华为认证 AI 系列教程

HCIP-AI Solution Architect Vision Transformer图像分类

# 实验手册

版本： 1.0



版权所有 (C) 华为技术有限公司 2024。 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: https://e.huawei.com

# 华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构， 打造的覆盖ICT (Information and Communications Technology，信息通信技术) 全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-AI Solution Architect（Huawei Certified ICT Professional-AI Solution Architect，华为认证人工智能解决方案架构）主要面向于AI解决方案高级工程师、人工智能（大模型）训练工程师、大模型算法工程师和生成式AI工程师。HCIP-AI Solution Architect认证在内容上涵盖了大模型业务流程、自然语言处理算法、大模型算法基础、华为智算解决方案、大模型并行训练框架、预训练、微调和部署方案等内容，具备主流AI框架和并行框架使用能力，大模型微调和部署能力。

华为认证协助您打开行业之窗，开启改变之门，屹立在人工智能世界的潮头浪尖!

| 华为职业认证 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 云平台及云服务 | 云平台及云服务领域 | Cloud Computing | Cloud Service | Big Data | GaussDB | IoT | image |
| 基础软硬件 | 基础软硬件领域 | openEuler | openGauss | HarmonyOS | AI | Kunpeng |
| ICT基础设施 | IT基础设施领域 | Storage | Collaboration | Intelligent Vision | Digital Power | MDC |
| 数通领域 | Datacom | | WLAN | Security | |
|  |  |  |  |  |  |
| 光领域 | Transmission | | Access | | |
| 无线领域 | 5G | |  | LTE | |
|  |  |  |  |  |  |  |

# 前 言

# 简介

本书为 HCIP-AI Solution Architect 认证培训教程，适用于准备参加 HCIP-AI Solution Architect 考试的学员或者希望了解 AI 基础知识、ViT 模型、图像分类模型训练流程等相关人工智能技术的读者。

# 内容描述

本实验指导书共包含 1 个实验，介绍使用 ViT（Vision Transformer）模型，对图片进行分类，

通过学习本课程，学员将掌握如何使用 ViT 模型，对图片进行分类。实验包括以下两个部分：

* 实验介绍，大致介绍实验所需要的实验环境，ViT 模型的相关框架，以及通过这个实验可以掌握训练图片分类模型的相关流程步骤。
* 实验整体流程，具体的训练过程以及相关代码。首先准备实验环境以及具体实验数据，其次是模型框架的搭建，包括 Attention 模块，Multi-Head Self-Attention 模块等具体代码。 最后训练模型，验证模型的效果，以及推理模型。

# 读者知识背景

本课程为华为认证基础课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件:

* 具有基本的人工智能知识背景，同时了解 python 语言的相关基础知识。

# 实验环境说明

本实验采用华为云资源，华为云 ModelArts 购买相应资源。

# 目录

前 言 ..4

简介. ... 4

内容描述. ..4

读者知识背景. ... 4

实验环境说明. ..4

1 ModelArts 开发环境搭建. ..6

1.1 准备实验环境. ... 6

2 Vision Transformer 图片分类. ..8

2.1 实验介绍. ... 8

2.1.1 实验环境与目的. ...8

2.1.2 实验整体流程. ...8

2.2 实验整体流程. ...9

2.2.1 环境准备与数据读取 ... 9

2.2.2 模型搭建. ..10

2.2.3 模型训练与推理. ...16

2.2.4 总结. ...23

2.3 思考题. ...24

# ModelArts 开发环境搭建

# 1.1 准备实验环境

步骤 1 使用华为云 IAM 账号，登录华为云。IAM 账号授课老师会提供。



图1-1 登录华为云账号

步骤 2 在控制台处选择西南-贵阳一区域，并搜索 ModelArts。

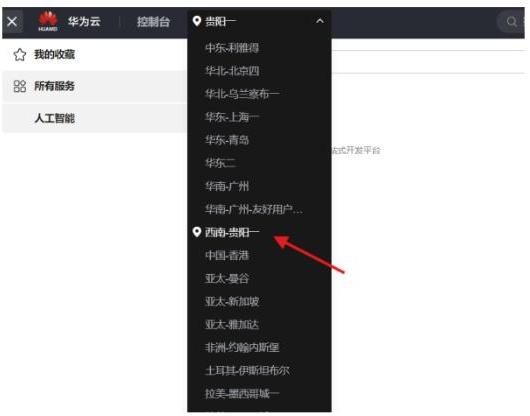
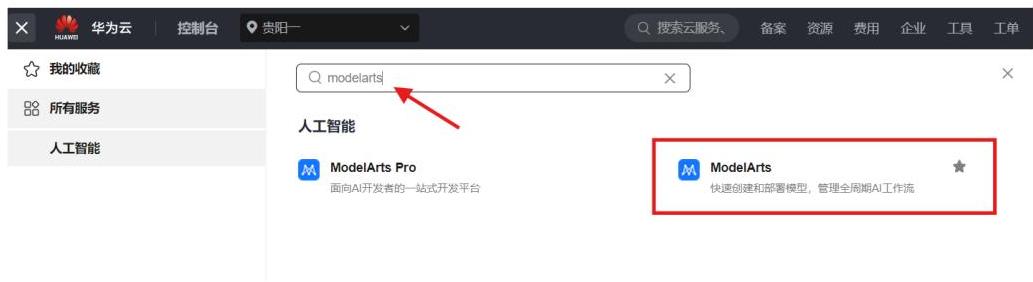


图1-2 选择西南-贵阳一



# 图1-3 搜索 ModelArts 并进入

步骤 3 在 ModelArts 中，选择开发环境，点击 Notebook，并创建对应实验资源的 Notebook。



图1-4 创建 Notebook

# 2 Vision Transformer 图片分类

# 2.1 实验介绍

近些年，随着基于自注意 (Self-Attention) 结构的模型的发展，特别是 Transformer 模型的提出，极大地促进了自然语言处理模型的发展。由于 Transformers 的计算效率和可扩展性，它已经能够训练具有超过 100B 参数的空前规模的模型。

ViT(Vision Transformer)则是自然语言处理和计算机视觉两个领域的融合结晶。在不依赖卷积操作的情况下，依然可以在图像分类任务上达到很好的效果。

本课程将介绍使用 ViT 模型，对图片进行分类，通过学习本课程，学员将掌握如何使用 ViT 模型，对图片进行分类。

# 2.1.1 实验环境与目的

实验采用华为云 ModelArts 平台，具体配置如下:

* 西南-贵阳一
* Notebook 镜像: mindspore\_2.2.0-cann\_7.0.1-py\_3.9-euler\_2.10.7-aarch64-snt9b。

规格：Ascend: 1\*ascend-snt9b(32G)|ARM: 24 核 192GB。

# 实验目的

* 理解 ViT 的基本原理。
* 理解 Self-Attention 的工作原理。
* 理解 Transformer 中不同模块的作用。
* 掌握利用 vit 实现图片分类以及具体流程。

# 2.1.2 实验整体流程

* 数据的下载、读取、预处理。
* ViT 模型的搭建。
* 模型的训练、验证、推理。

# 2.2 实验整体流程

# 2.2.1 环境准备与数据读取

1. 由于实验在华为云 modelarts 上进行，环境已经搭建完毕。

2. 首先我们需要下载本案例的数据集，可通过 http://image-net.org 下载完整的 ImageNet 数据集，本案例应用的数据集是从 ImageNet 中筛选出来的子集。

步骤 1 下载数据集需要用到 download, 首先下载 download 包

! pip install download

由于需要在华为云上下载数据集，以下代码用于访问这些地址时不经过代理，提高访问速度和

避免一些网络访问问题，加快数据的下载。

%env no\_proxy=repo.myhuaweicloud.com, modelarts.cn-north-9.myhuaweicloud.com, obs.cn-north-

9.myhuaweicloud.com, iam.cn-north-9.huaweicloud.com, iam-cache-proxy.cn-north-9.myhuaweicloud.com, dli.cn-

north-9.myhuaweicloud.com,127.0.0.1, localhost,172.16.\*

%env NO\_PROXY=repo.myhuaweicloud.com, modelarts.cn-north-9.myhuaweicloud.com, obs.cn-north-

9.myhuaweicloud.com, iam.cn-north-9.huaweicloud.com, iam-cache-proxy.cn-north-9.myhuaweicloud.com, dli.cn-

north-9.myhuaweicloud.com,127.0.0.1, localhost,172.16.\*

运行下面代码时会自动下载并解压。

from download import download

dataset\_url = "https://mindspore-website.obs.cn-north-

4.myhuaweicloud.com/notebook/datasets/ViT\_imagenet\_dataset.zip"

path "./"

path = download(dataset\_url, path, kind="zip", replace=True)

运行结果如下：

Downloading data from https://mindspore-website.obs.cn-north-

4.myhuaweicloud.com/notebook/datasets/ViT\_imagenet\_dataset.zip (489.1 MB)

file\_sizes: 100% | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 101:16<00:00,

Extracting zip file...

Successfully downloaded / unzipped to . /

运行后，将会得到如下数据集路径。

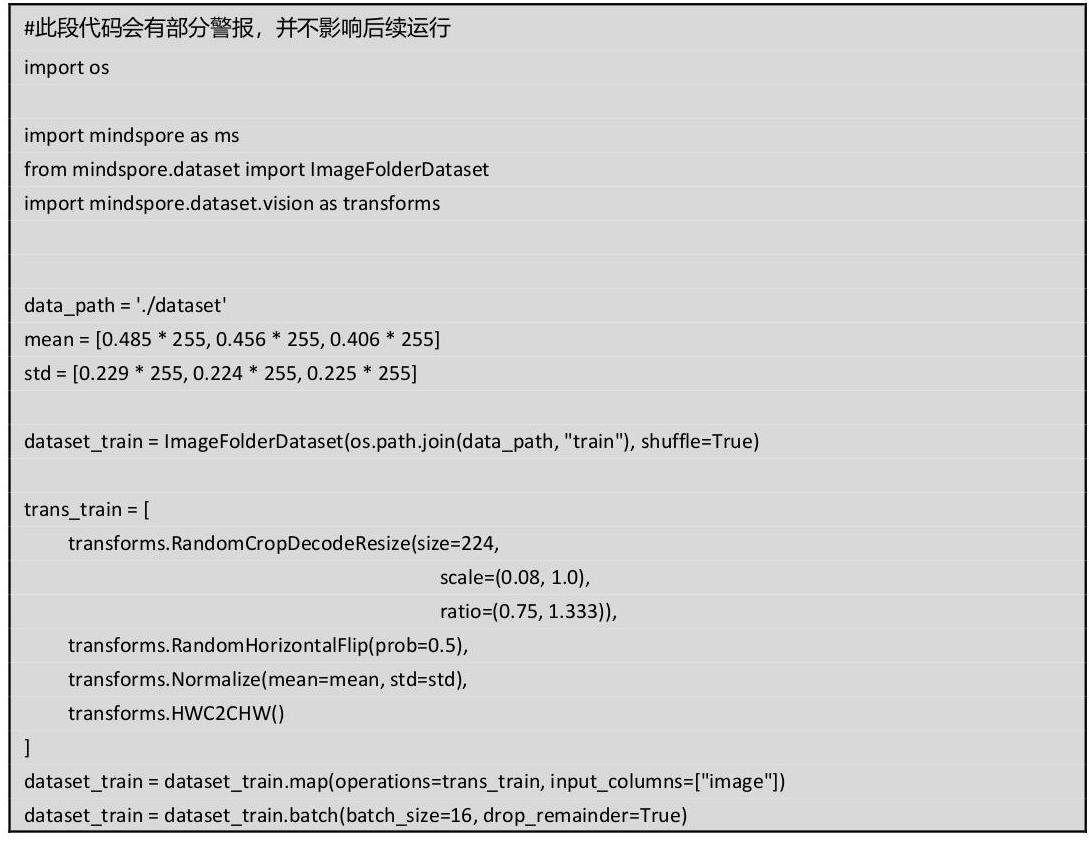
.dataset/

-— ILSVRC2012\_devkit\_t12.tar.gz

* train/
* infer/

L-val/

步骤 2 对下载好的数据集做一些简单的预处理。



# 2.2.2 模型搭建

Transformer 由 Encoder 和 Decoder 结构组成，而 Encoder 和 Decoder 中又包含多头注意力

(Multi-Head Attention) 层, Feed Forward 层, Normaliztion 层, 甚至残差连接 (Residual Connection，图中的“Add”）。不过，其中最重要的结构是多头注意力（Multi-Head Attention） 结构，该结构基于自注意力（Self-Attention）机制，是多个 Self-Attention 的并行组成， Transformer 具体结构如下:

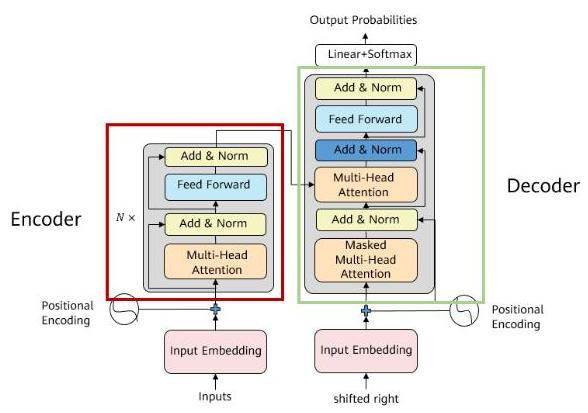


图2-1 Transformer 整体架构

# 步骤 1 Attention 模块

Self-Attention 是 Transformer 中的核心模块，其内容是为输入向量的每个单词学习一个权重。 通过给定一个任务相关的查询向量 Query 向量，计算 Query 和各个 Key 的相似性或者相关性得到注意力分布，即得到每个 Key 对应 Value 的权重系数，然后对 Value 进行加权求和得到最终的 Attention 数值。

from mindspore import nn, ops

class Attention(nn.Cell):

def \_\_init\_\_(self,

dim: int,

num\_heads: int ,

keep\_prob: float ,

attention\_keep\_prob: float = 1.0):

super(Attention, self).\_\_init\_\_()

self.num\_heads = num\_heads

head\_dim num\_heads

self.scale = ms.Tensor(head\_dim \*\* -0.5)

def construct(self, ):

"""Attention construct."""

.shape

self.qkv

qkv = ops.reshape(qkv, (b, n, 3, self.num\_heads, c // self.num\_heads))

ops.transpose

ops.unstack(qkv, axis )

attn self.q\_matmul\_k(q, k)

attn ops.mul(attn, self.scale)

attn self.softmax(attn)

attn = self.attn\_drop(attn)

out self.attn\_matmul\_v(attn, v)

out ops.transpose(out, )

out ops.reshape(out, )

out self.out(out)

out self.out\_drop(out)

return out

# 步骤 2 Transformer Encoder

1. 在了解了 Self-Attention 结构之后，通过与 Feed Forward，Residual Connection 等结构的拼接就可以形成 Transformer 的基础结构，下面代码实现了 Feed Forward，Residual Connection 结构。

from typing import Optional, Dict

class FeedForward(nn.Cell):

def \_\_init\_\_(self,

in\_features: int,

hidden\_features: Optional[int] None,

out\_features: Optional[int] = None,

activation: nn.Cell nn.GELU,

keep\_prob: float ):

super(FeedForward, self).\_\_init\_\_()

out\_features = out\_features or in\_features

hidden\_features hidden\_features or in\_features

self.dense1 = nn.Dense(in\_features, hidden\_features)

self.activation

self.dense2 = nn.Dense(hidden\_features, out\_features)

self.dropout = nn.Dropout(p=1.0-keep\_prob)

def construct(self, ):

"""Feed Forward construct."""

self.dense1(x)

self.activation

self.dropout

self.dense2(x)

self.dropout return

class ResidualCell(nn.Cell):

def \_\_init\_\_(self, cell):

super(ResidualCell, self).\_\_init\_\_()

self.cell cell

def construct(self, ):

"""ResidualCell construct."""

return self.cell(x) + x

2. 将 Transformer Encoder 结构和一个多层感知器 (MLP) 结合，就构成了 ViT 模型的 backbone 部分。



def construct(self, ):

"""Transformer construct."""

return self.layers(x)

# 步骤 3 ViT 模型的输入

传统的 Transformer 结构主要用于处理自然语言领域的词向量（Word Embedding or Word

Vector），词向量与传统图像数据的主要区别在于，词向量通常是一维向量进行堆叠，而图片则是二维矩阵的堆叠。

在 ViT 模型中:

1. 通过将输入图像在每个 channel 上划分为 个 patch，这一步是通过卷积操作来完成的，当然也可以人工进行划分，但卷积操作可以达到这一目的同时，还可以进行一次额外的数据处理；例如一幅输入 的图像，首先经过卷积处理得到 个 patch，那么每一个 patch 的大小就是 14 x 14。

2. 再将每一个 patch 的矩阵拉伸成为一个一维向量，从而获得了近似于向量堆叠的效果。上一步得到的 的 patch 就转换为长度为 196 的向量。

# 步骤 4 完整 VIT

class PatchEmbedding(nn.Cell):

MIN\_NUM\_PATCHES

def \_\_init\_\_(self,

image\_size: int ,

patch\_size: int ,

embed\_dim: int ,

input\_channels: int ):

super(PatchEmbedding, self).\_\_init\_\_()

self.image\_size image\_size

self.patch\_size patch\_size

self.num\_patches (image\_size // patch\_size) \*\* 2

self.conv = nn.Conv2d(input\_channels, embed\_dim, kernel\_size=patch\_size, stride=patch\_size,

has\_bias=True)

def construct(self, ):

"""Path Embedding construct."""

.shape

ops.reshape

ops.transpose

return

from mindspore.common.initializer import Normal

# class ViT(nn.Cell):

from mindspore.common.initializer import initializer

from mindspore import Parameter

def init(init\_type, shape, dtype, name, requires\_grad):

"""Init."""

initial = initializer(init\_type, shape, dtype).init\_data()

return Parameter(initial, name=name, requires\_grad=requires\_grad)

def \_\_init\_\_(self,

image\_size: int ,

input\_channels: int ,

patch\_size: int ,

embed\_dim: int ,

num\_layers: int ,

num\_heads: int ,

mlp\_dim: int ,

keep\_prob: float ,

attention\_keep\_prob: float ,

drop\_path\_keep\_prob: float = 1.0,

activation: nn.Cell = nn.GELU,

norm: Optional[nn.Cell] = nn.LayerNorm,

pool: str ’cls’) None:

super(ViT, self).\_\_init\_\_()

self.patch\_embedding PatchEmbedding(image\_size=image\_size,

patch\_size=patch\_size,

embed\_dim=embed\_dim,

input\_channels=input\_channels)

num\_patches = self.patch\_embedding.num\_patches

self.cls\_token = init(init\_type=Normal(sigma=1.0),

shape ,

dtype=ms.float32,

name=’cls’,

requires\_grad=True)

self.pos\_embedding = init(init\_type=Normal(sigma=1.0),

shape=(1, num\_patches + 1, embed\_dim),

dtype=ms.float32,

name=’pos\_embedding’,

requires\_grad=True)

self.pool pool

self.pos\_dropout = nn.Dropout(p=1.0-keep\_prob)

self.norm = norm((embed\_dim,))

self.transformer = TransformerEncoder(dim=embed\_dim,

keep\_prob=keep\_prob,

activation activation,

self.dropout = nn.Dropout(p=1.0-keep\_prob)

self.dense = nn.Dense(embed\_dim, num\_classes)

def construct(self, ):

"""ViT construct."""

self.patch\_embedding

cls\_tokens = ops.tile(self.cls\_token.astype(x.dtype), (x.shape[0], 1, 1))

ops.concat

self.pos\_embedding

self.pos\_dropout

self.transformer

self.norm

if self.training:

self.dropout

self.dense

return

# 2.2.3 模型训练与推理

步骤 1 模型训练

模型开始训练前，需要设定损失函数，优化器，回调函数等。

完整训练 ViT 模型需要很长的时间，实际应用时建议根据项目需要调整 epoch\_size，当正常输出每个 Epoch 的 step 信息时，意味着训练正在进行，通过模型输出可以查看当前训练的 loss 值和时间等指标。

#此段代码会出现部分警报，并不影响运行

from mindspore.nn import LossBase

from mindspore.train import LossMonitor, TimeMonitor, CheckpointConfig, ModelCheckpoint

from mindspore import train

# define super parameter

epoch\_size

momentum

num\_classes

resize

step\_size = dataset\_train.get\_dataset\_size()

# construct model

network

# load ckpt

ViT\_url = "https://download.mindspore.cn/vision/classification/ViT\_b\_16\_224.ckpt"

path = "./ckpt/ViT\_b\_16\_224.ckpt"

ViT\_path download(ViT\_url, path, replace=True)

param\_dict = ms.load\_checkpoint(ViT\_path)

ms.load\_param\_into\_net(network, param\_dict)

# define learning rate

lr = nn.cosine\_decay\_lr(min\_lr=float(0),

max\_lr=0.00005,

total\_step=epoch\_size \* step\_size,

step\_per\_epoch=step\_size,

decay\_epoch=10)

# define optimizer

network\_opt nn.Adam(network.trainable\_params(), lr, momentum)

