “ms\_learning”, 点击 “TAB” 键, 生成自定义的 LearningRate 代码块, 该代码块可记录训练过程中学习率的变化 (若 IDE 插件自动补全功能失效, 可取消 downstream\_task.py 中 LossCell、LearningRate 相关代码的注释)。



```
# LossCell template
class LossCell(nn.Cell):
    def __init__(self, model, objective_fn):
        super(LossCell, self).__init__()
        self.model = model
        self.objective_fn = objective_fn
        self.scalar_collector = ScalarSummary()

    def construct(self, input_ids, attention_mask, token_type_ids, labels):
        out = self.model(input_ids, attention_mask, token_type_ids)
        loss = self.objective_fn(out, labels)
        self.scalar_collector('loss', loss)
        return loss
```

ms\_template\_LearningRate LearningRate template

Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor Next Tip

ms\_learning

```
# LearningRate template
class LearningRate(LearningRateSchedule):
    def __init__(self, learning_rates):
        super(LearningRate, self).__init__()
        self.learning_rate = Tensor(learning_rates, mstype.float32)
        self.scalar_collector = ScalarSummary()

    def construct(self, global_step):
        lr = Gather()(self.learning_rate, global_step, 0)
        self.scalar_collector('lr', lr)
        return lr
```

步骤 5 打开 main.py 文件，该文件中配置了模型训练相关的资源路径，用户无需修改该文件。main.py 脚本启动所需的 data\_url 参数指定了模型训练所需的数据集路径（OBS 中的路径），train\_url 参数指定了模型训练产生的权重文件、summary 日志的保存路径；main 函数中，在实例化 Profiler 后，调用 train.py 中的 train 函数完成模型训练，最后调用 profiler.analyse()完成性能数据收集。

```
import argparse
import os
import numpy as np
from mindspore import context
from mindspore.profiler import Profiler

from train import train

np.random.seed(74)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='MindSpore Bert Training Example')
    parser.add_argument('--data_url', type=str, default="./data", help='Dir of dataset')
    parser.add_argument('--train_url', type=str, default="./output", help='Train Url')
    args = parser.parse_args()

    local_data_path = args.data_url
    local_output_path = args.train_url
    # Summary directory.
    SUMMARY_DIR = os.path.join(args.train_url, "summary")
    # Saved checkpoint path.
    CKPT_DIR = os.path.join(args.train_url, "ckpt")
    # Training dataset path.
    TRAIN_DATASET = os.path.join(args.data_url, "train.mindrecord")
    # Testing dataset path.
    TEST_DATASET = os.path.join(args.data_url, "test.mindrecord")
    # Pre-trained model ckpt path.
    PRETRAINED_MODEL_CKPT_PATH = os.path.join(args.data_url, "bert_zh.ckpt")

    context.set_context(mode=context.GRAPH_MODE, device_target="Ascend")
    profiler = Profiler(output_path=SUMMARY_DIR)
    train(TRAIN_DATASET, TEST_DATASET, PRETRAINED_MODEL_CKPT_PATH, SUMMARY_DIR, CKPT_DIR)
    profiler.analyse()
```

步骤 6 打开 train.py 脚本，该脚本中 train() 函数定义模型训练逻辑，本实验已预置了交叉熵损失函数（Softmax Cross Entropy with Logits）、多项式衰减学习率策略（Polynomial Decay Scheduler）、Momentum 优化器；**注意**：train() 函数中使用的 LossCell 为步骤 4 中在 downstream\_task.py 中创建的 LossCell 类的实例；另外，若将 model.train() 接口中 dataset\_sink\_mode 置为 True 的话，将会导致无法收集模型训练过程中的 Loss 曲线、学习率曲线等信息。



```
def train(train_dataset, test_dataset, pretrained_model_ckpt_path, summary_dir):
    epoch = 10
    batch_size = 32
    # Load training dataset.
    ds_train = load_dataset(train_dataset, batch_size)
    # Load migrated bert model.
    pretrained_model = BertWithPooler()
    param_dict = load_checkpoint(pretrained_model_ckpt_path)
    not_load_params = load_param_into_net(pretrained_model, param_dict)
    assert not not_load_params, "Params is not fully loaded."

    # Define downstream task.
    classifier = EmotionClassifier(pretrained_model)
    classifier.set_train(True)

    # Define loss function here.
    loss_fn = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
    model_with_loss = LossCell(classifier, loss_fn)

    # Define learning rate scheduler and optimizer here.
    learning_rate = polynomial_decay_scheduler(lr=1e-2, min_lr=1e-4, decay_steps=200,
                                              total_update_num=ds_train.get_dataset_size() * epoch,
                                              warmup_steps=100, power=1.2)
    optimizer = nn.Momentum(params=classifier.trainable_params(),
                           learning_rate=LearningRate(learning_rate),
                           momentum=0.9)
    train_model = TrainOneStepCell(model_with_loss, optimizer)

    model = Model(train_model)
    # TODO: Please create SummaryCollector to collect summary data,
    # and add it to model.train() callback interface.
    summary_collector = SummaryCollector(summary_dir=summary_dir, collect_freq=10)
    model.train(epoch, ds_train, callbacks=[summary_collector, LossMonitor(10)], dataset_sink_mode=False)
```