# define loss function

class CrossEntropySmooth(LossBase):

"""CrossEntropy."""

def \_\_init\_\_(self, sparse=True, reduction=’mean’, smooth\_factor=0., num\_classes=1000):

super(CrossEntropySmooth, self).\_init\_\_()

self.onehot ops.OneHot()

self.sparse sparse

self.on\_value ms.Tensor

self.off\_value ms.Tensor

self.ce = nn.SoftmaxCrossEntropyWithLogits(reduction=reduction)

def construct(self, logit, label):

if self.sparse:

label = self.onehot(label, ops.shape(logit)[1], self.on\_value, self.off\_value)

loss self.ce(logit, label)

return loss

network\_loss = CrossEntropySmooth(sparse=True, reduction "mean",

smooth\_factor ,

num\_classes=num\_classes)

# set checkpoint

ckpt\_config = CheckpointConfig(save\_checkpoint\_steps=step\_size, keep\_checkpoint\_max=100)

ckpt\_callback = ModelCheckpoint(prefix=’ViT\_b\_16’, directory=’./ViT’, config=ckpt\_config)

# initialize model

# "Ascend + mixed precision" can improve performance

ascend\_target = (ms.get\_context("device\_target") == "Ascend")

model = train.Model(network, loss\_fn=network\_loss, optimizer=network\_opt, metrics= {"acc" }, amp\_level="O2")

# train model

model.train(epoch\_size,

dataset\_train,

callbacks=[ckpt\_callback, LossMonitor(125), TimeMonitor(125)],

dataset\_sink\_mode=False,)

# 运行结果如下:

Downloading data from https://download-mindspore.osinfra.cn/vision/classification/ViT\_b\_16\_224.ckpt (330.2 MB)

| file\_sizes: 100%346M/346M [00:30<00:00, |
| --- |
|  |
| Successfully downloaded file to ./ckpt/ViT\_b\_16\_224.ckpt |
| epoch: 1 step: 125, loss is 6.892313 |
| Train epoch time: 30051.419 ms, per step time: 240.411 ms |
| epoch: 2 step: 125, loss is 6.945095 |
| Train epoch time: 26493.501 ms, per step time: 211.948 ms |
| epoch: 3 step: 125, loss is 6.8970213 |
| Train epoch time: 26630.793 ms, per step time: 213.046 ms |
| epoch: 4 step: 125, loss is 6.902484 |
| Train epoch time: 27683.166 ms, per step time: 221.465 ms |
| epoch: 5 step: 125, loss is 6.9667454 |
| Train epoch time: 26773.869 ms, per step time: 214.191 ms |
| epoch: 6 step: 125, loss is 6.9458294 |
| Train epoch time: 26669.241 ms, per step time: 213.354 ms |
| epoch: 7 step: 125, loss is 6.9281883 |
| Train epoch time: 26672.292 ms, per step time: 213.378 ms |
| epoch: 8 step: 125, loss is 6.968166 |
| Train epoch time: 26773.232 ms, per step time: 214.186 ms |
| epoch: 9 step: 125, loss is 6.893861 |
| Train epoch time: 26597.878 ms, per step time: |
| epoch: 10 step: 125, loss is 6.9039083 |
| Train epoch time: 26388.392 ms, per step time: |

步骤 2 模型验证

与训练过程相似，首先进行数据增强，然后定义 ViT 网络结构，加载预训练模型参数。随后设置损失函数，评价指标等，编译模型后进行验证。本实验采用了业界通用的评价标准 Top\_1\_Accuracy 和 Top\_5\_Accuracy 评价指标来评价模型表现。

在本实验中，这两个指标代表了在输出的 1000 维向量中，以最大值或前 5 的输出值所代表的

类别为预测结果时，模型预测的准确率。这两个指标的值越大，代表模型准确率越高。

#ImageFolderDataset 主要用于读取数据集

dataset\_val = ImageFolderDataset(os.path.join(data\_path, "val"), shuffle=True)

trans\_val

transforms.Decode(),

transforms.Resize ,

transforms.CenterCrop(224),

transforms.Normalize(mean=mean, std=std),

transforms.HWC2CHW()

1

dataset\_val = dataset\_val.map(operations=trans\_val, input\_columns=["image"])

dataset\_val = dataset\_val.batch(batch\_size=16, drop\_remainder=True)

# construct model

network

# load ckpt

param\_dict = ms.load\_checkpoint(ViT\_path)

ms.load\_param\_into\_net(network, param\_dict)

#CrossEntropySmooth 是损失函数实例化接口

network\_loss = CrossEntropySmooth(sparse=True,

reduction "mean",

smooth\_factor ,

num\_classes=num\_classes)

# define metric

eval\_metrics = {’Top\_1\_Accuracy’: train.Top1CategoricalAccuracy(),

’Top\_5\_Accuracy’: train.Top5CategoricalAccuracy() }

model = train.Model(network, loss\_fn=network\_loss, optimizer=network\_opt, metrics=eval\_metrics,

amp\_level="O2")

# evaluate model

result model.eval(dataset\_val)

print(result)

运行结果如下:

{’Top\_1\_Accuracy’: 0.7495, ’Top\_5\_Accuracy’: 0.928 }

# 步骤 3 模型推理

在进行模型推理之前，首先要定义一个对推理图片进行数据预处理的方法。该方法可以对我们的推理图片进行 resize 和 normalize 处理，这样才能与我们训练时的输入数据匹配。

本实验采用了一张 Doberman 的图片作为推理图片来测试模型表现，期望模型可以给出正确的预测结果。

dataset\_infer = ImageFolderDataset(os.path.join(data\_path, "infer"), shuffle=True)

trans\_infer

transforms.Decode(),

transforms.Resize([224, 224]),

transforms.Normalize(mean=mean, std=std),

transforms.HWC2CHW()

1

dataset\_infer = dataset\_infer.map(operations=trans\_infer,

input\_columns=["image"],

num\_parallel\_workers=1)

dataset\_infer = dataset\_infer.batch(1)

接下来，我们将调用模型的 predict 方法进行模型推理。

在推理过程中，通过 index2label 就可以获取对应标签，再通过自定义的 show\_result 接口将结果写在对应图片上。

import os

import pathlib

import cv2

import numpy as np

from PIL import Image

from enum import Enum

from scipy import io

class Color(Enum):

"""dedine enum color."""

red

green

blue

cyan

yellow

magenta

white

black

if isinstance(color, Color):

if isinstance(color, tuple):

assert len(color)

for channel in color:

assert channel

return color

if isinstance(color, int):

assert color

return color, color, color

if isinstance(color, np.ndarray):

assert color.ndim and color.size

assert np.all((color >= 0) & (color <= 255))

color color.astype(np.uint8)

return tuple(color)

raise TypeError(f’Invalid type for color: {type(color) }’)

def imread(image, mode=None):

"""imread."""

if isinstance(image, pathlib.Path):

image

if isinstance(image, np.ndarray):

pass

elif isinstance(image, str):

check\_file\_exist(image)

image Image.open(image)

if mode:

image np.array(image.convert(mode))

else:

raise TypeError("Image must be a ‘ndarray‘, ‘str‘ or Path object.")

return image

def imwrite(image, image\_path, auto\_mkdir=True):

"""imwrite."""

if auto\_mkdir:

dir\_name = os.path.abspath(os.path.dirname(image\_path))

if dir\_name != ":

dir\_name os.path.expanduser(dir\_name)

os.makedirs(dir\_name, mode=777, exist\_ok=True)

image Image.fromarray(image)

image.save(image\_path)

def imshow(img, win\_name=”, wait\_time=0):

"""imshow"""

cv2.imshow(win\_name, imread(img))

if wait\_time #prevent from hanging if windows was closed

while True:

ret cv2.waitKey

closed = cv2.getWindowProperty(win\_name, cv2.WND\_PROP\_VISIBLE) < 1

# if user closed window or if some key pressed

if closed or ret :

break

else:

ret cv2.waitKey(wait\_time)

# def show\_result(img: str,

result: Dict[int, float],

text\_color: str = ’green’,

font\_scale: float ,

row\_width: int ,

show: bool False,

win\_name: str = ",

wait\_time: int ,

out\_file: Optional[str] = None) -> None:

"""Mark the prediction results on the picture."""

, row\_width

text\_color color\_val(text\_color)

for , v in result.items():

if isinstance(v, float):

label\_text

cv2.putText(img, label\_text, (x, y), cv2.FONT\_HERSHEY\_COMPLEX,

font\_scale, text\_color)

row\_width

if out\_file:

show False

imwrite(img, out\_file)

# # Read data for inference

if show:

imshow(img, win\_name, wait\_time)

def index2label():

"""Dictionary output for image numbers and categories of the ImageNet dataset."""

metafile = os.path.join(data\_path, "ILSVRC2012\_devkit\_t12/data/meta.mat")

meta io.loadmat(metafile, squeeze\_me=True)[’synsets’]

nums\_children = list(zip(\*meta))[4]

meta [meta[idx] for idx, num\_children in enumerate(nums\_children) if num\_children ]

\_, wnids, classes = list(zip(\*meta))[:3]

clssname

wnid2class wnid: clss for wnid, clss in zip(wnids, clssname)

wind2class\_name = sorted(wnid2class.items(), key=lambda x: x[0])

mapping

for index, (\_, class\_name) in enumerate(wind2class\_name):

mapping[index] = class\_name[0]

return mapping

for i, image in enumerate(dataset\_infer.create\_dict\_iterator(output\_numpy=True)):

image image

image ms.Tensor(image)

prob model.predict(image)

label

mapping index2label()

output mapping

print(output)

show\_result(img="./dataset/infer/n01440764/ILSVRC2012\_test\_00000279.JPEG",

result output,

out\_file="./dataset/infer/ILSVRC2012\_test\_00000279.JPEG")

运行结果如下:

{236: ’Doberman’ }

推理过程完成后，在推理文件夹下可以找到对应图片的推理结果，可以看出预测结果是

Doberman，与期望结果相同，验证了模型的准确性。

# 2.2.4 总结

本实验完成了一个 ViT 模型在 ImageNet 数据上进行训练，验证和推理的过程，其中，对关键

的 ViT 模型结构和原理作了讲解。通过学习本实验，理解源码可以帮助学员掌握 Multi-Head Attention，TransformerEncoder，pos\_embedding 等关键概念，如果要详细理解 ViT 的模型原理，建议基于源码更深层次的详细阅读。

# 2.3 思考题

在完整 ViT 模型的最后，只有一层全连接，为什么没有加入 Softmax 激活函数?

答案: 因为在定义损失函数时，采用 nn.SoftmaxCrossEntropyWithLogits 接口，这个接口在计算损失函数前，要先经过一层 Softmax 函数。

华为认证 AI 系列教程

HCIP-AI Solution Architect

# 大模型并行训练框架 DeepSpeed -Baichuan2-7B 实验手册

版本：1.0



版权所有 (C) 华为技术有限公司 2024。 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: https://e.huawei.com

# 华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构， 打造的覆盖ICT (Information and Communications Technology，信息通信技术) 全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-AI Solution Architect（Huawei Certified ICT Professional-AI Solution Architect，华为认证人工智能解决方案架构）主要面向于AI解决方案高级工程师、人工智能（大模型）训练工程师、大模型算法工程师和生成式AI工程师。HCIP-AI Solution Architect认证在内容上涵盖了大模型业务流程、自然语言处理算法、大模型算法基础、华为智算解决方案、大模型并行训练框架、预训练、微调和部署方案等内容，具备主流AI框架和并行框架使用能力，大模型微调和部署能力。

华为认证协助您打开行业之窗，开启改变之门，屹立在人工智能世界的潮头浪尖!

| 华为职业认证 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 云平台及云服务 | 云平台及云服务领域 | Cloud Computing | Cloud Service | Big Data | GaussDB | IoT | image |
| 基础软硬件 | 基础软硬件领域 | openEuler | openGauss | HarmonyOS | AI | Kunpeng |
| ICT基础设施 | IT基础设施领域 | Storage | Collaboration | Intelligent Vision | Digital Power | MDC |
| 数通领域 | Datacom | | WLAN | Security | |
|  |  |  |  |  | |
| 光领域 | Transmission | |  | Access | |
| 无线领域 | 5G | | | LTE | |
|  |  |  |  |  |  | |

# 前 言

# 简介

本书为 HCIP-AI Solution Architect 认证培训教程，适用于准备参加 HCIP-AI Solution Architect 考试的学员。

# 内容描述

本实验指导书共包含 2 个实验，包括开发环境搭建和 PyTorch 环境准备。具体两个实验为：

* 实验一为模型推理实验。
* 实验二为 LoRA 微调实验。

# 实验环境说明

华为云 ModelArts 购买相应资源。

# 目录

前 言 ..3

简介. ... 3

内容描述. ..3

实验环境说明. ... 3

1 ModelArts 开发环境搭建. ..6

1.1 进入华为云 ModelArts 控制台 ... 6

1.2 创建 Notebook 训练作业. .. 6

1.3 启动 Notebook 进入开发环境. ...8

1.4 停止 Notebook 训练作业. ...9

2 Baichuan2-7B 大模型实操-DeepSpeed. .11

2.1 实验介绍. ...11

2.1.1 关于本实验. ...11

2.1.2 实验目的. ...11

2.1.3 实验流程. ...11

2.1.4 实验环境. ... 12

2.2 环境准备. ...12

2.2.1 安装 PyTorch. ...12

2.2.2 安装 torch-npu 插件. ... 12

2.3 模型推理. ...12

2.3.1 下载 HuggingFace 模型文件. ...12

2.3.2 新建推理脚本. ...13

2.3.3 执行推理脚本. ...14

2.4 LoRA 微调. ...15

2.4.1 获取源码. ...15

2.4.2 安装依赖库. ..16

2.4.3 新建 lora 微调启动脚本. ...16

2.4.4 执行微调脚本. ...18

2.5 实验总结. ...18

2.6 思考题. ...19

# 1.1 进入华为云 ModelArts 控制台

在华为云 ModelArts 主页（https://www.huaweicloud.com/product/modelarts.html），点击“控制台”进入 ModelArts 的管理页面。

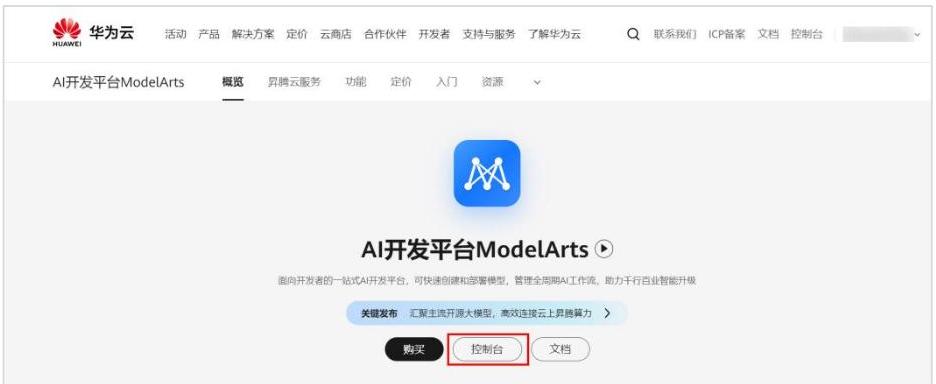


图1-1 华为云 ModelArts 主页

# 1.2 创建 Notebook 训练作业

控制台区域选择“西南-贵阳一”，在左侧菜单栏中选择“开发环境”的“Notebook”，点击进入 Notebook 创建页面。

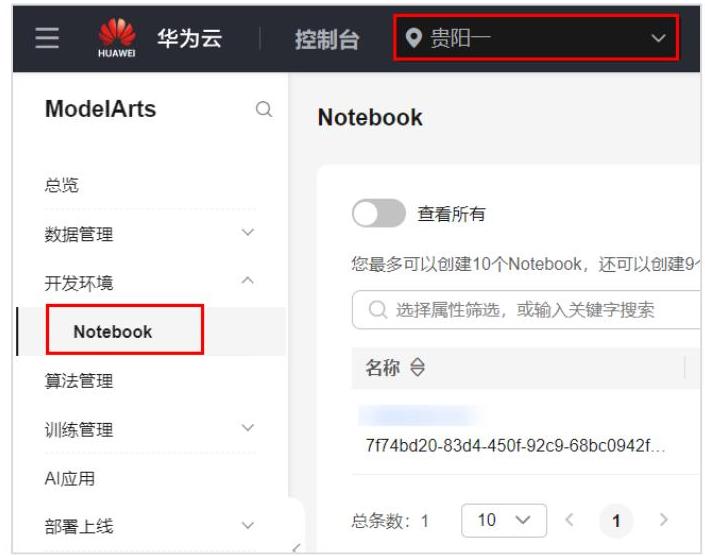


图1-2 ModelArts 控制台

点击“创建”按钮，创建一个新的 Notebook，其配置如下：

* 名称: 自定义。
* 自动停止: 建议 2 小时。
* 镜像：mindspore\_2.2.0-cann\_7.0.1-py\_3.9-euler\_2.10.7-aarch64-snt9b。
* 资源池: 公共资源池。
* 类型: ASCEND。
* 规格: Ascend: 1\*ascend-snt9b(64G)|ARM: 24 核 192GB。
* 存储配置: 云硬盘 EVS。
* 磁盘规格: 1000GB。

如图所示:



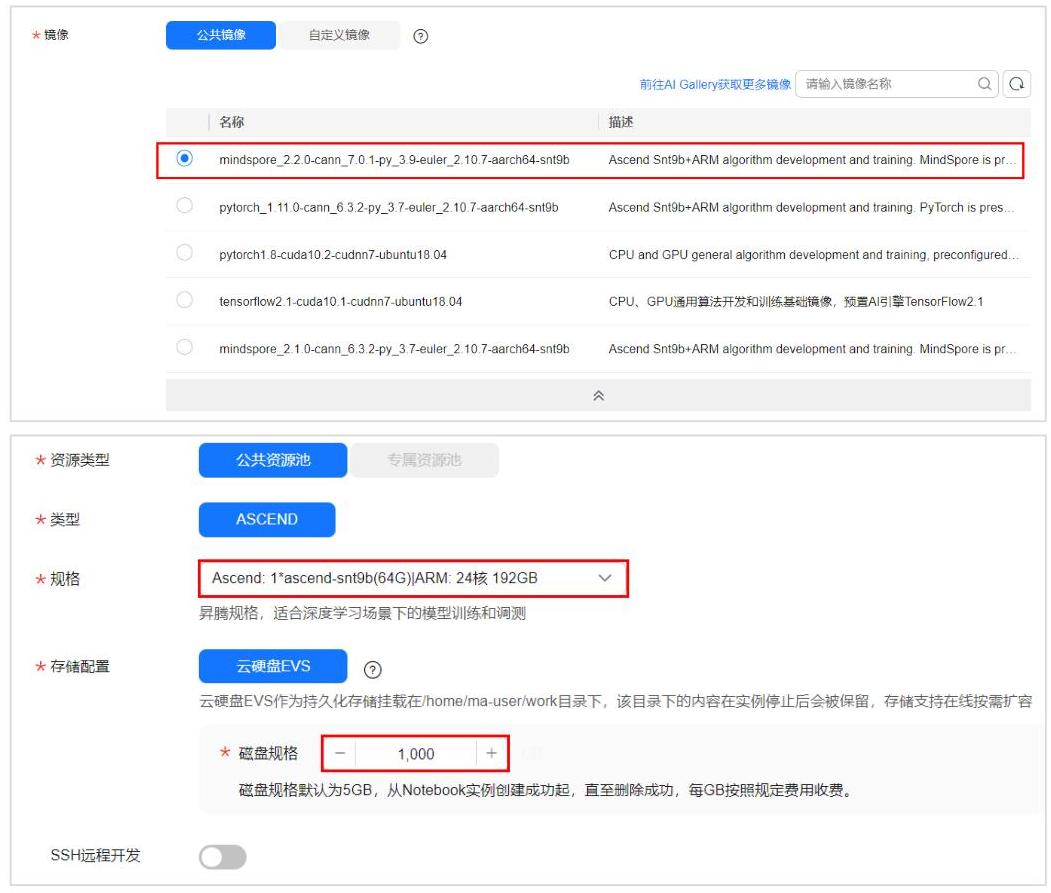


图1-3 Notebook 配置

配置完成之后“立即创建”，规格确认无误之后“提交”。

# 1.3 启动 Notebook 进入开发环境

当上一步创建好 Notebook 状态显示为 “运行中” 时，在右侧 “操作” 栏中 “打开”，即可进入在线编程页面。



图1-4 打开 Notebook 环境

可以在此页面创建或编辑项目，如图所示：

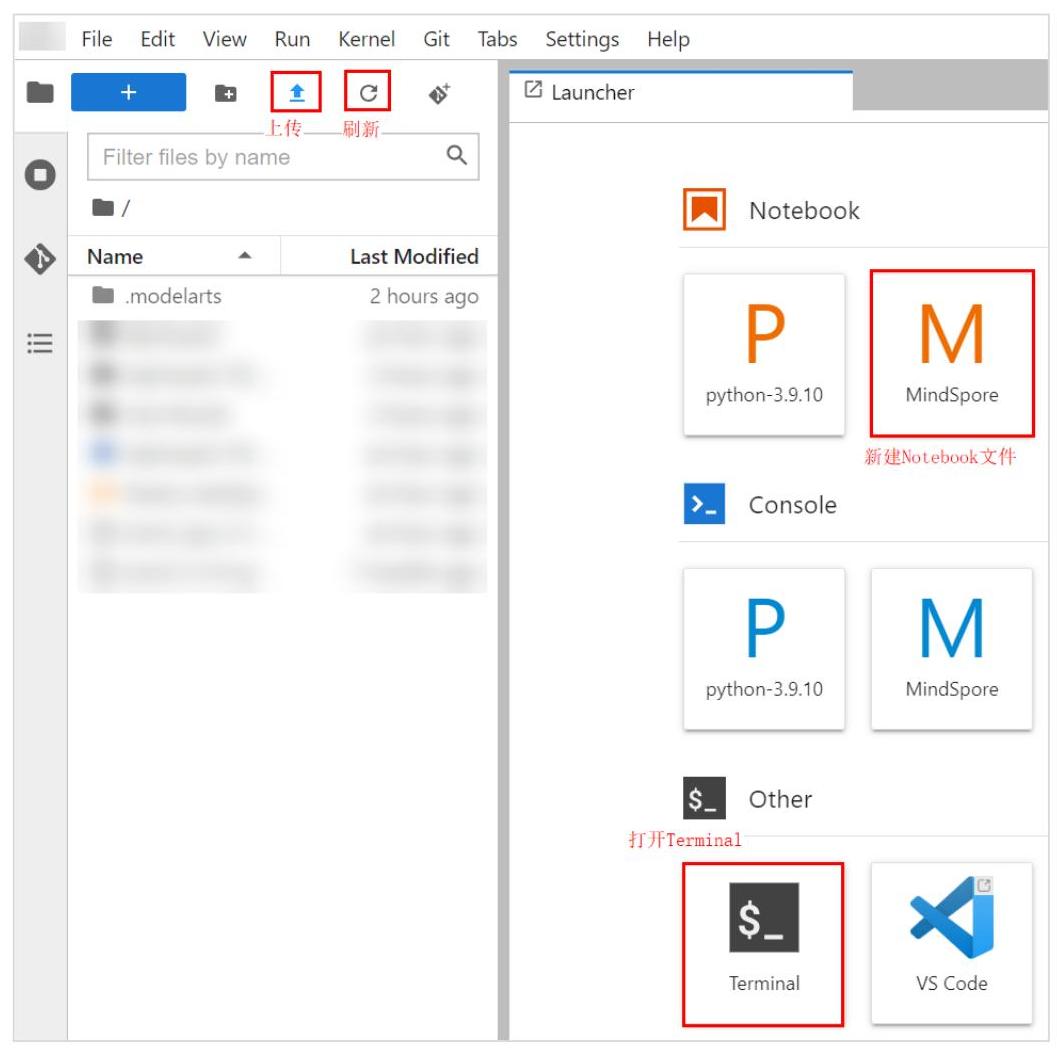


图1-5 页面操作

\*注意：Notebook 环境内上传、创建和编辑的文件均在/home/ma-user/work 目录下。

# 1.4 停止 Notebook 训练作业

实验完成之后，请及时关闭 Notebook 训练作业，避免产生不必要的资源浪费。

登录华为云 ModelArts 控制台，在“操作”栏选择“停止”操作。 如下图所示:



# 图1-6 及时停止 Notebook

至此，训练用的线上 Notebook 环境搭建完成。

# C Baichuan2-7B 大模型实操-DeepSpeed

# 2.1 实验介绍

# 2.1.1 关于本实验

本实验介绍如何在 ModelArts 平台上使用 HuggingFace Transformers 大模型套件和 DeepSpeed 加速库进行 Baichuan2-7B 模型的推理和微调。

DeepSpeed 是一个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。它通过多种技术手段来加速训练，包括模型并行化、梯度累积、动态精度缩放、本地模式混合精度等。DeepSpeed 还提供了一些辅助工具，如分布式训练管理、内存优化和模型压缩等，以帮助开发者更好地管理和优化大规模深度学习训练任务。

Baichuan 2 是百川智能推出的新一代开源大语言模型，采用 2.6 万亿 Tokens 的高质量语料训

练。Baichuan 2 在多个权威的中文、英文和多语言的通用、领域 benchmark 上取得同尺寸最佳的效果。本实验使用 Baichuan2-7B-Chat 版本模型。

# 2.1.2 实验目的

* 掌握 DeepSpeed 与 Huggingface transformers 库结合使用;
* 掌握 Baichuan2-7B 大模型的推理和微调流程和方法。

# 2.1.3 实验流程

* 环境准备;
* 获取 HuggingFace 模型文件;
* 执行模型推理;
* 获取 Baichuan2 源码；
* 执行模型微调。

# 2.1.4 实验环境

表2-1 实验清单

| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| --- | --- | --- | --- | --- |
| Baichuan2-7B大 模型实操 | Baichuan2-7B- Chat模型微调& 推理 | 进阶 | pytorch-2.1.0 transformers-4.36.0 deepspeed-0.12.4 | ModelArts Ascend |

# 2.2 环境准备

# 2.2.1 安装 PyTorch

进入 Notebook 开发环境，打开命令行终端，执行以下命令下载并安装 CPU 版本 PyTorch，

PyTorch 版本为 2.1.0。

wget https://download.pytorch.org/whl/cpu/torch-2.1.0-cp39-cp39-

manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl

pip install torch-2.1.0-cp39-cp39-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl

# 2.2.2 安装 torch-npu 插件

命令行终端执行以下命令下载并安装对应版本 torch-npu 插件。

| wget https://gitee.com/ascend/pytorch/releases/download/v5.0.0-pytorch2.1.0/torch\_npu-2.1.0-cp39-cp39-

manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl

pip install torch\_npu-2.1.0-cp39-cp39-manylinux\_2\_17\_aarch64.manylinux2014\_aarch64.whl

# 2.3 模型推理

# 2.3.1 下载 HuggingFace 模型文件

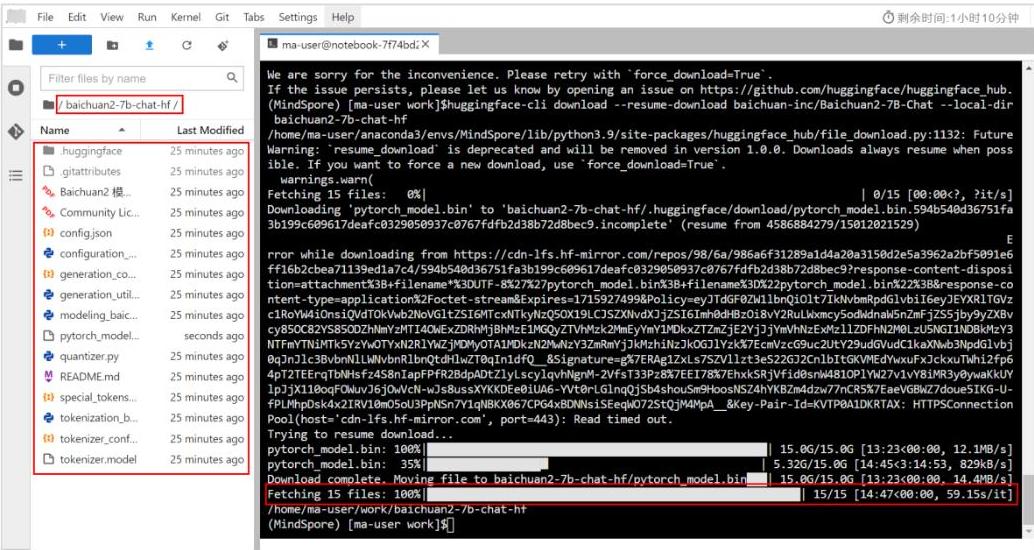
命令行终端执行以下命令，下载 baichuan2-7b-chat 模型文件。

pip install -U huggingface\_hub -i https://pypi.tuna.tsinghua.edu.cn/simple

export HF\_ENDPOINT=https://hf-mirror.com

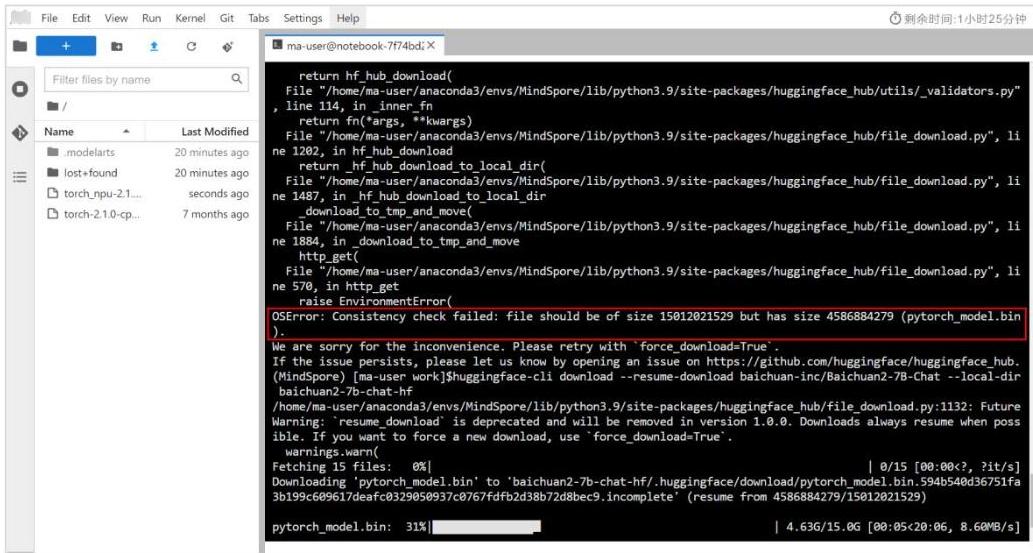
huggingface-cli download - resume-download baichuan-inc/Baichuan2-7B-Chat -local-dir baichuan2-7b-chat-hf

模型大小约 15G，下载时间约 20 分钟。下载完成，如下所示：



# 图2-1 下载模型文件

如果模型权重文件下载过程中报错提示 OSError，重新运行上述 huggingface-cli download 命令恢复下载即可。



# 图2-2 模型文件下载报错

# 2.3.2 新建推理脚本

在左侧文件浏览窗口点击鼠标右键，选择“New File” 新建脚本文件，重命名为 baichuan2-7b-chat-test.py，输入脚本内容如下:

import torch

from transformers import AutoModelForCausalLM, AutoTokenizer

from transformers.generation.utils import GenerationConfig

tokenizer = AutoTokenizer.from\_pretrained("./baichuan2-7b-chat-hf", use\_fast=False, trust\_remote\_code=True) model = AutoModelForCausalLM.from\_pretrained("./baichuan2-7b-chat-hf", torch\_dtype=torch.float16, trust\_remote\_code=True).npu()

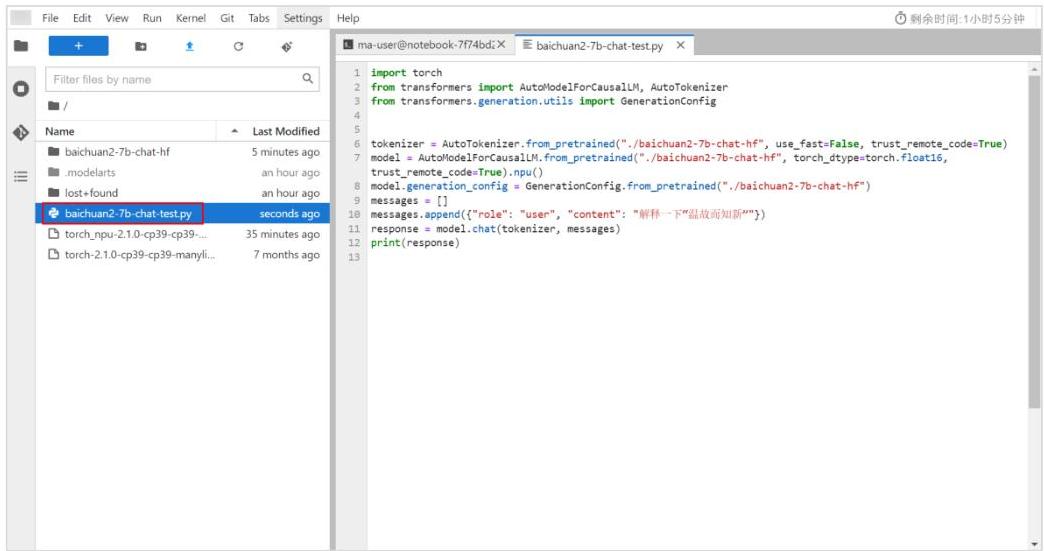
model.generation\_config = GenerationConfig.from\_pretrained("./baichuan2-7b-chat-hf")

messages

messages.append( {"role": "user", "content": "解释一下"温故而知新"" })

response model.chat(tokenizer, messages)

print(response)



# 图2-3 新建推理脚本

目前 transformers 库已原生支持昇腾 NPU，推理代码只需少量修改即可运行在昇腾环境上。

# 2.3.3 执行推理脚本

步骤 1 安装依赖

命令行终端执行以下命令：

pip install transformers==4.36.0 accelerate==0.25.0

注意: 需要安装 accelerate 库才能正常加载模型并运行在昇腾 npu 上。transformers、

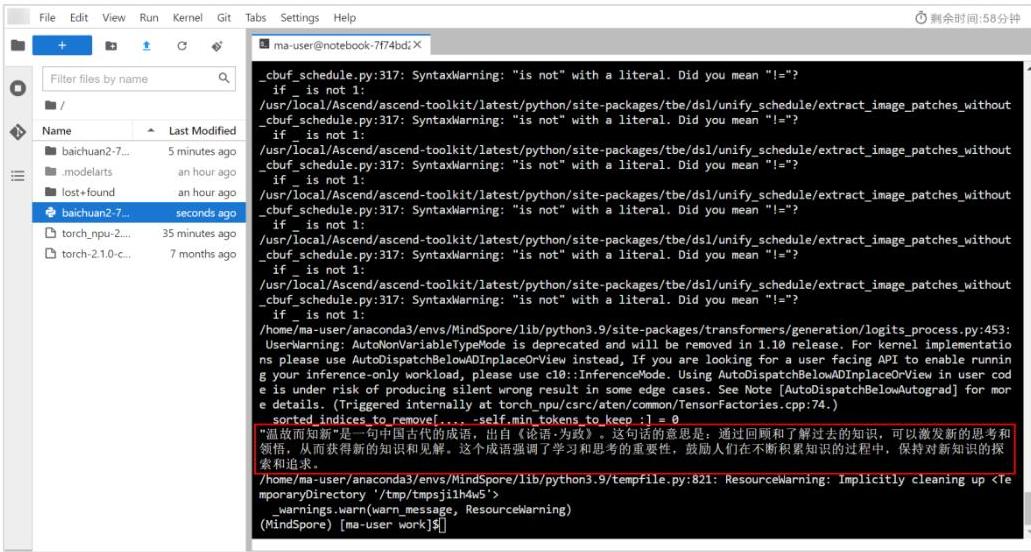
accelerate 库均已原生支持昇腾 NPU。

# 步骤 2 启动推理脚本

命令行终端执行以下命令：

python baichuan2-7b-chat-test.py

执行成功，如下所示：



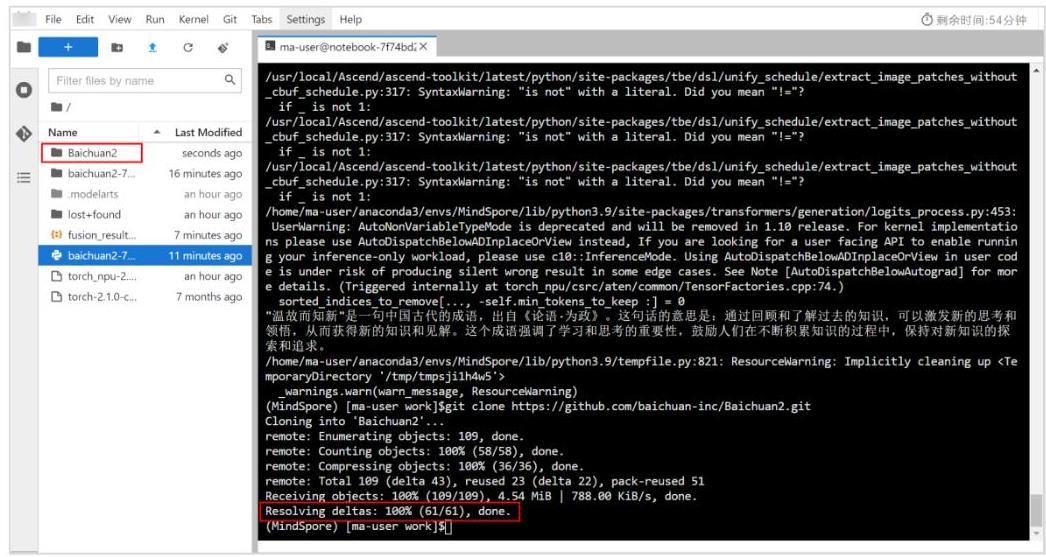
# 图2-4 推理输出

# 2.4 LoRA 微调

# 2.4.1 获取源码

命令行终端执行以下命令，下载源码。

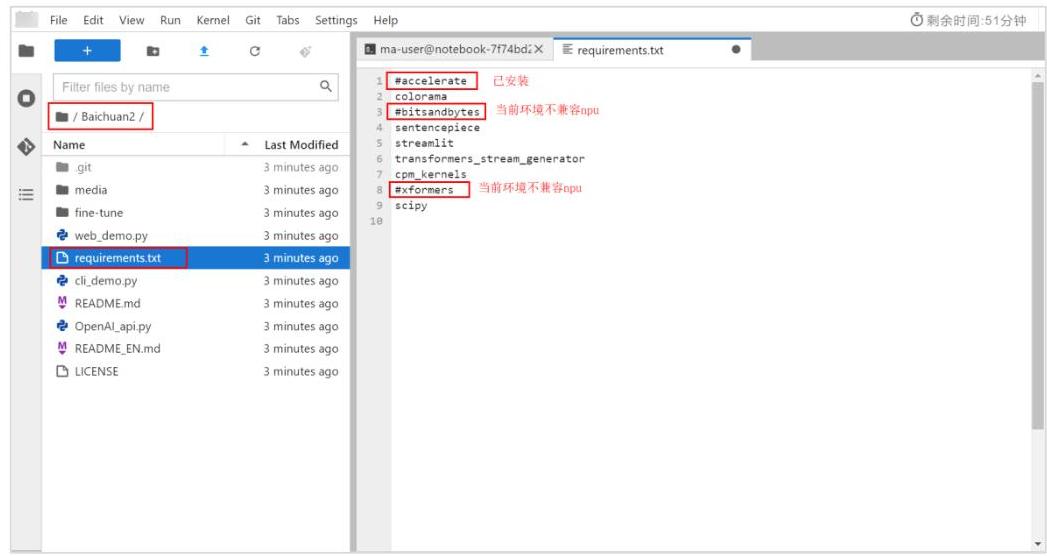
git clone https://github.com/baichuan-inc/Baichuan2.git



# 图2-5 下载源码

# 2.4.2 安装依赖库

参考下图，注释掉 accelerate、bitsandbytes 和 xformers 库。



# 图2-6 注释安装依赖库

修改完成，按 CTRL+S 保存。

命令行终端执行以下命令，安装依赖。

cd Baichuan2

pip install -r requirements.txt

pip install deepspeed==0.12.4 peft==0.9.0

deepspeed 是用于分布式训练的框架，现也原生支持昇腾 npu，peft 为高效微调的组件，支持 Iora 等高效低参微调方法，也已原生支持昇腾 NPU。

# 2.4.3 新建 lora 微调启动脚本

进入 Baichuan2/fine-tune 目录，新建 lora-fine-tune.sh 脚本文件，输入如下内容:

hostfile=""

deepspeed –hostfile= $hostfile fine-tune.py

–report\_to "none"

–data\_path "data/belle\_chat\_ramdon\_10k.json"

–model\_name\_or\_path "/home/ma-user/work/baichuan2-7b-chat-hf"

–output\_dir "output"

–model\_max\_length 512

–num\_train\_epochs 4

–per\_device\_train\_batch\_size 16

| –gradient\_accumulation\_steps 1 |
| --- |
| –save\_strategy epoch |
| –learning\_rate 2e-5 |
| –lr\_scheduler\_type constant |
| –adam\_beta10.9 |
| –adam\_beta2 0.98 |
| –adam\_epsilon 1e-8 |
| –max\_grad\_norm 1.0 |
| –weight\_decay 1e-4 |
| –warmup\_ratio 0.0 |
| –logging\_steps 1 |
| –gradient\_checkpointing True |
| –deepspeed ds\_config.json |
| –bf16 True |
| –tf32 False |
| –use\_lora True |