实例化迁移后的预训练 BERT 模型

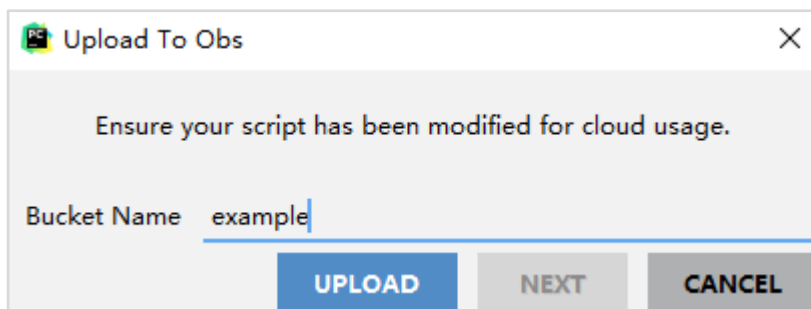
实例化情感分类模型，并将其设置为可训练状态

使用交叉熵损失函数，并使用自定义 LossCell 记录训练过程中的 Loss 曲线

使用多项式衰减策略调整学习率，并使用 Momentum 优化器训练模型

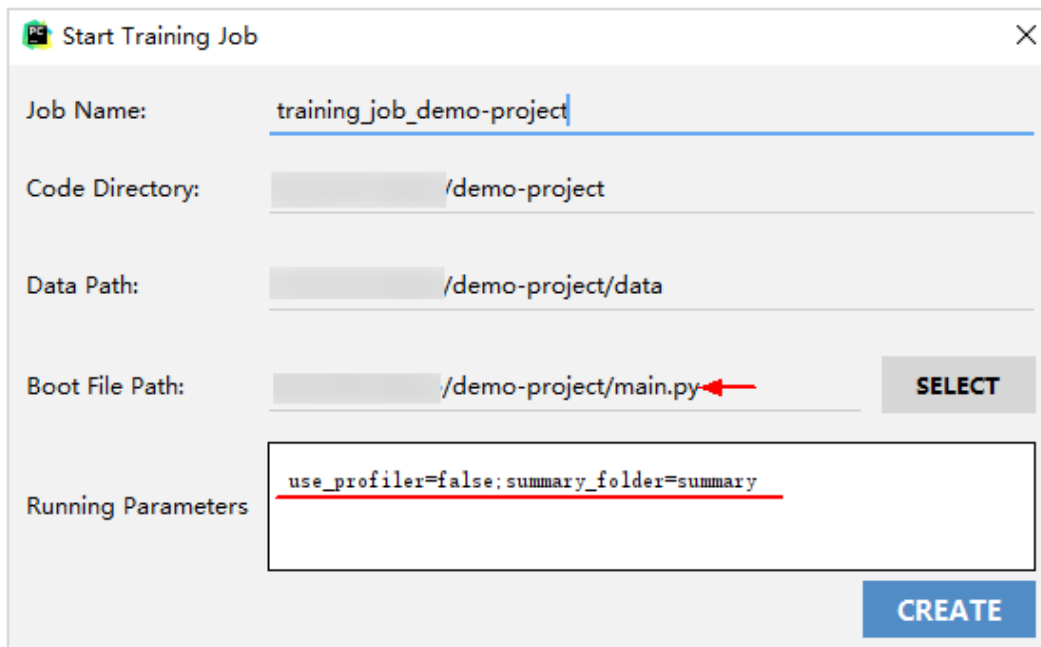
收集训练过程数据，启动训练

步骤 7 点击 PyCharm 任务栏的“MindSpore”菜单中“Cloud”的“Upload”，填入在 2.3.2 节中创建的 OBS 数据桶名称，假设数据桶名称为“example”，点击“Upload”按钮即可将当前项目的脚本、数据集等文件上传至 OBS，稍候会提示“Upload to obs successfully.”；点击“NEXT”进入创建训练作业界面。



步骤 8 创建训练作业界面，可配置模型训练启动脚本（Boot File Path）、源码路径（Code Directory）、训练资源路径（Data Path）以及其他启动参数。如上文所述，这里我们选择 main.py 作为训练启动脚本，点击“CREATE”即可开始训练；若提示“Training job created successfully”，则代表任务创建成功。训练作业默认不收集模型性能数据，如需启动 Profiler 性能数据收集，可在 Running Parameters 中填入“use\_profiler=true”；另外，默认的 summary 数据保存在 summary 目录中，如需修改，可在 Running Parameters 中填入“summary\_folder=summary\_folder”即

可将 summary 数据保存在 summary\_folder 目录下；如需要同时增加 “use\_profiler” 和 “summary\_folder” 参数，参数间用分号隔开。（若在此步骤提示 “ModelArts 未被授权” 问题，需要按照 2.3.2 中指引进行 ModelArts 资源授权）



Start Training Job

Job Name: training\_job\_demo-project

Code Directory: /demo-project

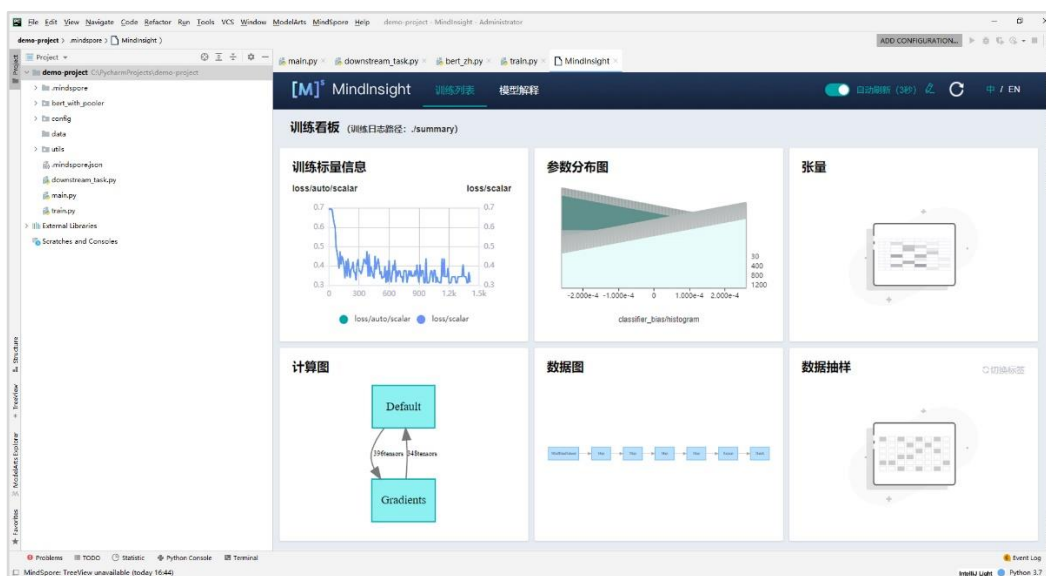
Data Path: /demo-project/data

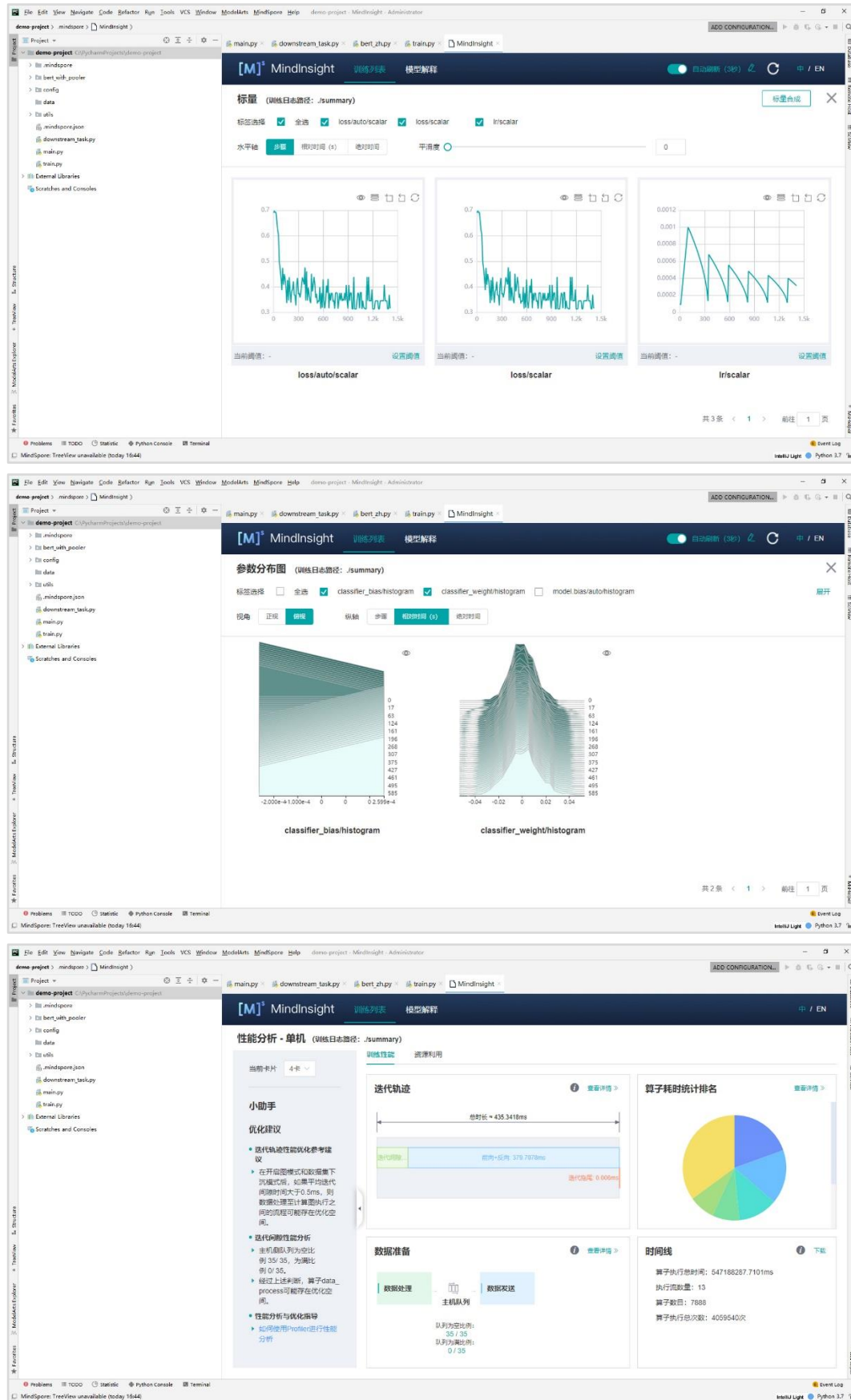
Boot File Path: /demo-project/main.py

Running Parameters: use\_profiler=false; summary\_folder=summary

CREATE

步骤 9 作业创建成功后，会自动打开 MindInsight 可视化界面，用户可通过 MindInsight 界面观测训练过程。由于 MindSpore 产生的训练过程的数据与 MindInsight 同步有一定时间间隔，用户可通过 TrainingJobLog 查看训练已经进行 3 个 Epoch 后，再刷新 MindInsight 界面（刷新方法：关闭当前 MindInsight 窗口，通过 PyCharm 菜单栏的 **MindSpore->MindInsight->Launch** 重新打开页面）。由于高峰期访问量较大或用户网络不佳的原因，自动打开的 MindInsight 窗口若显示 “系统繁忙” 或 “找不到资源” 的情况，也可通过上述方法刷新页面。