编辑完成，按 CTRL+S 保存。

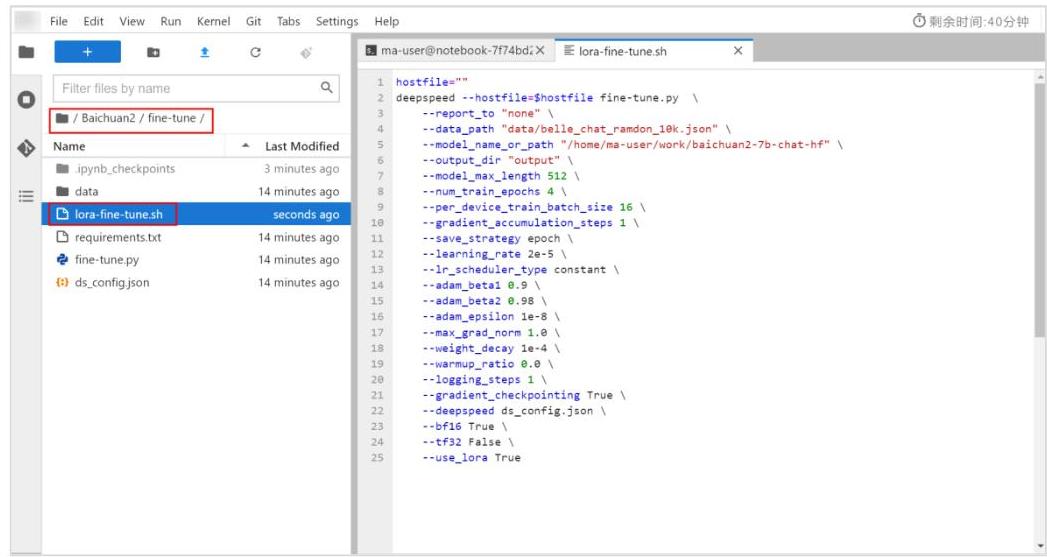


图2-7 新建 LoRA 微调 shell 脚本

# 注意:

* 上述脚本中使用了 deepspeed, deepspeed 是分布式训练框架，单卡下也可使用，也能对显存进行优化，单卡使用 deepspeed 只需将 hostfile=""置为空即可。
* 单卡微调需开启-bf16 True，否则会报显存不足错误，同时关闭- rf32 False，昇腾 NPU 暂不支持 tf32 数据类型。
* 上述脚本适合单机单卡和单机多卡环境，deepspeed 会自动识别单机环境下的所有 NPU， 多机多卡环境下需要指定具体的 hostfile。

# 2.4.4 执行微调脚本

命令行终端执行以下命令，启动微调脚本。

cd /̃work/Baichuan2/fine-tune

bash lora-fine-tune.sh

执行成功，如下所示：

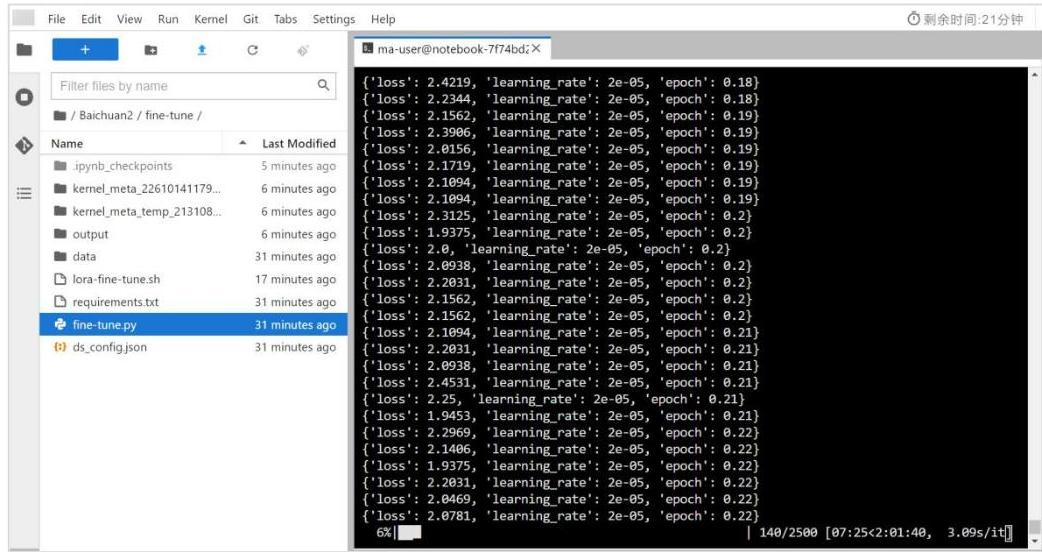


图2-8 启动训练

# 注意:

* baichuan2 核心微调代码脚本 fine-tune.py 使用的是 transformers+deepspeed 的方式， transformers 和 deepspeed 目前均已原生支持昇腾 NPU, 所以此处我们并未对 fine-tune.py 脚本做任何代码修改。fine-tune.py 脚本代码详情请自行阅读。
* 上述微调脚本使用的数据为样例数据，路径在：Baichuan2/fine-

tune/data/belle\_chat\_ramdon\_10k.json，只跑通流程，不保证最终效果，数据详情请自行查看。

# 2.5 实验总结

本实验介绍了使用 HuggingFace Transformers 大模型套件和 DeepSpeed 加速库进行 Baichuan2- 7B 模型的推理和微调，包括环境准备、模型准备、模型推理、模型微调。

# 2.6 思考题

请思考：如何可以提升模型微调效果?

答案：使用更多数据，增加 epoch 数量，修改优化器配置，增加 Ir 调度器，修改 DeepSpeed 配置文件等。

华为认证 AI 系列教程

HCIP-AI Solution Architect

# 大模型微调Llama2-7B

# 实验手册

版本：1.0



华为技术有限公司

版权所有 (C) 华为技术有限公司 2024。 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: https://e.huawei.com

# 华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构， 打造的覆盖ICT (Information and Communications Technology，信息通信技术) 全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-AI Solution Architect (Huawei Certified ICT Professional-AI Solution Architect，华为认证人工智能解决方案架构）主要面向于AI解决方案高级工程师、人工智能（大模型）训练工程师、大模型算法工程师和生成式AI工程师。HCIP-AI Solution Architect认证在内容上涵盖了大模型业务流程、自然语言处理算法、大模型算法基础、华为智算解决方案、大模型并行训练框架、预训练、微调和部署方案等内容，具备主流AI框架和并行框架使用能力，大模型微调和部署能力。

华为认证协助您打开行业之窗，开启改变之门，屹立在人工智能世界的潮头浪尖!

| 华为职业认证 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 云平台及云服务 | 云平台及云服务领域 | Cloud Computing | Cloud Service | Big Data | GaussDB | IoT | image |
| 基础软硬件 | 基础软硬件领域 | openEuler | openGauss | HarmonyOS | AI | Kunpeng |
| ICT基础设施 | IT基础设施领域 | Storage | Collaboration | Intelligent Vision | Digital Power | MDC |
| 数通领域 | Datacom | | WLAN | Security | |
|  |  |  |  |  | |
| 光领域 | Transmission | |  | Access | |
| 无线领域 | 5G | | | LTE | |
|  |  |  |  |  |  | |

# 前 言

# 简介

本书为 HCIP-AI Solution Architect 认证培训教程，适用于准备参加 HCIP-AI Solution Architect 考试的学员。

# 内容描述

本实验指导书共包含 4 个实验，包括开发环境搭建和 MindFormers 环境准备。具体四个实验为:

* 实验一为模型文件准备实验。
* 实验二为数据准备实验。
* 实验三为模型训练实验。
* 实验四为模型推理实验。

# 实验环境说明

华为云 ModelArts 购买相应资源。

# 目录

前 言 ..3

简介. ... 3

内容描述. ... 3

实验环境说明. ... 3

1 ModelArts 开发环境搭建. ..6

1.1 进入华为云 ModelArts 控制台 ... 6

1.2 创建 Notebook 训练作业. .. 6

1.3 启动 Notebook 进入开发环境. ...8

1.4 停止 Notebook 训练作业. ...9

2 Llama2-7B 大模型微调-MindFormers .11

2.1 实验介绍. ...11

2.1.1 关于本实验. ...11

2.1.2 实验目的. ...11

2.1.3 实验流程. ...11

2.1.4 实验环境. ...11

2.2 环境准备. ...12

2.2.1 安装 MindFormers. ...12

2.3 模型文件准备. ... 12

2.3.1 创建权重存放目录. ...12

2.3.2 权重下载. ...12

2.3.3 分词器文件下载. ...13

2.4 数据准备. ...14

2.4.1 创建数据存放目录. ...14

2.4.2 下载原始数据. ...14

2.4.3 数据格式转换. ...15

2.4.4 分词器预处理并保存成 Mindrecord 格式 ...18

2.5 模型训练. ...19

2.5.1 修改配置文件. ...19

2.5.2 启动训练脚本. ...20

2.6 模型推理. ..22

2.6.1 修改配置文件. ...22

2.6.2 新建推理脚本. ...22

2.6.3 执行推理脚本. ...25

2.7 实验总结. ...26

2.8 思考题. ...26

# 1.1 进入华为云 ModelArts 控制台

在华为云 ModelArts 主页（https://www.huaweicloud.com/product/modelarts.html），点击“控制台”进入 ModelArts 的管理页面。

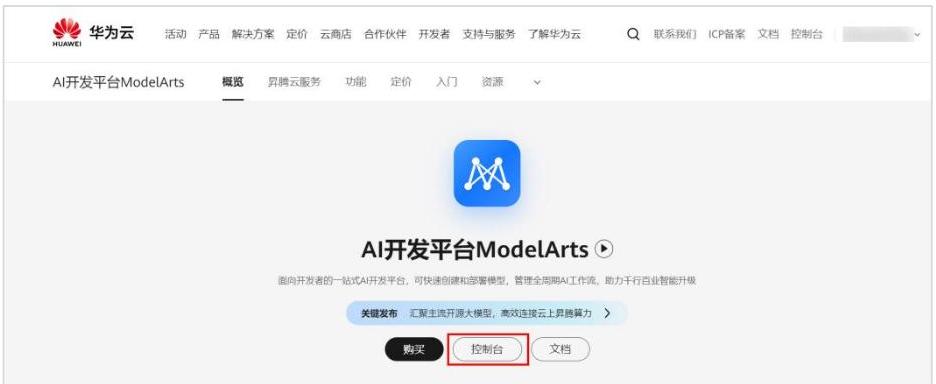


图1-1 华为云 ModelArts 主页

# 1.2 创建 Notebook 训练作业

控制台区域选择“西南-贵阳一”，在左侧菜单栏中选择“开发环境”的“Notebook”，点击进入 Notebook 创建页面。

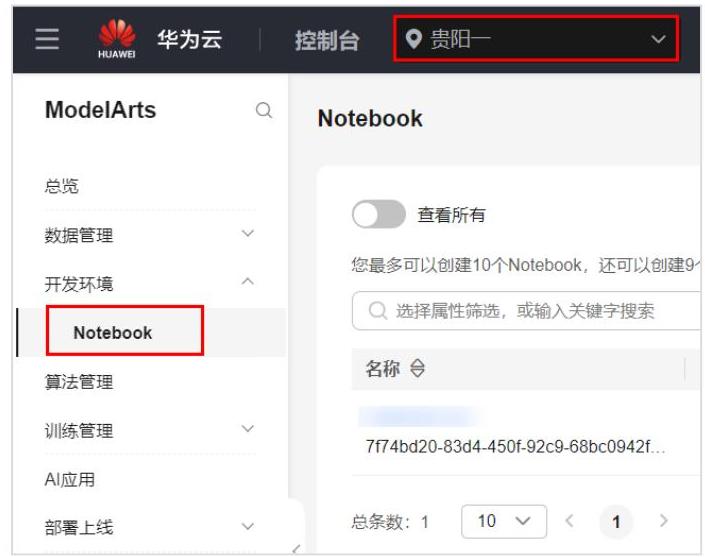


图1-2 ModelArts 控制台

点击“创建”按钮，创建一个新的 Notebook，其配置如下：

* 名称: 自定义。
* 自动停止: 建议 2 小时。
* 镜像：mindspore\_2.2.0-cann\_7.0.1-py\_3.9-euler\_2.10.7-aarch64-snt9b。
* 资源池: 公共资源池。
* 类型: ASCEND。
* 规格: Ascend: 1\*ascend-snt9b(64G)|ARM: 24 核 192GB。
* 存储配置: 云硬盘 EVS。
* 磁盘规格: 1000GB。

如图所示:



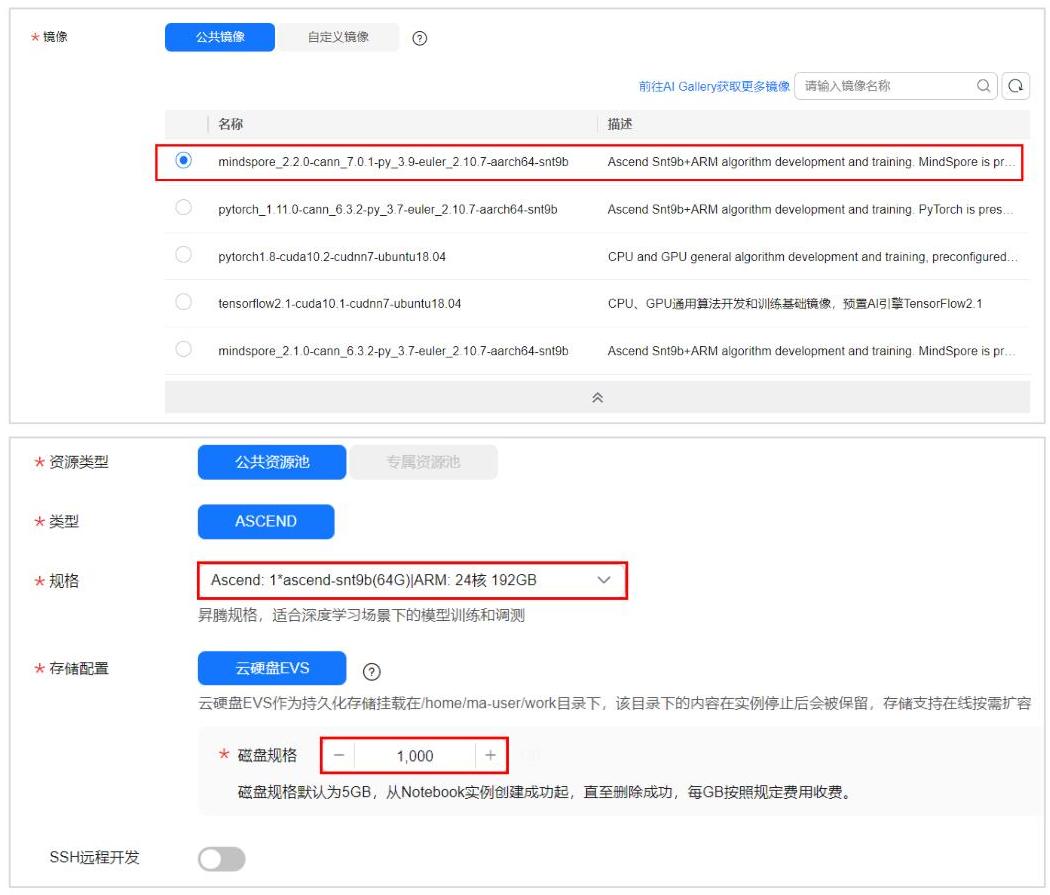


图1-3 Notebook 配置

配置完成之后“立即创建”，规格确认无误之后“提交”。

# 1.3 启动 Notebook 进入开发环境

当上一步创建好 Notebook 状态显示为 “运行中” 时，在右侧 “操作” 栏中 “打开”，即可进入在线编程页面。



图1-4 打开 Notebook 环境

可以在此页面创建或编辑项目，如图所示：

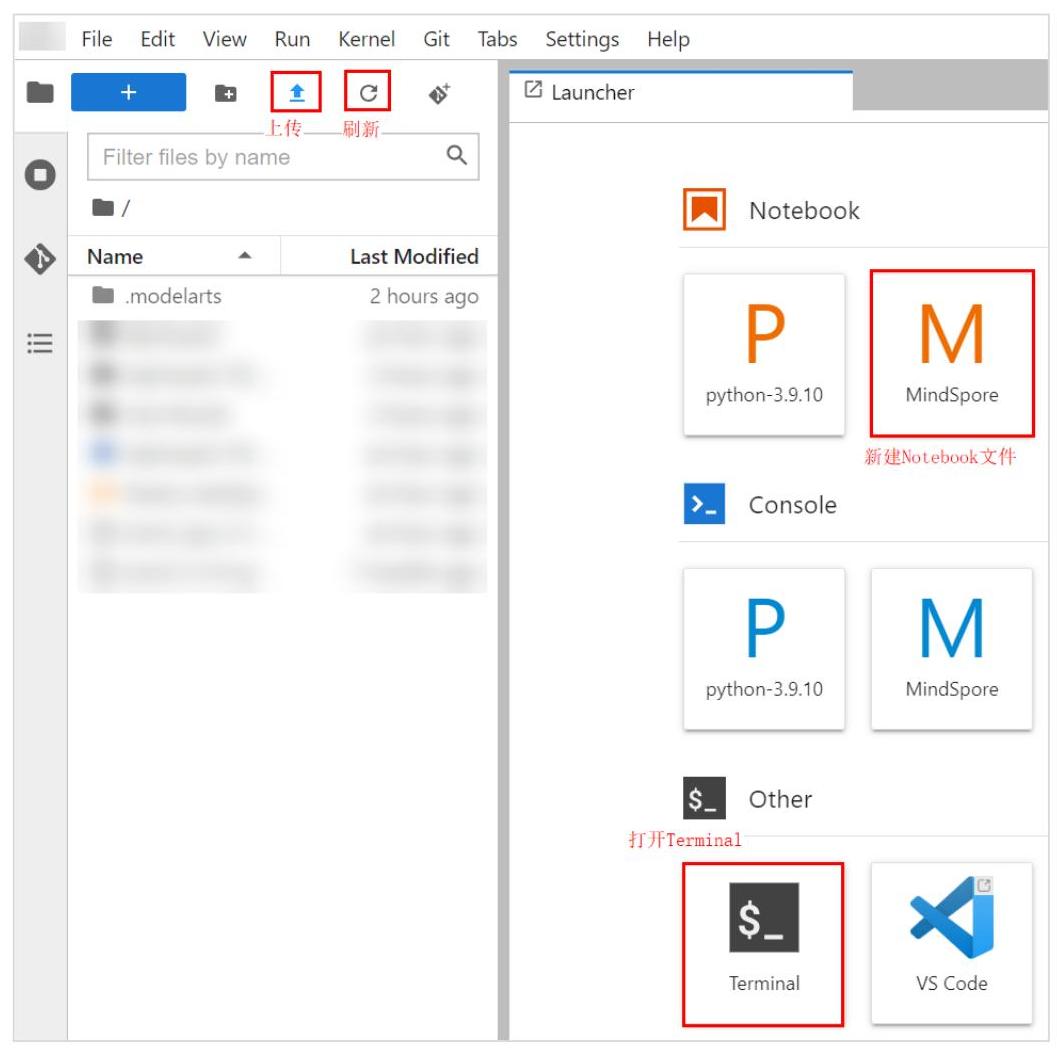


图1-5 页面操作

注意：Notebook 环境内上传、创建和编辑的文件均在/home/ma-user/work 目录下。

# 1.4 停止 Notebook 训练作业

实验完成之后，请及时关闭 Notebook 训练作业，避免产生不必要的资源浪费。

登录华为云 ModelArts 控制台，在“操作”栏选择“停止”操作。 如下图所示:



# 图1-6 及时停止 Notebook

至此，训练用的线上 Notebook 环境搭建完成。

# C Llama2-7B 大模型微调-MindFormers

# 2.1 实验介绍

# 2.1.1 关于本实验

本实验介绍如何在 ModelArts 平台上使用 MindFormers 大模型套件进行 Llama2-7B 模型的 LoRA

微调和推理，MindSpore Transformers 套件的目标是构建一个大模型训练、微调、评估、推

理、部署的全流程开发套件：提供业内主流的 Transformer 类预训练模型和 SOTA 下游任务应

用，涵盖丰富的并行特性，期望帮助用户轻松的实现大模型训练和创新研发。

# 2.1.2 实验目的

* 掌握 MindFormers 套件的使用流程和方法;
* 掌握 Llama2-7B 大模型的 LoRA 微调流程和方法。

# 2.1.3 实验流程

* 环境准备;
* 模型文件准备;
* 数据准备;
* 模型训练;
* 模型推理。

# 2.1.4 实验环境

表2-1 实验清单

| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| --- | --- | --- | --- | --- |
| Llama2-7B大模 型实操 | Llama2-7B模型 LoRA微调&推理 | 进阶 | mindspore-2.2.0 mindformers-1.0.2 | ModelArts Ascend |

# 2.2 环境准备

# 2.2.1 安装 MindFormers

进入 Notebook 开发环境，打开命令行终端，执行以下命令安装 MindFormers。

git clone -b r1.0 https://gitee.com/mindspore/mindformers.git cd mindformers bash build.sh