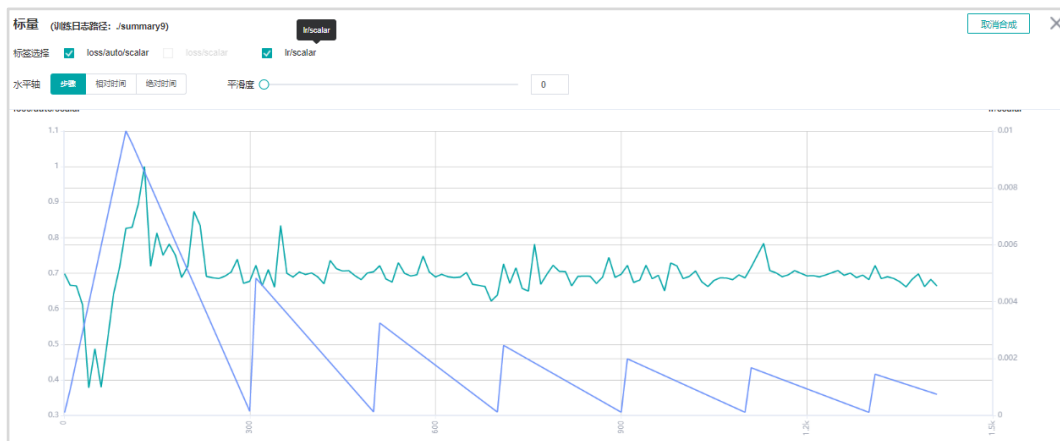
## 2.5 结果验证

步骤 1 点击 PyCharm 右下方的 Training Log，即可查看当前训练作业的训练日志，通过 Ctrl+F 组合键激活“文本搜索”窗口，搜索关键字“Accuracy”即可查看当前作业模型的训练精度，模型推理精度高于 90%说明已完成实验二；若未达到预期精度，请参考 2.4 中步骤 6 的图片内容来调节学习率的方法来使模型精度达到预期。



```
TrainingJobLog Training Log
Q Accuracy
/usr/local/ma/python3.7/lib/python3.7/multiprocessing/semaphore_tracker.py:144: UserWarning: semaphore_tracker:
len(cache))
[WARNING] PRE_ACT(60,python):2021-04-14-16:23:26.525.742 [mindspore/ccsrc/backend/optimizer/somas/somas.cc:764]
[WARNING] PRE_ACT(60,python):2021-04-14-16:23:26.525.773 [mindspore/ccsrc/backend/optimizer/somas/somas.cc:764]
[WARNING] MD(60,python):2021-04-14-16:23:33.896.906 [mindspore/ccsrc/minddata/dataset/util/task.cc:158] Join] Mo
Accuracy: 92.23214285714286 模型精度
optype_name compute_time(ms, per-step) called_times(per-step) percent
Gather 1.0844760263424515 7 0.2
AssignAdd 0.0014951205673758865 1 0.0
OneHot 0.0021500921985815593 1 0.0
Cast 3.302123640324215 607 0.6
BroadcastTo 12.346227879432625 144 2.24
TransData 89.24797531509626 771 16.21
Add 28.380655007092198 346 5.15
Add 0.0000000000000000 1 0.0
```

步骤 2 若 2.3.3 中下载的 train.py 脚本未经修改，出现精度未达标的现象，通过 MindInsight 的标量合成功能观测训练结果，可以发现是由于起始学习率过大，随着学习率的增大，损失值也在回升，导致最终学习率下降模型也难以收敛。可通过调整 train.py 脚本中的起始学习率，重新对模型进行训练，再次观测模型精度。若已修改下载的 train.py 脚本文件，也可以参考如下超参对模型训练，另模型精度达标。



```
def train(train_dataset, test_dataset, pretrained_model_ckpt_path, summary_dir):
    epoch = 10
    batch_size = 32
    # Load training dataset.
    ds_train = load_dataset(train_dataset, batch_size)
    # Load migrated bert model.
    pretrained_model = BertWithPooler()
    param_dict = load_checkpoint(pretrained_model_ckpt_path)
    not_load_params = load_param_into_net(pretrained_model, param_dict)
    assert not not_load_params, "Params is not fully loaded."

    # Define downstream task.
    classifier = EmotionClassifier(pretrained_model)
    classifier.set_train(True)

    # Define loss function here.
    loss_fn = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
    model_with_loss = LossCell(classifier, loss_fn)

    # Define learning rate scheduler and optimizer here. 请将脚本中 lr 的值修改为 1e-3
    learning_rate = polynomial_decay_scheduler(lr=1e-3 min_lr=1e-4, decay_steps=200,
                                              total_update_num=ds_train.get_dataset_size() * epoch,
                                              warmup_steps=100, power=1.2)
    optimizer = nn.Momentum(params=classifier.trainable_params(),
                            learning_rate=LearningRate(learning_rate),
                            momentum=0.9)
    train_model = TrainOneStepCell(model_with_loss, optimizer)

    model = Model(train_model)
    # TODO: Please create SummaryCollector to collect summary data,
    # and add it to model.train() callback interface.
    summary_collector = SummaryCollector(summary_dir=summary_dir, collect_freq=10)
    model.train(epoch, ds_train, callbacks=[summary_collector, LossMonitor(10)], dataset_sink_mode=False)
```

# 3 资源释放

## 3.1 实验说明

在完成所有实验之后，需停止 MindSpore 训练作业，以及 MindInsight 可视化作业，停止方法如下：

- 复制 <https://console.huaweicloud.com/modelarts/?region=cn-north-4&locale=zh-cn#/trainingJobBeta>，并在浏览器中打开，即可查询训练作业状态。若训练作业处于“**运行中**”，可点击“**终止**”按钮停止相应训练作业。
- 复制 <https://console.huaweicloud.com/modelarts/?region=cn-north-4&locale=zh-cn#/trainingJobs>，并在浏览器中打开，切换至“可视化作业”页面，即可查看“**运行中**”的可视化作业，点击“**停止**”按钮，即可停止相应作业。