执行成功，如下所示：

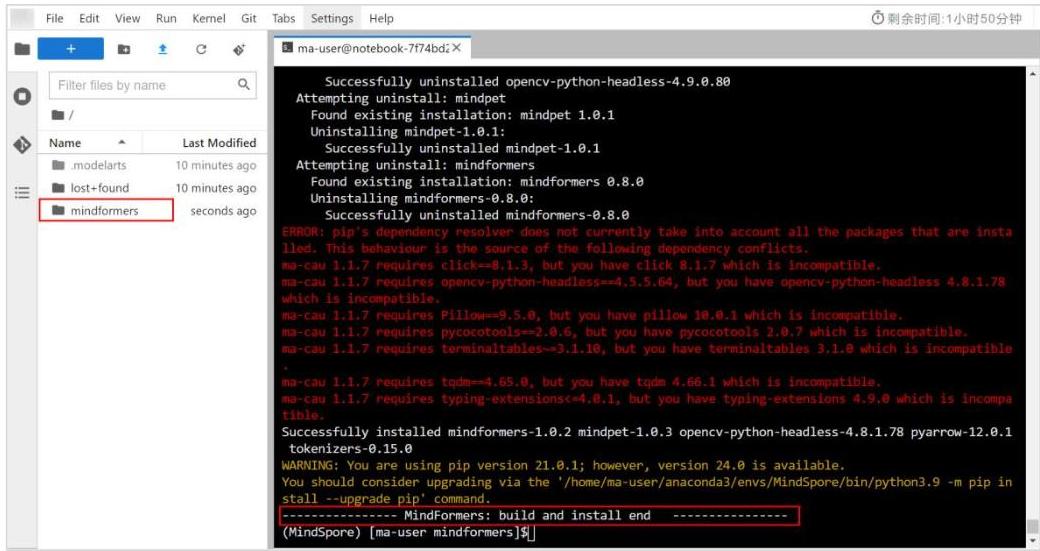


图2-1 安装 MindFormers

# 2.3 模型文件准备

# 2.3.1 创建权重存放目录

命令行终端执行以下命令，创建并进入模型权重存放目录。

mkdir -p /̃work/mindformers/models/llama2

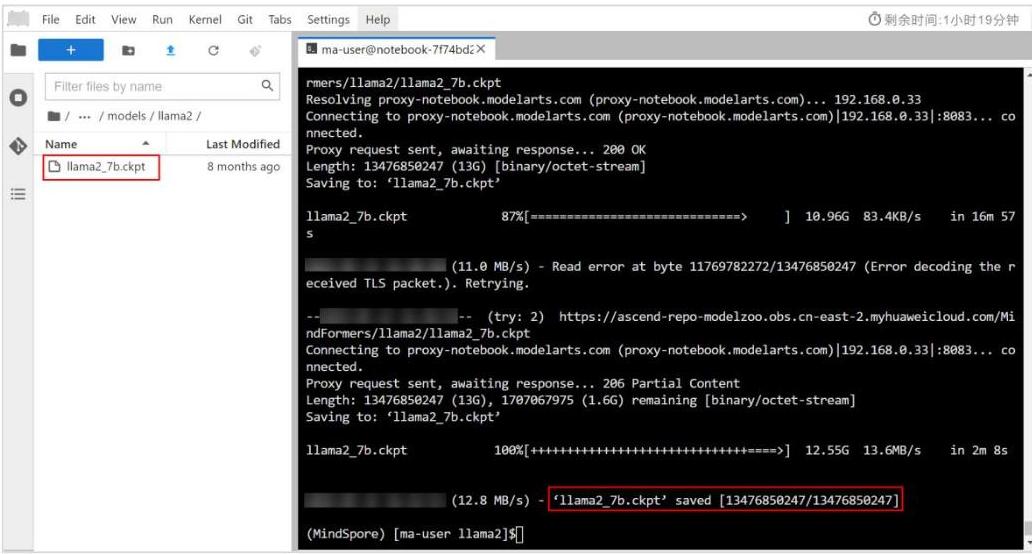
cd models/llama2

# 2.3.2 权重下载

命令行终端执行以下命令，下载转换好的 Llama2-7B 预训练权重。

wast https://ascend-repo-modelzoo.obs.cn-east-2.myhuaweicloud.com/MindFormers/llama2/llama2\_7b.ckpt

执行成功，如下所示:



# 图2-2 下载权重文件

注意: 上述权重文件不是存放在贵阳一区域，直接下载可能会提示超时下载失败，如果遇到下载失败的情况，此时可配置绕过内部网络代理，命令行终端执行以下命令进行配置。

export no\_proxy=repo.myhuaweicloud.com, modelarts.cn-southwest-2.myhuaweicloud.com, obs.cn-southwest-

2.myhuaweicloud.com, iam.cn-southwest-2.huaweicloud.com, iam-cache-proxy.cn-southwest-

2.myhuaweicloud.com, dli.cn-southwest-2.myhuaweicloud.com,127.0.0.1, localhost,172.16.\*

export NO\_PROXY=repo.myhuaweicloud.com, modelarts.cn-southwest-2.myhuaweicloud.com, obs.cn-southwest-

2.myhuaweicloud.com, iam.cn-southwest-2.huaweicloud.com, iam-cache-proxy.cn-southwest-

2.myhuaweicloud.com, dli.cn-southwest-2.myhuaweicloud.com,127.0.0.1, localhost,172.16.\*

配置完后重新执行下载命令即可。

# 2.3.3 分词器文件下载

命令行终端执行以下命令，下载分词器文件。

we thttps://ascend-repo-modelzoo.obs.cn-east-2.myhuaweicloud.com/MindFormers/llama2/tokenizer.model

执行成功，如下所示:

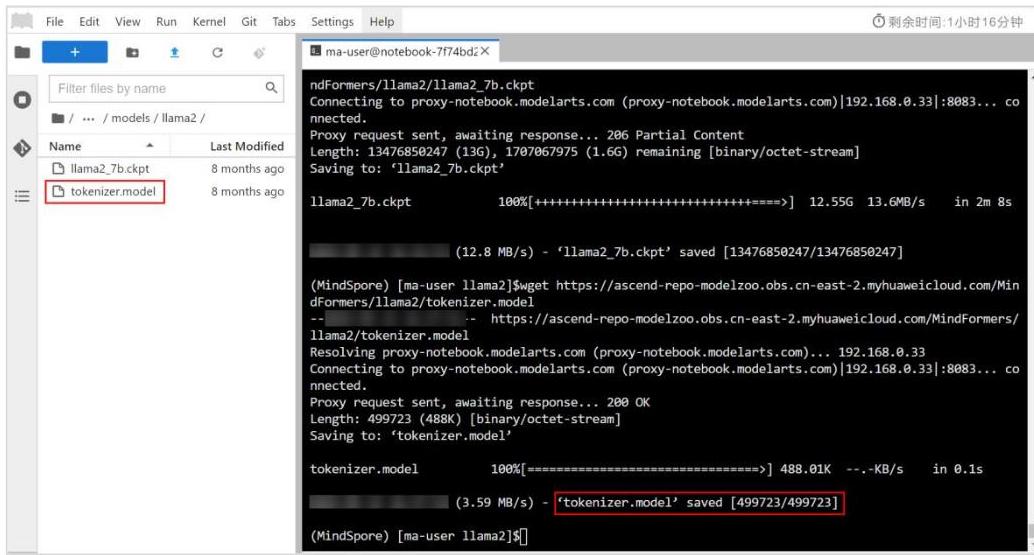


图2-3 下载分词器文件

# 2.4 数据准备

# 2.4.1 创建数据存放目录

命令行终端执行以下命令创建并进入数据存放目录。

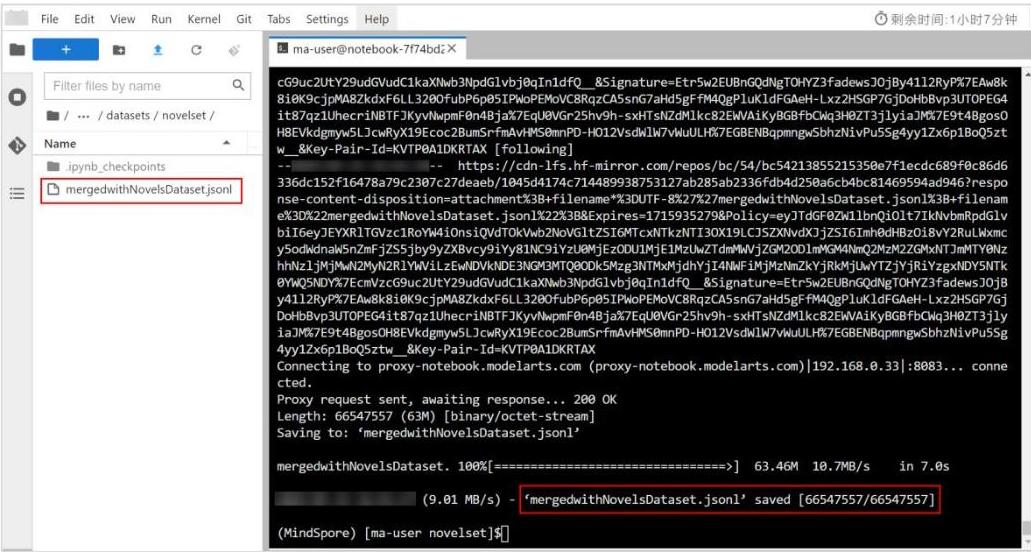
| mkdir -p /̃work/mindformers/datasets/novelset |
| --- |
| cd /̃work/mindformers/datasets/novelset |

# 2.4.2 下载原始数据

命令行终端执行以下命令下载原始数据。

Wget https://hf-mirror.com/datasets/Delius/ChineseWebNovel/resolve/main/mergedwithNovelsDataset.json1

执行成功，如下所示：



# 图2-4 下载原始数据

NovelSet 数据集是开源的指令微调数据集，包括新闻撰写、常识对话、小说扩写等指令数据， 数据样例如下所示:

{"Instruction": "请将给你的文本内容扩写成新闻", "Input": "95岁老人五代同堂,一大家人其乐融融", "Response": "今年 95岁的顾鸭英老人住在南通港闸区陈桥街道树北村。 n虽然年事已高, 可老人身体健朗, 耳不聋眼不花。 n而最令她高兴的是, 生活了几十年的老房子里, 如今已是五代同堂, 一大家子其乐融融。 n今天上午, 记者见到顾鸭莢老人时, 她正坐在村口和女儿儿媳一起打牌。 n顾鸭英老人共育有一对儿女。 n日子虽不富裕,但老人从小都教导子女要与人为善。 n儿女长大后,各自成家,老人留在老宅和儿子儿媳一起生活。 n然而，前些年，儿子突然中风，老人便和儿媳继续包揽了家里的家务活儿。 n老人身体一直不错，长寿的秘诀都归结于开朗的性格。 如今,家里五代同堂,看着慢慢长大的孙辈,成了老人晚年生活最幸福的事。" }

{"Instruction": "组织一场以环境保护为主题的志愿者活动。 n地点: 城市公园 n时间: 周六, 上午9点至下午2点 n资源: 纸

张、笔、清洁工具、环保小贴士等", "Input": "", "Response": "招募志愿者参加环保活动: 时间为周六上午9点至下午2点, 地点在城市公园, 提供资源包括纸张、笔、清洁工具、环保小贴士等。活动包括垃圾清理、环保知识宣传等。感谢大家一起为环保出一份力! " }

# 图2-5 原始数据样例

# 2.4.3 数据格式转换

# 步骤 1 新建数据转换脚本

在当前数据存放目录下新建数据转换脚本 novelset2alpaca.py，将数据集转换成 alpaca 数据格式，方便后续使用 alpaca 数据预处理工具。脚本内容如下：

import json

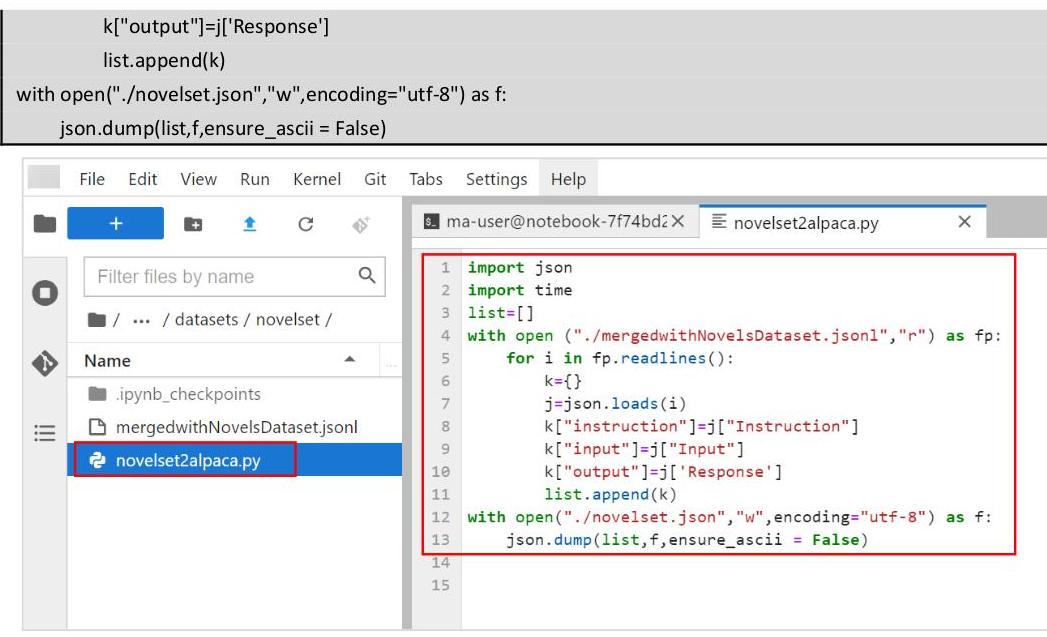
import time

list=[]

with open ("./mergedwithNovelsDataset.jsonl","r") as fp:

for i in fp.readlines():

son.loads(i)



# 图2-6 数据转换脚本

# 步骤 2 命令行终端执行转换脚本

python novelset2alpaca.py

执行成功，生成转换后的 novelset.json 文件，如下所示：

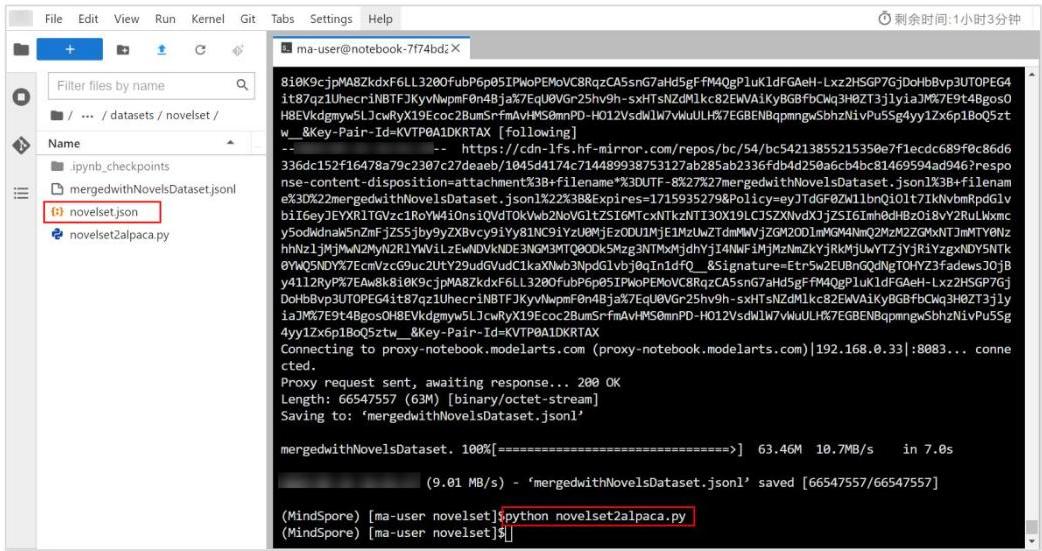


图2-7 Alpaca 数据文件

转换后的 alpaca 数据样例如下所示:

# 大模型微调 Llama2-7B 实验手册

v 1213:

instruction: "请将给你的文本内容扩写成新闻"

input: "95岁老人五代同堂, 一大家人其乐融融"

output: "今年95岁的顾鸭英老人住在南通港闸区陈桥街道树北村。虽然年事已高, 可老人身体健朗, 耳不聋眼不花。 而最令她高兴的是, 生活了几十年的老房子里, 如今已是五代同堂, 一大家子其乐融融。 今天上午, 记者见到顾鸭英老人时, 她正坐在村口和女儿儿媳一起打牌。 顾鸭英老人共育有一对儿女。 日子虽不富裕, 但老人从小都教导子女要与人为善。 儿女长大后, 各自成家, 老人留在老宅和儿子儿媳一起生活。 然而, 前些年, 儿子突然中风, 老人便和儿媳继续包揽了家里的家务活儿。 老人身体一直不错, 长寿的秘诀都归结于开朗的性格。如今, 家里五代同堂, 看着慢慢长大的孙辈, 成了老人晚年生活最幸福的事。" v 3707:

instruction: "组织一场以环境保护为主题的志愿者活动。 地点: 城市公园 时间: 周六, 上午9点至下午2点 资源: 纸张、笔、 清洁工具、环保小贴士等” input: ""

output: "招募志愿者参加环保活动: 时间为周六上午9点至下午2点, 地点在城市公园, 提供资源包括纸张、笔、清洁工具、环保小贴士等。活动包括垃圾清理、环保知识宣传等。感谢大家一起为环保出一份力! "

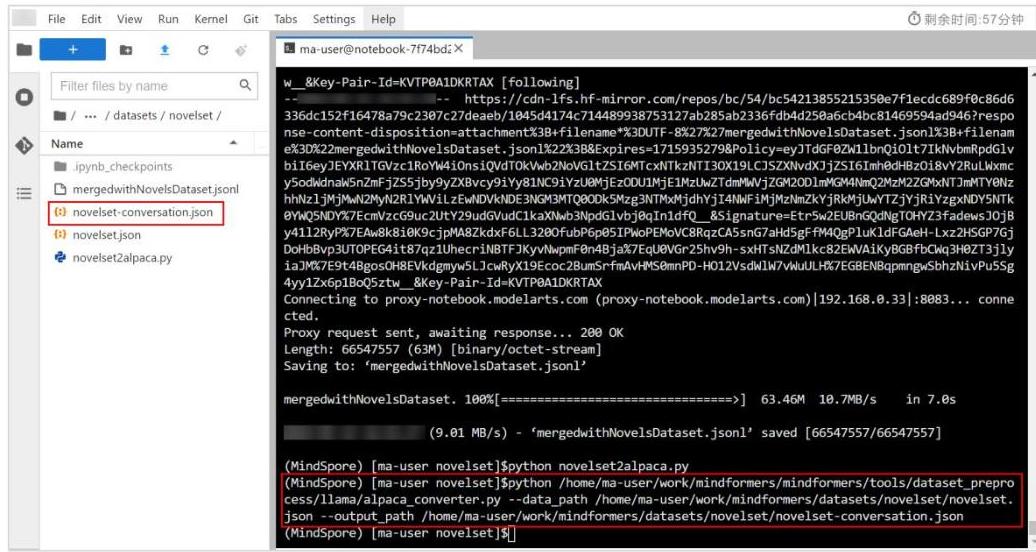
# 图2-8 Alpaca 数据样例

# 步骤 3 多轮对话格式转换

继续执行 alpaca\_converter.py，使用 fastchat 工具添加 prompts 模板，将原始数据集转换为多轮对话格式。

python /home/ma-user/work/mindformers/mindformers/tools/dataset\_preprocess/llama/alpaca\_converter.py – data\_path /home/ma-user/work/mindformers/datasets/novelset/novelset.json –output\_path /home/mauser/work/mindformers/datasets/novelset/novelset-conversation.json

执行成功，如下所示：



# 图2-9 多轮对话数据文件

转换后，多轮对话数据样例如下:

v 1213: id: "1214" - conversations: [] 2 items 70: from: "human" value: "Below is an instruction that describes a task, paired with an input that provides further context of the context. ext. Write a response that appropriately completes the request. ### Instruction: 请将给你的文本内容扩写成新闻 ### Input: 95岁老人五代同堂,一大家人其乐融融 ### Response:" - 1: from: "gpt" value: "今年95岁的顾鸭英老人住在南通港闸区陈桥街道树北村。虽然年事已高, 可老人身体健朗, 耳不聋眼不花。 而最令她高兴的是, 生活了几十年的老房子里, 如今已是五代同堂, 一大家子其乐融融。 今天上午, 记者见到顾鸭英老人时, 她正坐在村口和女儿儿媳一起打牌。 顾鸭英老人共育有一对儿女。 日子虽不富裕, 但老人从小都教导子女要与人为善。 儿女长大后, 各自成家, 老人留在老宅和儿子儿媳一起生活。 然而, 前些年, 儿子突然中风, 老人便和儿媳继续包揽了家里的家务活儿。 老人身体一直不错, 长寿的秘诀都归结于开朗的性格。如今, 家里五代同堂, 看着慢慢长大的孙辈, 成了老人晚年生活最幸福的事。"

v 3707:

id: "3708"

* conversations: [] 2 items

v

from: "human"

value: "Below is an instruction that describes a task. Write a response that appropriately completes the completest structure of the state of the state.

he request. ### Instruction: 组织一场以环境保护为主题的志愿者活动。 地点: 城市公园 时间: 周六, 上午9点至下

午2点 资源: 纸张、笔、清洁工具、环保小贴士等 ### Response:"

* 1:

from: "gpt"

value: "招募志愿者参加环保活动: 时间为周六上午9点至下午2点, 地点在城市公园, 提供资源包括纸张、笔、清洁工具、环保小贴士等。活动包括垃圾清理、环保知识宣传等。感谢大家一起为环保出一份力! "

# 图2-10 多轮对话数据样例

# 2.4.4 分词器预处理并保存成 Mindrecord 格式

执行 Ilama\_preprocess.py，进行数据预处理、Mindrecord 数据生成，将带有 prompt 模板的数据转换为 Mindrecord 格式。

python /home/ma-user/work/mindformers/mindformers/tools/dataset\_preprocess/llama/llama\_preprocess.py

–dataset\_type qa

-input\_glob /home/ma-user/work/mindformers/datasets/novelset/novelset-conversation.json

–model\_file /home/ma-user/work/mindformers/models/llama2/tokenizer.model

–seq\_length 2048

-output\_file /home/ma-user/work/mindformers/datasets/novelset/novelset.mindrecord

执行成功，如下所示：

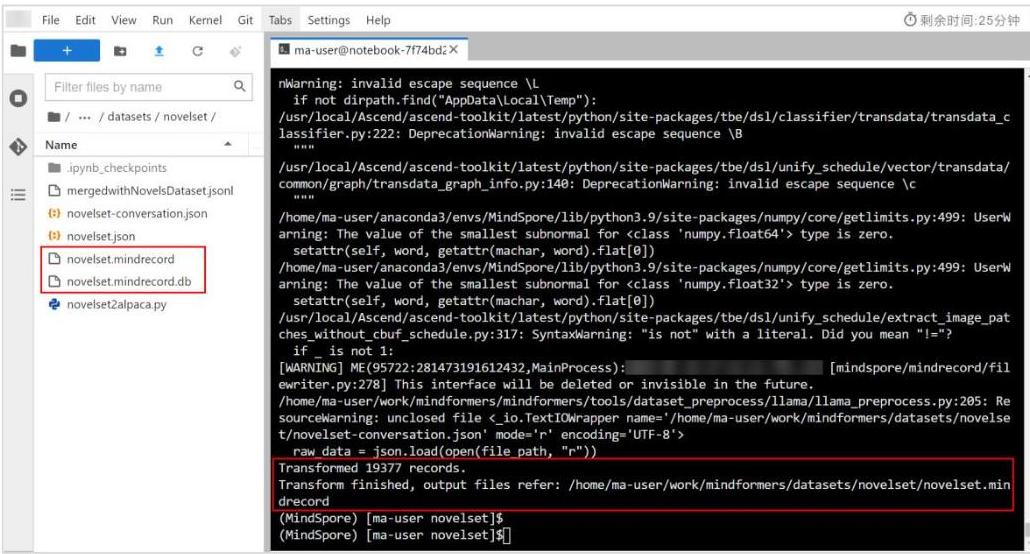


图2-11 Mindrecord 数据文件

# 2.5 模型训练

# 2.5.1 修改配置文件

打开并编辑 mindformers/configs/llama2/run\_llama2\_7b\_lora\_910b.yaml 配置文件。

# 步骤 1 配置权重路径

load\_checkpoint: ’/home/ma-user/work/mindformers/models/llama2/llama2\_7b.ckpt’

seed: 0

output\_dir: ’./output’ # path to save checkpoint/strategy

load\_checkpoint: ’/home/ma-user/work/mindformers/models/llama2/llama2\_7b.ckpt’

src\_strategy\_path\_or\_dir: ”

auto\_trans\_ckpt: False # If true, auto transform load\_checkpoint to load in distributed model

only\_save\_strategy: False

resume\_training: False

run\_mode: ’finetune’

# 图2-12 配置权重路径

# 步骤 2 配置训练数据路径

dataset\_dir: "/home/ma-user/work/mindformers/datasets/novelset/novelset.mindrecord"

# dataset

train\_dataset: &train\_dataset

data\_loader:

type: MindDataset

dataset\_dir: "/home/ma-user/work/mindformers/datasets/novelset/novelset.mindrecord"

shuffle: True

input\_columns: ["input\_ids", "labels"] # "input\_ids", "labels", labels are used in instruction fi

python\_multiprocessing: False

drop\_remainder: True

batch\_size: 4

repeat: 1

numa\_enable: False

prefetch\_size: 1

train\_dataset\_task:

type: CausalLanguageModelDataset

dataset\_config: \*train\_dataset

# if True, do evaluate during the training process. if false, do nothing.

; # note that the task trainer should support \_evaluate\_in\_training function.

do\_eval: False

# 图2-13 配置数据集路径

# 步骤 3 关闭分布式并行



# 图2-14 关闭分布式并行

修改完成，按 CTRL+S 保存。

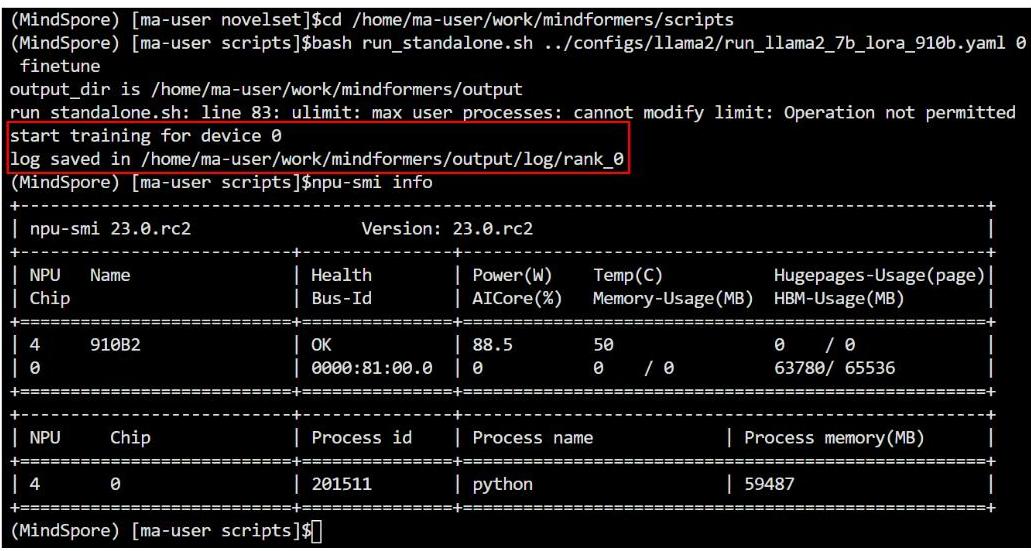
# 2.5.2 启动训练脚本

命令行终端执行以下命令，启动训练脚本。

cd /home/ma-user/work/mindformers/scripts

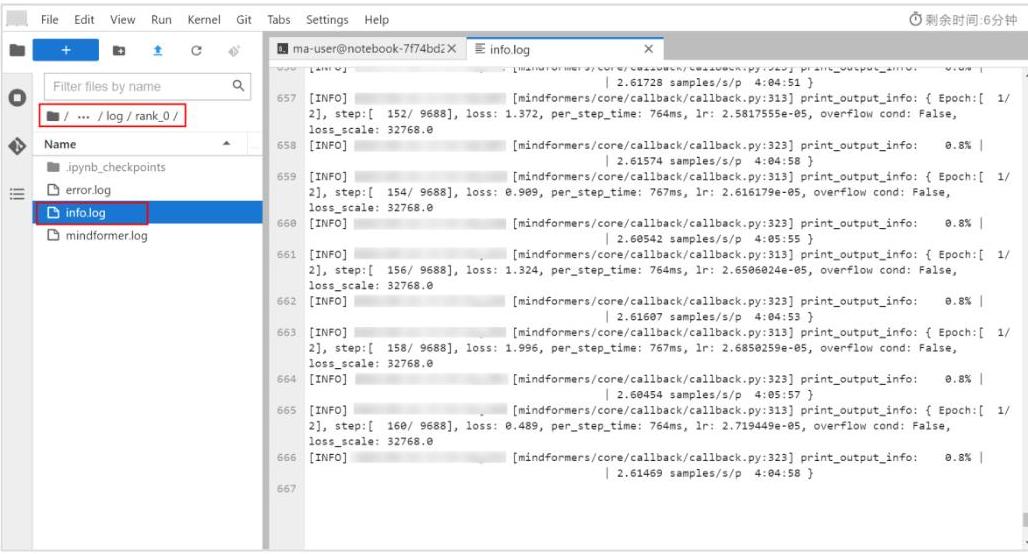
bash run\_standalone.sh ../configs/llama2/run\_llama2\_7b\_lora\_910b.yaml 0 finetune

执行成功，如下所示:



# 图2-15 启动训练

训练进程后台运行，可在/mindformers/output/log/rank\_0 目录下查看实时更新的训练日志。



# 图2-16 训练日志

训练 2 个 epoch，时长约 4 个小时。训练完成，output 目录下生成训练好的权重文件:

* checkpoint(含优化器参数)存储路径：mindformers/output/checkpoint。
* checkpoint(不含优化器参数)存储路径：mindformers/output/checkpoint\_network。

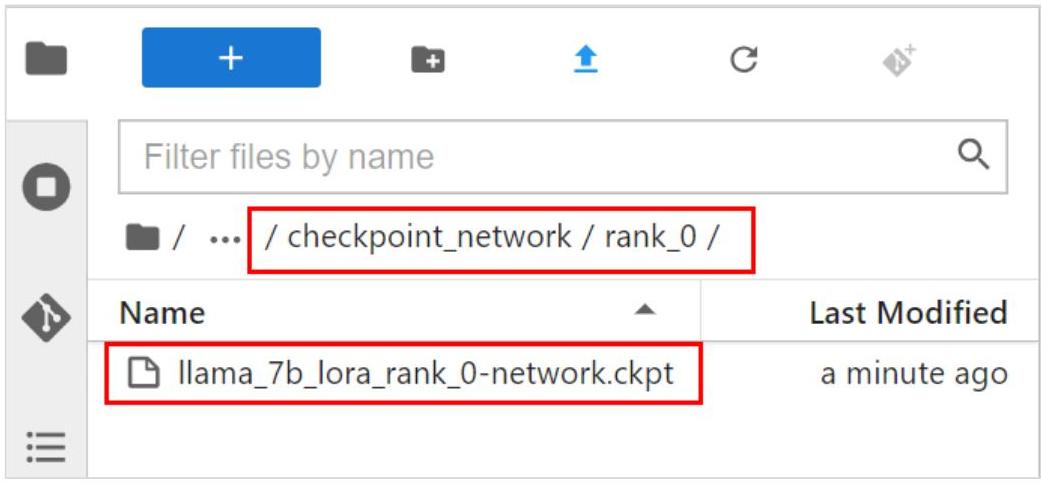


图2-17 训练权重文件

# 2.6 模型推理

# 2.6.1 修改配置文件

打开并编辑 mindformers/configs/llama2/run\_llama2\_7b\_lora\_910b.yaml 配置文件，修改权重初

始化类型，将默认的“float16”改为“float32”，保障推理精度（非必须，精度要求高场景下可开启）。



# 图2-18 修改权重初始化类型

# 2.6.2 新建推理脚本

work 目录下新建推理脚本 llama2\_7b\_infer.py，如下所示： 大模型微调 Llama2-7B 实验手册

(c) 剩余时间: 51分钟 from mindformers import MindFormerConfig, LlamaConfig, TransformerOpParallelConfig, AutoTokenizer, LlamaForCausalLM, AutoProcessor from mindformers.tools.utils import str2bool, get\_real\_rank from mindformers.trainer.utils import get\_last\_checkpoint from mindformers.pet import get\_pet\_model, LoraConfig # Sbatch输入 prompt\_input\_temple = ( "Below is an instruction that describes a task, paired with an input that provides further context. " "Write a response that appropriately completes the request. n n" "### Instruction: n {instruction } n n### Input: n {input } n n### Response:" ) prompt\_no\_input\_temple = ( "Below is an instruction that describes a task. " "Write a response that appropriately completes the request. n n" "### Instruction: n {instruction } n n### Response:" inputs = [prompt\_input\_temple.format\_map(("instruction":"请将给你的文本内容扩写成新闻", "input":"杭州某公司员工中1亿元彩票大奖。" }),

# 图2-19 新建推理脚本

# 脚本代码详情如下:

import argparse

import mindspore as ms

import numpy as np

import os

from mindspore import load\_checkpoint, load\_param\_into\_net

from mindspore.train import Model

from mindformers import MindFormerConfig, LlamaConfig, TransformerOpParallelConfig, AutoTokenizer,

LlamaForCausalLM, AutoProcessor

from mindformers import init\_context, ContextConfig, ParallelContextConfig

from mindformers.tools.utils import str2bool, get\_real\_rank

from mindformers.trainer.utils import get\_last\_checkpoint

from mindformers.pet import get\_pet\_model, LoraConfig

# 多 batch 输入

prompt\_input\_temple

"Below is an instruction that describes a task, paired with an input that provides further context. "

"Write a response that appropriately completes the request. n n"

"### Instruction: n {instruction } n n### Input: n {input } n n### Response:"

)

prompt\_no\_input\_temple = (

"Below is an instruction that describes a task. "

"Write a response that appropriately completes the request. n n"

"### Instruction: n {instruction } n n### Response:"

)

# # 初始化环境

inputs = [prompt\_input\_temple.format\_map( {"instruction":"请将给你的文本内容扩写成新闻", "input":"杭州某公司员工中 1 亿元彩票大奖。" }),

prompt\_no\_input\_temple.format\_map( {"instruction":"请帮我制定一份详细的健身计划？" })]

# set model config

llama2\_7b\_config\_path = "/home/ma-user/work/mindformers/configs/llama2/run\_llama2\_7b\_lora\_910b.yaml"

config = MindFormerConfig(llama2\_7b\_config\_path)

init\_context(use\_parallel=config.use\_parallel,

context\_config=config.context,

parallel\_config=config.parallel)

model\_config = LlamaConfig(\*\*config.model.model\_config)

model\_config.parallel\_config = TransformerOpParallelConfig(\*\*config.parallel\_config)

model\_config.batch\_size len(inputs)

model\_config.use\_past = True

model\_config.seq\_length 推理时可适当改小，提升推理速度

model\_config.do\_sample True # 开启后，减少推理结果中词语重复的情况

model\_config.checkpoint\_name\_or\_path = "/home/ma-

user/work/mindformers/output/checkpoint\_network/rank\_0/llama\_7b\_lora\_rank\_0-network.ckpt"

print(f"config is: {model\_config }")

# build tokenizer

# 在线加载

tokenizer = AutoTokenizer.from\_pretrained("Ilama2\_7b")

# build model from config

model = LlamaForCausalLM(model\_config)

if model\_config.pet\_config:

print("————–Init lora params————–")

pet\_config LoraConfig

lora\_rank=config.model.model\_config.pet\_config.lora\_rank,

lora\_alpha=config.model.model\_config.pet\_config.lora\_alpha,

lora\_dropout=config.model.model\_config.pet\_config.lora\_dropout,

target\_modules=config.model.model\_config.pet\_config.target\_modules

)

model get\_pet\_model(model, pet\_config)

for example\_input in inputs:

input\_ids = tokenizer([example\_input], max\_length=model\_config.seq\_length,

padding="max\_length")["input\_ids"]

outputs model.generate(input\_ids,

# # lora 权重初始化

max\_length=model\_config.max\_decode\_length,

do\_sample=model\_config.do\_sample,

top\_k=model\_config.top\_k,

top\_p=model\_config.top\_p)

print(tokenizer.decode(outputs[0]))

# 2.6.3 执行推理脚本

命令行终端执行以下命令，启动推理脚本。

work

python llama2\_7b\_infer.py

执行成功，如下所示：

<s>Below is an instruction that describes a task, paired with an input that provides further containing a single state. ext. Write a response that appropriately completes the request.

# Instruction:

请将给你的文本内容扩写成新闻

# Input:

杭州某公司员工中1亿元彩票大奖。

# Response:

中国彩票网2016-11-2422:50。

记者访问。

杭州市彩票总公司。

杭州某投资集团员工在杭州彩票中心投注了100元，准备念念不忘的报车去上班。

可惜, 他的念头竟然被 1亿元彩金弄到了头上。

杭州市某投资公司的一位员工叫彭某，他报车上班时，偶尔会报车带上一张杭州快三。

2015年11月18日早上，报车上班的时候，彭某忍不住想报一张，但又有点担心。

“刚才跟别人聊聊彩票，说刚刚报了一张快三，报了100元，报到了1000000，我忍不住拿出来看看，忽然发现，

这张报的号码刚好是今天的报号，我就把报号??

<s>Below is an instruction that describes a task. Write a response that appropriately completes the same component of the same state. he request.

# Instruction:

请帮我制定一份详细的健身计划?

# Response: 好的, 以下是一个详细的健身计划, 可以根据需要进行调整:

日期 | 时间 | 体育活动 | 锻炼 | 卡路里 | 评分

45 分钟

1200 卡路里 | 5

| 2100 卡路里 | 7

1200 卡路里 | 5

存泳 │ 45 分钟 │ 1800 卡路里 │ 6

7 | 早上 | 跑步或游泳 | 30 分钟 | 1200 卡路里 | 5

8 | 下午 | 跑步或游泳 | 60 分-

(MindSpore) [ma-user work] $

# 图2-20 模型推理输出

# 2.7 实验总结

本实验介绍了使用 MindFormers 大模型套件进行 Llama2-7B 模型的 LoRA 微调和推理，包括环

境准备、模型文件准备、数据处理、模型 LoRA 微调、模型推理。

# 2.8 思考题

请思考：推理脚本中的 model\_config.use\_past = True 代表什么含义?

答案：开启增量推理，自回归生成过程中，存在重复计算项，可复用以往的计算结果以节省时间，用于加速文本生成。

华为认证 AI 系列教程

HCIP-AI Solution Architect

大模型压缩部署

# 实验手册

版本：1.0

版权所有 (C) 华为技术有限公司 2024。 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: https://e.huawei.com

# 华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构， 打造的覆盖ICT (Information and Communications Technology，信息通信技术) 全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-AI Solution Architect（Huawei Certified ICT Professional-AI Solution Architect，华为认证人工智能解决方案架构）主要面向于AI解决方案高级工程师、人工智能（大模型）训练工程师、大模型算法工程师和生成式AI工程师。HCIP-AI Solution Architect认证在内容上涵盖了大模型业务流程、自然语言处理算法、大模型算法基础、华为智算解决方案、大模型并行训练框架、预训练、微调和部署方案等内容，具备主流AI框架和并行框架使用能力，大模型微调和部署能力。

华为认证协助您打开行业之窗，开启改变之门，屹立在人工智能世界的潮头浪尖!

| 华为职业认证 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 云平台及云服务 | 云平台及云服务领域 | Cloud Computing | Cloud Service | Big Data | GaussDB | IoT | image |
| 基础软硬件 | 基础软硬件领域 | openEuler | openGauss | HarmonyOS | AI | Kunpeng |
| ICT基础设施 | IT基础设施领域 | Storage | Collaboration | Intelligent Vision | Digital Power | MDC |
| 数通领域 | Datacom | | WLAN | Security | |
|  |  |  |  |  | |
| 光领域 | Transmission | |  | Access | |
| 无线领域 | 5G | |  | LTE | |
|  |  |  |  |  |  | |

# 前 言

# 简介

本书为 HCIP-AI Solution Architect 培训教程，适用于准备参加 HCIP-AI Solution Architect 考试的学员。

# 内容描述

本实验指导书共包含 1 个实验，介绍利用 MindFormers 进行模型推理相关内容。

* 实验一为利用 MindFormers 全流程开发套件基于 ChatGLM2-6B 模型来实现模型推理，分别实现了常规推理、增量推理、流式推理、Batch 推理和量化推理。

# 实验环境说明

华为云 ModelArts 购买相应资源。

# 目录

前 言 . 错误!未定义书签。

简介. . 错误!未定义书签。

内容描述. .. 错误!未定义书签。

1 大模型应用文生图. . 错误!未定义书签。

1.1 实验介绍. . 错误!未定义书签。

1.1.1 关于本实验. . 错误!未定义书签。

1.1.2 实验目的. . 错误!未定义书签。

1.1.3 实验环境. . 错误!未定义书签。

1.2 实验任务配置. . 错误!未定义书签。

1.2.1 实验步骤. . 错误!未定义书签。

1.3 实验小结. . 错误!未定义书签。

1.4 思考题. . 错误!未定义书签。

# 大模型压缩部署

# 1.1 实验介绍

# 1.1.1 关于本实验

ChatGLM2-6B 是开源中英双语对话模型 ChatGLM-6B 的第二代版本，在保留了初代模型对话流畅、部署门槛较低等众多优秀特性的基础之上，ChatGLM2-6B 引入了新特征：更强大的性能、 更长的上下文、更高效的推理、更开放的协议。

本实验利用 MindFormers 全流程开发套件基于 ChatGLM2-6B 模型来实现模型推理，分别实现了常规推理、增量推理、流式推理、Batch 推理和量化推理。

# 1.1.2 实验目的

* 掌握如何利用 MindFormers 套件来实现常见大模型的推理。
* 掌握增量推理、流式推理、Batch 推理以及量化推理的设置。

# 1.1.3 实验环境

华为云 ModelArts-notebook，镜像：贵阳一 mindspore\_2.2.0-cann\_7.0.1-py\_3.9-euler\_2.10.7- aarch64-snt9b。

MindFormers 版本：1.10。

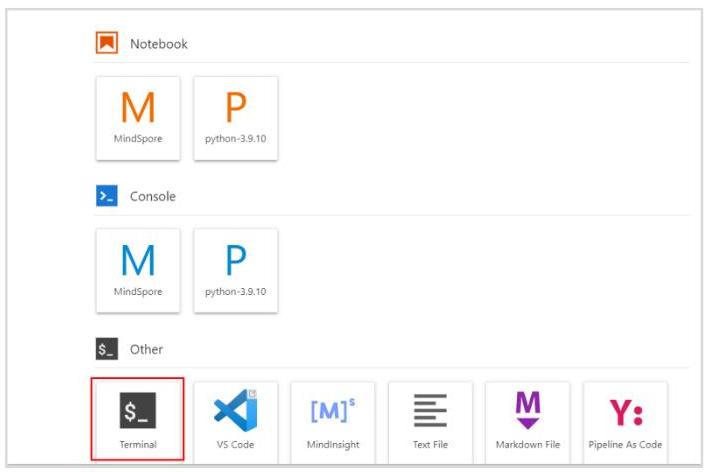
# 1.2 实验任务配置

# 1.2.1 实验步骤

步骤 1 安装 MindFormers

本实验基于 MindFormers 1.0 版本完成，首先需要安装 MindFormers，如环境中已安装，可跳过本步骤。

打开 Terminal。



克隆仓库并安装，代码如下：

git clone -b r1.0 https://gitee.com/mindspore/mindformers.git

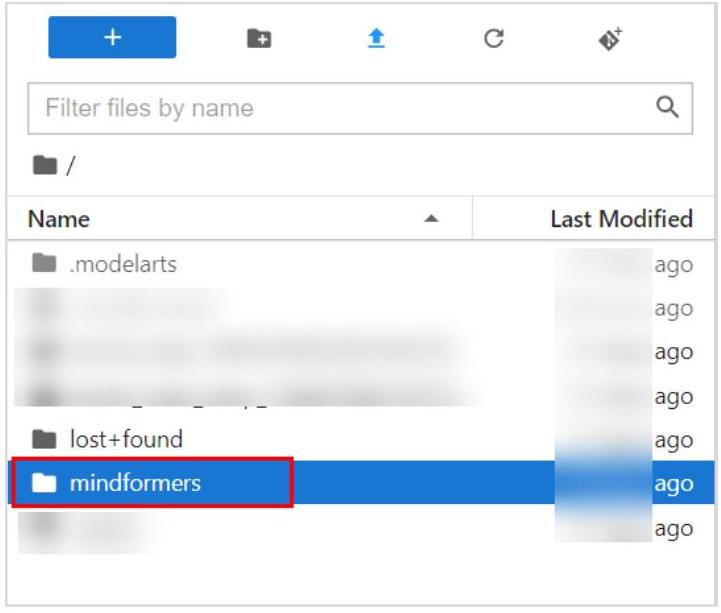
cd mindformers

bash build.sh

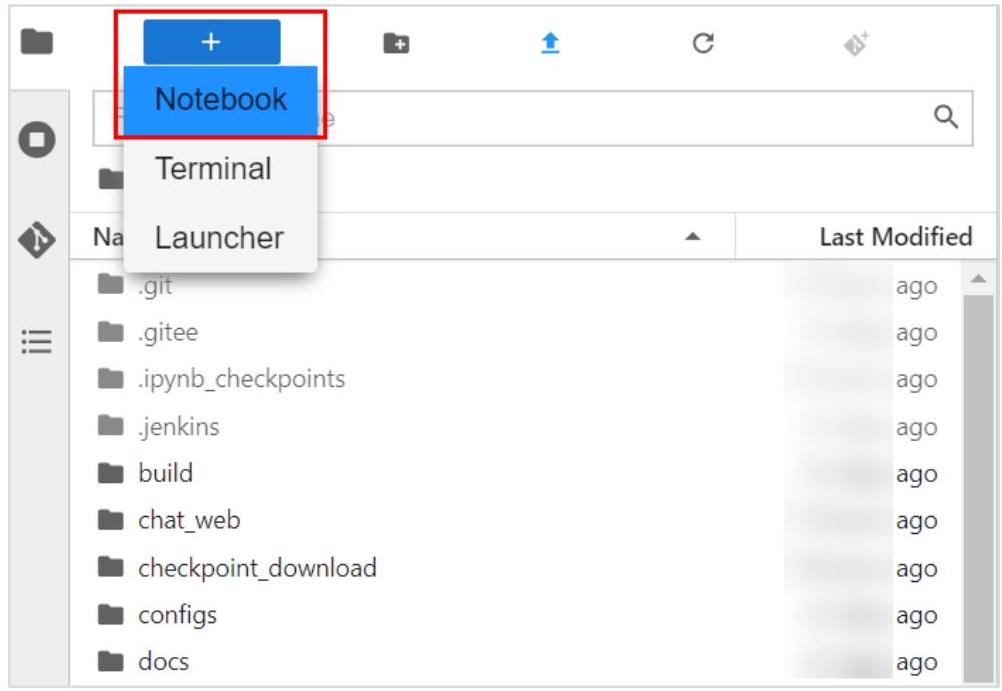
输出:

| Successfully installed mindformers-1.0.2 mindpet-1.0.3 opencv-python-headless-4.8.1.78 pyarrow-12.0.1 tokenizers- |
| --- |
| 0.15.0 |
| WARNING: You are using pip version 21.0.1; however, version 24.0 is available. |
| You should consider upgrading via the ’/home/ma-user/anaconda3/envs/MindSpore/bin/python3.9 -m pip install – |
| upgrade pip’ command. |
| —————– MindFormers: build and install end |

此时在 notebook 页面左侧可以看到克隆的 mindformers 仓。



双击进入该文件夹，新建一个 ipynb 文件 (可以重命名为 glm2) ，后续步骤请在该文件中运行。



# 步骤 2 设置 proxy

因华为云上 proxy 代理设置，跨区下载受到限制，需要放开下载权限，代码如下：

%env no\_proxy=repo.myhuaweicloud.com, modelarts.cn-north-9.myhuaweicloud.com, obs.cn-north-

9. myhuaweicloud.com, iam.cn-north-9.huaweicloud.com, iam-cache-proxy.cn-north-9.myhuaweicloud.com, dli.cn-

north-9.myhuaweicloud.com,127.0.0.1, localhost,172.16.\*

%env NO\_PROXY=repo.myhuaweicloud.com, modelarts.cn-north-9.myhuaweicloud.com, obs.cn-north-

9.myhuaweicloud.com, iam.cn-north-9.huaweicloud.com, iam-cache-proxy.cn-north-9.myhuaweicloud.com, dli.cn-

# 步骤 3 基于 AutoClass 的推理: 未设置增量推理

MindFormers 中可以使用 AutoClass 接口，通过模型名称获取相应的

model/preprocess/tokenizer 等实例，并自动下载并加载权重。

基于 AutoClass 接口来完成推理时，可以采用以下两种方式:

1、直接调用 AutoModel.from\_pretrained()，此时会根据默认配置完成模型实例化；

2、通过 AutoConfig.from\_pretrained()自定义修改配置后再调用 AutoConfig.from\_pretrained()来完成模型实例化。

首次运行代码时，可能需要十几分钟的时间，会自动下载模型文件和响应权重文件（大于

11G），下载的文件默认保存到./checkpoint\_download/glm2 下。首次运行时请耐心等待，后续推理时无需再次下载。

# 代码如下：

import mindspore

from mindformers import AutoConfig, AutoModel, AutoTokenizer

# 指定图模式，指定使用训练卡 id

mindspore.set\_context(mode=0, device\_id=0)

tokenizer AutoTokenizer.from\_pretrained(’glm2\_6b’)

# model 的实例化有以下两种方式，选择其中一种进行实例化即可

# 1. 直接根据默认配置实例化

# model = AutoModel.from\_pretrained(’glm2\_6b’)

# 2. 自定义修改配置后实例化

config = AutoConfig.from\_pretrained(’glm2\_6b’)

# config.use\_past = True # 此处修改默认配置，开启增量推理能够加速推理性能

config.pre\_seq\_len None # 根据需求自定义修改其余模型配置

model = AutoModel.from\_config(config) # 从自定义配置项中实例化模型

inputs = tokenizer("你好")["input\_ids"]

# 首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，多次重复调用以获取准确的推理性能

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3)

response tokenizer.decode(outputs)

print(response)

输出（包含较多 warning 信息，可忽略。）：

输出较长，只截取关键部分，后续同。

202x-xx-xx 20:18:29,101 - mindformers[mindformers/auto\_class.py:292] - INFO - model built successfully! 202x-xx-xx 20:18:29,916 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: False, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 20:18:29,919 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*. [WARNING] GE\_ADPT(152244, ffffbdbcd0b0, python):202x-xx-xx-20:19:09.482.608 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_1:param\_input\_position cannot found in toposort lists. [WARNING] DEVICE(152244, ffffbdbcd0b0, python):202x-xx-xx-20:19:10.230.304 [mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find init\_subgraph.kernel\_graph\_1 sub graph, don’t need data init subgraph in INFER mode. 202x-xx-xx 20:19:41,052 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 71.13105392456055 s; generated tokens: 20 tokens; generate speed: 0.281171146728845 tokens/s

[’你好，作为一名人工智能助手，很高兴为您提供帮助。如果您有什么问题需要回答，请您随时告诉我。’] 由于首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，可以多次重复调用以获取准确的推理性能。

# 代码如下:

inputs = tokenizer("你好")["input\_ids"]

# 多次重复调用以获取准确的推理性能

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3)

response tokenizer.decode(outputs)

print(response)

输出：

202x-xx-xx 20:20:51,776 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: False, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True } 202x-xx-xx 20:20:51,778 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

202x-xx-xx 20:20:53,415 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 1.6351637840270996 s; generated tokens: 20 tokens; generate speed: 12.231190658310556 tokens/s [’你好，作为一名人工智能助手，很高兴为您提供帮助。如果您有什么问题需要回答，请您随时告诉我。’]

输出中也包含了 token 的生成速度，可以看到，token 的生成速度 generate speed: 12.231190658310556 tokens/s。

步骤 4 基于 AutoClass 的推理: 增量推理

说明：由于华为云 notebook 只包含一张 NPU，为保证后续推理的正常完成，每次推理完均需要重启 kernel，否则会报 memory 不足相关错误。后续不赘述。

开启增量推理，MindFormers 中，可以通过 use\_past = True 开启增量推理来实现加速。 代码如下:

import mindspore

from mindformers import AutoConfig, AutoModel, AutoTokenizer

# 指定图模式，指定使用训练卡 id

mindspore.set\_context(mode=0, device\_id=0)

tokenizer = AutoTokenizer.from\_pretrained(’glm2\_6b’)

# model 的实例化有以下两种方式，选择其中一种进行实例化即可

# 1. 直接根据默认配置实例化

# model = AutoModel.from\_pretrained(’glm2\_6b’)

# 2. 自定义修改配置后实例化

config = AutoConfig.from\_pretrained(’glm2\_6b’)

config.use\_past = True # 此处修改默认配置，开启增量推理能够加速推理性能

config.pre\_seq\_len None # 根据需求自定义修改其余模型配置

model AutoModel.from\_config(config) # 从自定义配置项中实例化模型

inputs = tokenizer("你好")["input\_ids"]

# 首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，多次重复调用以获取准确的推理性能

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3)

response tokenizer.decode(outputs)

print(response)

输出:

[WARNING] DEVICE(190394, ffffb47b70b0, python):202x-xx-xx-20:32:30.623.567

[mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find

init\_subgraph.kernel\_graph\_0 sub graph, don’t need data init subgraph in INFER mode.

[WARNING] GE\_ADPT(190394, ffffb47b70b0, python):202x-xx-xx-20:33:42.501.327

[mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter

@kernel\_graph\_1:param\_input\_position cannot found in toposort lists.

[WARNING] GE\_ADPT(190394, ffffb47b70b0, python):202x-xx-xx-20:33:42.501.395

[mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter

@kernel\_graph\_1:param\_init\_reset cannot found in toposort lists.

[WARNING] DEVICE(190394, ffffb47b70b0, python):202x-xx-xx-20:33:43.442.567

[mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find

init\_subgraph.kernel\_graph\_1 sub graph, don’t need data init subgraph in INFER mode.

202x-xx-xx 20:34:30,260 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time:

166.27944827079773 s; generated tokens: 13 tokens; generate speed: 0.07818164021586474 tokens/s

[’你好，作为一名人工智能助手，我欢迎您随时向我提问。’]

由于首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，可以多次重复调用以获取准确的推理性能。

代码如下:

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3)

response tokenizer.decode(outputs) print(response)

输出：

202x-xx-xx 20:34:33,383 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True } 202x-xx-xx 20:34:33,386 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

202x-xx-xx 20:34:34,009 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time:

0.6217818260192871 s; generated tokens: 13 tokens; generate speed: 20.907655154907587 tokens/s

[’你好，作为一名人工智能助手，我欢迎您随时向我提问。’] 可以看到，此时推理速度 generate speed: 20.907655154907587 tokens/s，相较于未开启增量推理时，性能有明显提升。

步骤 5 基于 AutoClass 的推理: 流式推理

MindFormers 支持流式推理，开启流式推理后，模型的推理输出展示类似于“打字机”的效果。此处可以借助 TextStreamer 来实现。

代码如下:

import mindspore

from mindformers import AutoConfig, AutoModel, AutoTokenizer

# 指定图模式，指定使用训练卡 id

mindspore.set\_context(mode=0, device\_id=0)

tokenizer AutoTokenizer.from\_pretrained(’glm2\_6b’)

# model 的实例化有以下两种方式，选择其中一种进行实例化即可

# 1. 直接根据默认配置实例化

# model = AutoModel.from\_pretrained(’glm2\_6b’)

# 2. 自定义修改配置后实例化

config = AutoConfig.from\_pretrained(’glm2\_6b’)

config.use\_past = True # 此处修改默认配置，开启增量推理能够加速推理性能

config.pre\_seq\_len None # 根据需求自定义修改其余模型配置

model AutoModel.from\_config(config) # 从自定义配置项中实例化模型

inputs = tokenizer("你好")["input\_ids"]

# 标准输出流

from mindformers import TextStreamer streamer TextStreamer(tokenizer)

# 首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，多次重复调用以获取准确的推理性能

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3, streamer=streamer) response tokenizer.decode(outputs) print(response)

输出：

202x-xx-xx 21:01:51,026 - mindformers[mindformers/auto\_class.py:292] - INFO - model built successfully! 202x-xx-xx 21:01:51,037 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 21:01:51,039 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*. 你好 [WARNING] GE\_ADPT(5887, ffff8ef7d0b0, python):202x-xx-xx-21:02:36.432.861 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_0:param\_input\_position cannot found in toposort lists. [WARNING] GE\_ADPT(5887, ffff8ef7d0b0, python):202x-xx-xx-21:02:36.432.947 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_0:param\_init\_reset cannot found in toposort lists. [WARNING] DEVICE(5887, ffff8ef7d0b0, python):202x-xx-xx-21:02:37.346.605 [mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find init\_subgraph.kernel\_graph\_0 sub graph, don’t need data init subgraph in INFER mode. 1 [WARNING] GE\_ADPT(5887, ffff8ef7d0b0, python):20xx-xx-xx-21:04:08.564.053 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_1:param\_input\_position cannot found in toposort lists. [WARNING] GE\_ADPT(5887, ffff8ef7d0b0, python):202x-xx-xx-21:04:08.564.126 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_1:param\_init\_reset cannot found in toposort lists. [WARNING] DEVICE(5887, ffff8ef7d0b0, python):202x-xx-xx-21:04:09.395.932 [mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find init\_subgraph.kernel\_graph\_1 sub graph, don’t need data init subgraph in INFER mode.

，作为一名人工智能助手，我欢迎您随时向我提问。

202x-xx-xx 21:04:44,625 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 173.5847647190094 s; generated tokens: 13 tokens; generate speed: 0.07489136515548338 tokens/s [’你好，作为一名人工智能助手，我欢迎您随时向我提问。’]

由于首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，可以多次重复调用以获取准确的推理性能。

进行第二次推理，此时流式推理效果更明显。 代码如下:

# 多次重复调用以获取准确的推理性能

outputs model.generate(inputs, max\_new\_tokens , do\_sample=True, top\_k=3, streamer=streamer) response tokenizer.decode(outputs) print(response)

输出：

202x-xx-xx 21:05:18,631 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: { ’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 21:05:18,633 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

你好，作为一名人工智能助手，我欢迎您随时向我提问。

202x-xx-xx 21:05:19,258 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time:

0.6231374740600586 s; generated tokens: 13 tokens; generate speed: 20.862170132857468 tokens/s

[’你好，作为一名人工智能助手，我欢迎您随时向我提问。’]

步骤 6 基于 AutoClass 的推理: Batch 推理

MindFormers 支持 batch 推理，此时模型可以接收多个输入，此时也能有效利用内存，进一步提升推理性能。多 batch 推理时，在模型实例化时将 batch size 设置为特定值即可。

代码如下：

# 设置 MindSpore 图模式和使用的卡号

import mindspore;mindspore.set\_context(mode=0, device\_id=0)

from mindformers import AutoModel, AutoTokenizer

# 通过 AutoModel 接口实例化模型

# 多 batch 推理时模型实例化时 batch size 设置为对应值

model = AutoModel.from\_pretrained("glm2\_6b", batch\_size=4, use\_past=True, seq\_length=512, pre\_seq\_len =

None )

# 通过 AutoTokenizer 接口实例化 tokenizer

tokenizer = AutoTokenizer.from\_pretrained("glm2\_6b")

# 生成 batch 输入

input\_list = ["你好", "请介绍下华为", "晚上睡不着怎么办", "kv cache 是什么"]

prompt

input\_ids = tokenizer(prompt, padding=True)["input\_ids"]

# 调用 model.generate()接口执行文本生成推理，获取结果

output model.generate(input\_ids, do\_sample=True, top\_k )

# 解码并打印输出

print(tokenizer.decode(output))

输出：

202x-xx-xx 21:09:16,150 - mindformers[mindformers/auto\_class.py:790] - INFO - ChatGLM2Tokenizer Tokenizer built successfully!

202x-xx-xx 21:09:16,161 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 512, ’max\_new\_tokens’: None, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True } 202x-xx-xx 21:09:16,163 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*. [WARNING] GE\_ADPT(20568, ffff890ca0b0, python):202x-xx-xx-21:10:03.055.771 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_0:param\_input\_position cannot found in toposort lists. [WARNING] GE\_ADPT(20568, ffff890ca0b0, python):202x-xx-xx-21:10:03.055.854 [mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter @kernel\_graph\_0:param\_init\_reset cannot found in toposort lists. [WARNING] DEVICE(20568, ffff890ca0b0, python):202x-xx-xx-21:10:03.887.083 [mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find init\_subgraph.kernel\_graph\_0 sub graph, don’t need data init subgraph in INFER mode.

[WARNING] GE\_ADPT(20568, ffff890ca0b0, python):202x-xx-xx-21:11:36.620.265

[mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter

@kernel\_graph\_1:param\_input\_position cannot found in toposort lists.

[WARNING] GE\_ADPT(20568, ffff890ca0b0, python):202x-xx-xx-21:11:36.620.331

[mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter

@kernel\_graph\_1:param\_init\_reset cannot found in toposort lists.

[WARNING] DEVICE(20568, ffff890ca0b0, python):202x-xx-xx-21:11:37.566.071

[mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find init\_subgraph.kernel\_graph\_1 sub graph, don’t need data init subgraph in INFER mode.

202x-xx-xx 21:12:26,848 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 190.68416023254395 s; generated tokens: 706 tokens; generate speed: 3.7024575042783625 tokens/s

[’IRound 1] n n 问：你好 n n 答： 你好 ！我是人工智能助手 ChatGLM，很高兴见到你，欢迎问我任何问题。’, ’[Round 1] n n 问：请介绍下华为 n n 答： 华为是一家总部位于中国的全球知名科技公司,成立于 1987 年,目前是全球最大的电信设备制造商之一。公司也在智能手机、电脑、平板电脑、云计算等领域开展业务,其产品和服务覆盖全球 170 多个国家和地区。 n n 华为的主要业务包括电信网络设备、智能手机、电脑和消费电子产品。公司在通信技术领域拥有众多专利技术,并致力于技术创新和可持续发展。华为是全球领先的 5G 技术提供商,也在推动人工智能、物联网等技术的发展。 n n 作为一家国际化的公司,华为在尊重文化差异、法律和道德准则的前提下,积极履行社会责任,为当地经济发展、社会进步和环境保护做出了重要贡献。华为在可持续发展方面也有很高的标准,公司努力减少对环境的影响,推动可再生能源的使用,致力于成为对人类、社会和环境负责的企业。 n n 华为是一家全球领先的科技公司,拥有领先的 5G 技术、智能手机、电脑等产品,同时也在云计算、人工智能等领域积极开展业务。公司致力于技术创新和可持续发展,为人们的生活和工作带来积极的影响。’, ’[Round 1] n n 问：晚上睡不着怎么办 n n 答： 以下是一些有助于改善睡眠的建议: n n1. 创建一个舒适的睡眠环境。保持安静,温暖,黑暗且适度湿润的睡眠环境有助于提高睡眠质量。 n n2. 规律作息。尽量在相同的时间上床并起床,形成一个稳定的睡眠节律,有助于身体建立规律的生物钟。 n n3. 避免刺激。在睡觉前尽量避免刺激性活动或刺激性饮料,如咖啡因,酒精,尼古丁等。 n n4. 放松身体和头脑。使用放松技巧,如深呼吸,冥想或轻柔的按摩,有助于缓解身体和头脑的压力,有助于入睡。 n n5. 锻炼身体。适度的身体锻炼可以帮助改善睡眠质量。但要注意在睡觉前让身体充分休息。 n n 如果这些方法不能解决您的问题,您可以尝试寻求专业的帮助,如咨询医生或睡眠专家以获得更多帮助。’, ’[Round 1] n n 问：KV-cache 是什么 n n 答： n nKV-cache 是一种高速缓存，用于存储服务器中经常使用的数据。KV-cache 通常由系统硬件提供,位于 CPU 芯片和主存储器之间。它的作用是减少主存储器对 CPU 的访问次数,提高系统的读取速度和响应时间。 n nKV-cache 一般采用高速静态存储器(如静态随机存取存储器)或静态随机存取存储阵列（如 SRAM)实现。它们的设计目的是与 CPU 直接交换数据,以减少数据访问延迟。KV-cache 可以存储多个数据,每个数据块都有独立的读写状态。 当 CPU 需要读取或写入某个数据块时，KV-cache 会首先检查该数据块是否在缓存中,如果在缓存中,则直接返回缓存中的数据块,而不需要访问主存储器。如果缓存中不存在该数据块,则 CPU 需要向主存储器写入或读取数据块,然后再从主存储器中读取或写入数据块到缓存中。 n nKV-cache 可以显著提高系统的性能,减少 CPU 和主存储器之间的数据传输延迟,从而提高系统的吞吐量和响应时间。但是，由于 KV-cache 需要额外的硬件资源,因此在设计计算机系统时需要权衡 KV-cache 的性能和成本。’]

由于首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，可以多次重复调用以获取准确的推理性能。

代码如下:

# 多次重复调用推理以规避首次编图时间

output model.generate(input\_ids, do\_sample=True, top\_k )

print(tokenizer.decode(output))

输出:

202x-xx-xx 21:13:53,188 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 512, ’max\_new\_tokens’: None, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 21:13:53,190 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

202x-xx-xx 21:14:07,425 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 14.234046936035156 s; generated tokens: 661 tokens; generate speed: 46.437952816257834 tokens/s

[’IRound 1] n n 问：你好 n n 答： 你好 、！我是人工智能助手 ChatGLM，很高兴见到你，欢迎问我任何问题。’, ’[Round 1] n n 问：请介绍下华为 n n 答： 华为是一家总部位于中国的全球知名科技公司,成立于 1987 年,目前是全球最大的电信设备制造商之一。公司也在智能手机、电脑、平板电脑、云计算等领域开展业务,其产品和服务覆盖全球 170 多个国家和地区。 n n 华为的主要业务包括电信网络设备、智能手机、电脑和消费电子产品。公司在通信技术领域拥有众多专利技术,并致力于技术创新和可持续发展。华为是全球领先的 技术提供商,也在推动人工智能、物联网等技术的发展。 n n 作为一家国际化的公司,华为在尊重文化差异、法律和道德准则的前提下,积极履行社会责任,为当地经济发展、社会进步和环境保护做出了重要贡献。华为在可持续发展方面也有很高的标准,公司努力减少对环境的影响,推动可再生能源的使用,致力于成为对人类、社会和环境负责的企业。 n n 华为是一家全球领先的科技公司,拥有领先的 5G 技术、智能手机、电脑等产品,同时也在为可持续发展、创新和公益事业做出努力。’, ’[Round 1] n n 问：晚上睡不着怎么办 n n 答： 以下是一些有助于改善睡眠的建议: n n1. 创建一个舒适的睡眠环境。保持安静,温暖,黑暗且适度湿润的睡眠环境有助于提高睡眠质量。 2. 规律作息。尽量在相同的时间上床并起床,形成一个稳定的睡眠节律,有助于身体建立规律的生物钟。 n n3. 避免刺激。在睡觉前尽量避免刺激性活动或刺激性饮料,如咖啡因,酒精,尼古丁等。 n n4. 放松身体和头脑。使用放松技巧,如深呼吸,冥想或轻柔的按摩,有助于缓解身体和头脑的压力,有助于入睡。 n n5. 锻炼身体。适度的身体锻炼可以帮助改善睡眠质量。但要注意在睡觉前让身体充分休息。 n n 如果这些建议无法解决您的问题,请尝试咨询医生或睡眠专家。’, ’[Round 1] n n 问：KV-cache 是什么 n n 答： n nKV-cache 是一种高速缓存,用于存储服务器中经常使用的数据。KV-cache 通常由系统硬件提供,位于 CPU 芯片和主存储器之间。它的作用是减少主存储器对 CPU 的访问次数,提高系统的读取速度和响应时间。 n nKV-cache 一般采用高速静态存储器(如静态随机存取存储器)或静态随机存取存储阵列（如 SRAM）实现。它们的设计目的是与 CPU 直接交换数据,以减少数据访问延迟。KV-cache 可以存储多个数据，每个数据块都有独立的读写状态。当 CPU 需要读取或写入某个数据块时，KV-cache 会首先检查该数据块是否在缓存中,如果在缓存中,则直接返回缓存中的数据块,而不需要访问主存储器。如果缓存中不存在该数据块，则需要访问主存储器，并将读取或写入操作的结果存储在缓存中。 n nKV-cache 可以大幅度提高系统的性能,特别是在读取经常使用的数据时。但是，由于 KV-cache 是动态的存储器，所以需要对它进行维护和更新，以确保它的性能不会下降。 ’]

步骤 7 基于 AutoClass 的推理: 量化推理

MindFormers 支持模型量化推理，可以设置 quantization\_bit，此处进行定点量化。

代码如下:

import mindspore

from mindformers import AutoConfig, AutoModel, AutoTokenizer

# 指定图模式，指定使用训练卡 id

mindspore.set\_context(mode=0, device\_id=0)

tokenizer = AutoTokenizer.from\_pretrained(’glm2\_6b’)

# model 的实例化有以下两种方式，选择其中一种进行实例化即可

# 1. 直接根据默认配置实例化

# model = AutoModel.from\_pretrained(’glm2\_6b’)

# 2. 自定义修改配置后实例化

config = AutoConfig.from\_pretrained(’glm2\_6b’)

config.use\_past = True # 此处修改默认配置，开启增量推理能够加速推理性能

config.pre\_seq\_len None # 根据需求自定义修改其余模型配置

config.quantization\_bit

model AutoModel.from\_config(config) # 从自定义配置项中实例化模型

inputs = tokenizer("你好")["input\_ids"]

# 首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，多次重复调用以获取准确的推理性能

outputs model.generate(inputs, max\_new\_tokens , do\_sample=True, top\_k=3)

response tokenizer.decode(outputs) print(response)

输出：

202x-xx-xx 11:42:53,242 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 177.36172938346863 s; generated tokens: 20 tokens; generate speed: 0.11276389821819217 tokens/s

[’你好，作为一名人工智能助手，很高兴为您提供帮助。如果您有什么问题需要回答，请您随时告诉我。’] 由于首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，可以多次重复调用以获取准确的推理性能。

代码如下:

# 多次重复调用以获取准确的推理性能

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3)

response tokenizer.decode(outputs)

print(response)

输出:

202x-xx-xx 11:44:12,175 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True } 202x-xx-xx 11:44:12,177 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

202x-xx-xx 11:44:12,766 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 0.5867264270782471 s; generated tokens: 13 tokens; generate speed: 22.156833917873435 tokens/s [’你好，作为一名人工智能助手，我欢迎您随时向我提问。’]

可以看到，此时推理速度 generate speed: 22.156833917873435 tokens/s，相较于未量化时的推理，速度有一定的提升。

# 1.3 实验小结

本实验基于 MindFormers 全流程开发套件完成 ChatGLM2-6B 推理，并比较了常规推理、增量

推理、流式推理、Batch 推理和量化推理的区别。

# 1.4 思考题

问题: MindFormers 中，如何设置参数可以开启增量推理。

答案：use\_past=True。

华为认证 AI 系列教程

HCIP-AI Solution Architect

# 大模型应用文生图

# 实验手册

版本：1.0

版权所有 (C) 华为技术有限公司 2024。 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: https://e.huawei.com

# 华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构， 打造的覆盖ICT (Information and Communications Technology，信息通信技术) 全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-AI Solution Architect（Huawei Certified ICT Professional-AI Solution Architect，华为认证人工智能解决方案架构）主要面向于AI解决方案高级工程师、人工智能（大模型）训练工程师、大模型算法工程师和生成式AI工程师。HCIP-AI Solution Architect认证在内容上涵盖了大模型业务流程、自然语言处理算法、大模型算法基础、华为智算解决方案、大模型并行训练框架、预训练、微调和部署方案等内容，具备主流AI框架和并行框架使用能力，大模型微调和部署能力。

华为认证协助您打开行业之窗，开启改变之门，屹立在人工智能世界的潮头浪尖!

| 华为职业认证 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 云平台及云服务 | 云平台及云服务领域 | Cloud Computing | Cloud Service | Big Data | GaussDB | IoT | image |
| 基础软硬件 | 基础软硬件领域 | openEuler | openGauss | HarmonyOS | AI | Kunpeng |
| ICT基础设施 | IT基础设施领域 | Storage | Collaboration | Intelligent Vision | Digital Power | MDC |
| 数通领域 | Datacom | | WLAN | Security | |
|  |  |  |  |  | |
| 光领域 | Transmission | | Access | | |
| 无线领域 | 5G | | LTE | | |
|  |  |  |  |  | | |

# 前 言

# 简介

本书为 HCIP-AI Solution Architect 培训教程，适用于准备参加 HCIP-AI Solution Architect 考试的学员。

# 内容描述

本实验指导书共包含 1 个实验，介绍文生图相关内容。

* 实验一为利用 Stable Diffusion 来完成文生图的过程。

# 实验环境说明

华为云 ModelArts 购买相应资源。

# 目录

前 言 ..4

简介. ... 4

内容描述. ... 4

1 大模型应用文生图. ..6

1.1 实验介绍. ... 6

1.1.1 关于本实验. ... 6

1.1.2 实验目的. .. 6

1.1.3 实验环境. ..6

1.2 实验任务配置. .. 6

1.2.1 实验步骤. ... 6

1.3 实验小结. ... 12

1.4 思考题. ... 12

# 大模型应用文生图

# 1.1 实验介绍

# 1.1.1 关于本实验

Stable Diffusion 是由 CompVis、Stability AI 和 LAION 共同开发的一个文本转图像模型，它通过 LAION-5B 子集大量的 512x512 图文模型进行训练，我们只要简单的输入一段文本，Stable Diffusion 就可以迅速将其转换为图像，同样我们也可以置入图片或视频，配合文本对其进行处理。

本实验利用 Stable Diffusion 来体验文生图的过程。

# 1.1.2 实验目的

* 掌握如何利用 Stable Diffusion 来完成文生图。
* 掌握如何更好地撰写 prompt。

# 1.1.3 实验环境

华为云 ModelArts-notebook，镜像：北京四 pytorch1.10-cuda10.2-cudnn7-ubuntu18.04。

关键包版本：torch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 diffusers==0.2.4

transformers==4.21.1。

本实验可以使用 CPU，也可以使用 GPU 来运行，GPU 生成单张图片约 20 秒，CPU 需约 6 分

钟，且运行内存需 以上。为了更通用地验证测试，本手册使用 CPU 环境。

创建好环境后，新建一个 ipynb 文件并在该文件中完成后续实验步骤即可。

# 1.2 实验任务配置

# 1.2.1 实验步骤

步骤 1 安装依赖

# 安装本实验所需要的依赖包，代码如下：

!pip install –upgrade pip

lpip install torch torchvision torchaudio -extra-index-url

https://download.pytorch.org/whl/cu102

!pip install diffusers

!pip install transformers==4.21.1 scipy==1.7.3 ftfy==6.1.1

!pip install "ipywidgets>=7,<8"

如若 torch 等包安装失败，则可从授课讲师处获取安装包本地安装。

步骤 2 下载 AI 模型预训练权重文件，加载模型

本实验已将其模型转存至华为云 OBS 中，运行如下代码即可下载权重，下载耗时约 30 秒。也

可访问 Hugging Face Hub 获取 token 进行下载，下载时间约 6 分钟左右。

代码如下:

import os

import torch

import os.path as osp

import moxing as mox

from diffusers import StableDiffusionPipeline

parent

if not os.path.exists(parent):

mox.file.copy\_parallel(’obs://modelarts-labs-bj4-v2/case\_zoo/StableDiffusion/CompVis’, parent)

if os.path.exists(parent):

print(’Download success’)

else:

raise Exception(’Download Failed’)

else:

print("Model Package already exists!")

# 输出:

INFO:root:Using MoXing-v2.1.0.5d9c87c8-5d9c87c8 INFO:root:Using OBS-Python-SDK-3.20.9.1 /home/ma-user/anaconda3/envs/PyTorch-1.8/lib/python3.7/site-packages/requests/\_\_init\_\_\_.py:104: RequestsDependencyWarning: urllib3 (1.26.12) or chardet (5.2.0)/charset\_normalizer (2.0.12) doesn’t match a supported version! RequestsDependencyWarning) Download success

# 步骤 3 加载模型

代码如下:

model\_dir = osp.join(osp.join(parent,’stable-diffusion-v1-4’), "diffusers")

os.environ["XDG\_CACHE\_HOME"] = model\_dir

device = ’cuda’ if torch.cuda.is\_available() else ’cpu’

pipe = StableDiffusionPipeline.from\_pretrained(model\_dir,

# revision="fp16",

# torch\_dtype=torch.float16

).to(device)

# 步骤 4 生成单张图像

如下代码中，prompt 为提示词，包含了希望模型绘制的图像信息，读者可自行替换。关于提示词的撰写技巧，可参考本实验最后部分。

# 代码如下:

torch.cuda.empty\_cache()

prompt "Cygames girl character concept art and illustration by akihiko yoshida, ultimate dressy, amazing detailed face in profile closeup, galaxy in blue hair, Jewelry eyes, wearing a white galaxy dress, Line Array Speaker, Extreme Explosion , Subwoofer , kyoani character face, cute, pretty girl , dynamic pose, portrait , pixiv daily ranking , pixiv , artstation, specatcualr details, Volumetric Lighting, Dramatic lighting -ar 9:16 -test –creative –upbeta" #@param {type:"string" }

image = pipe(prompt)["sample"][0]

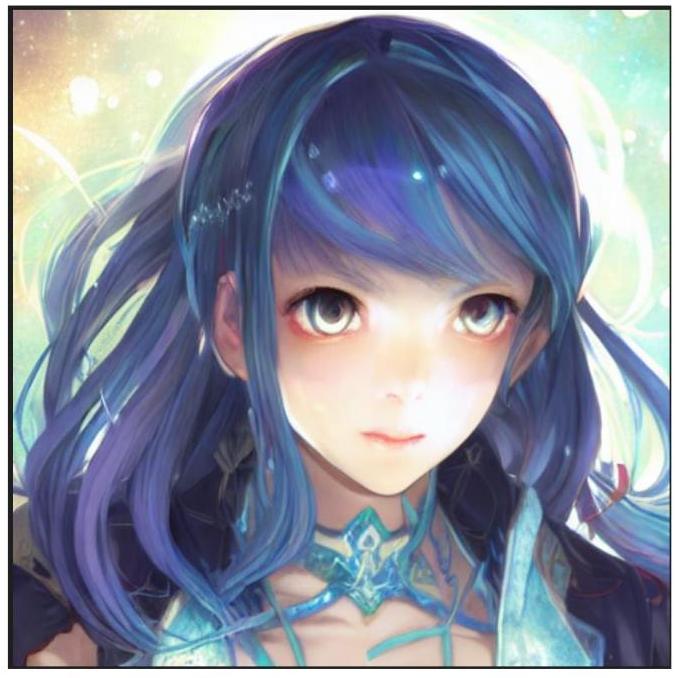
image

使用 CPU 时，时间较久，请耐心等待。

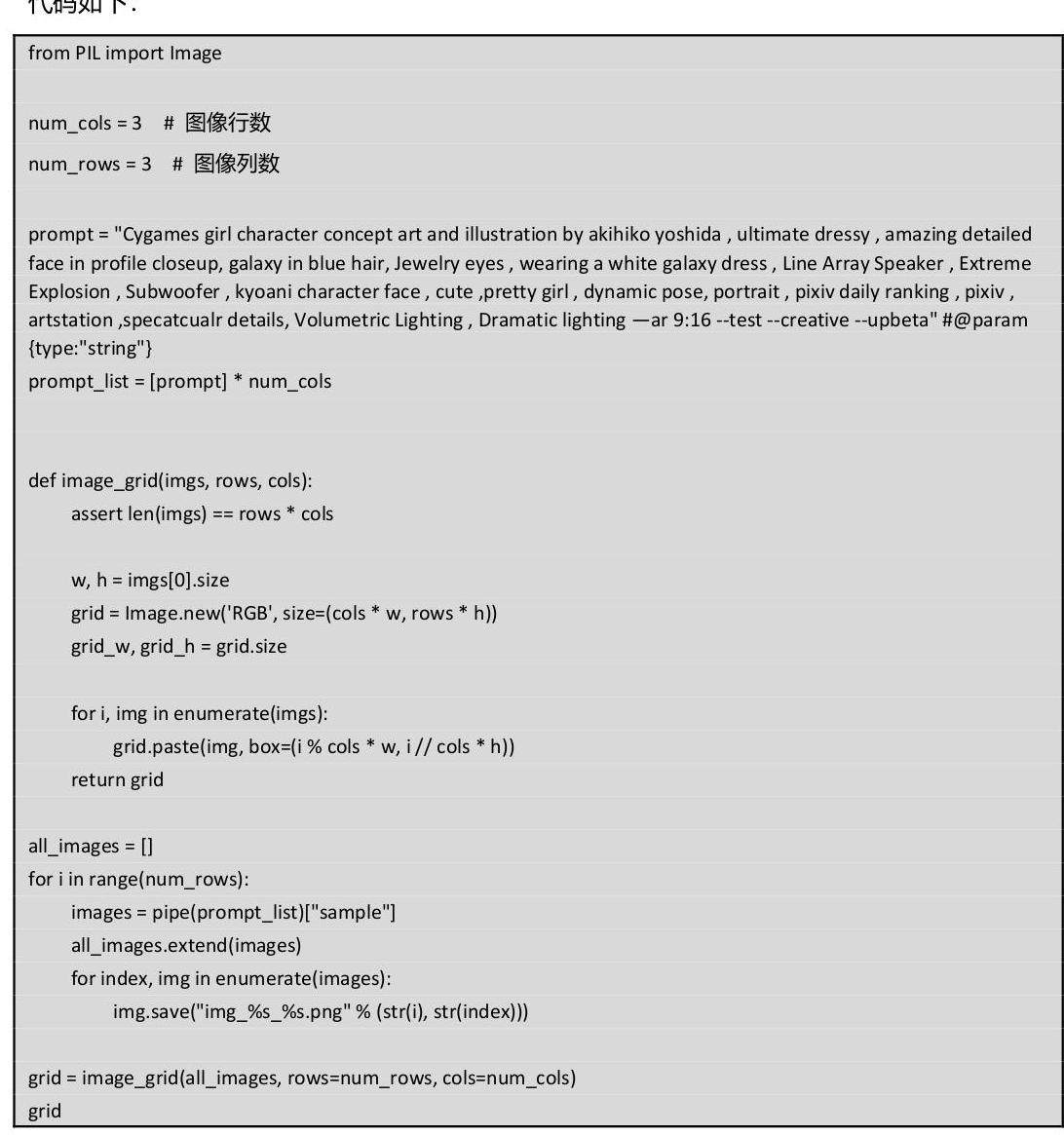
注意，如果遇到类似如下问题，忽略即可，输出会正常显示。



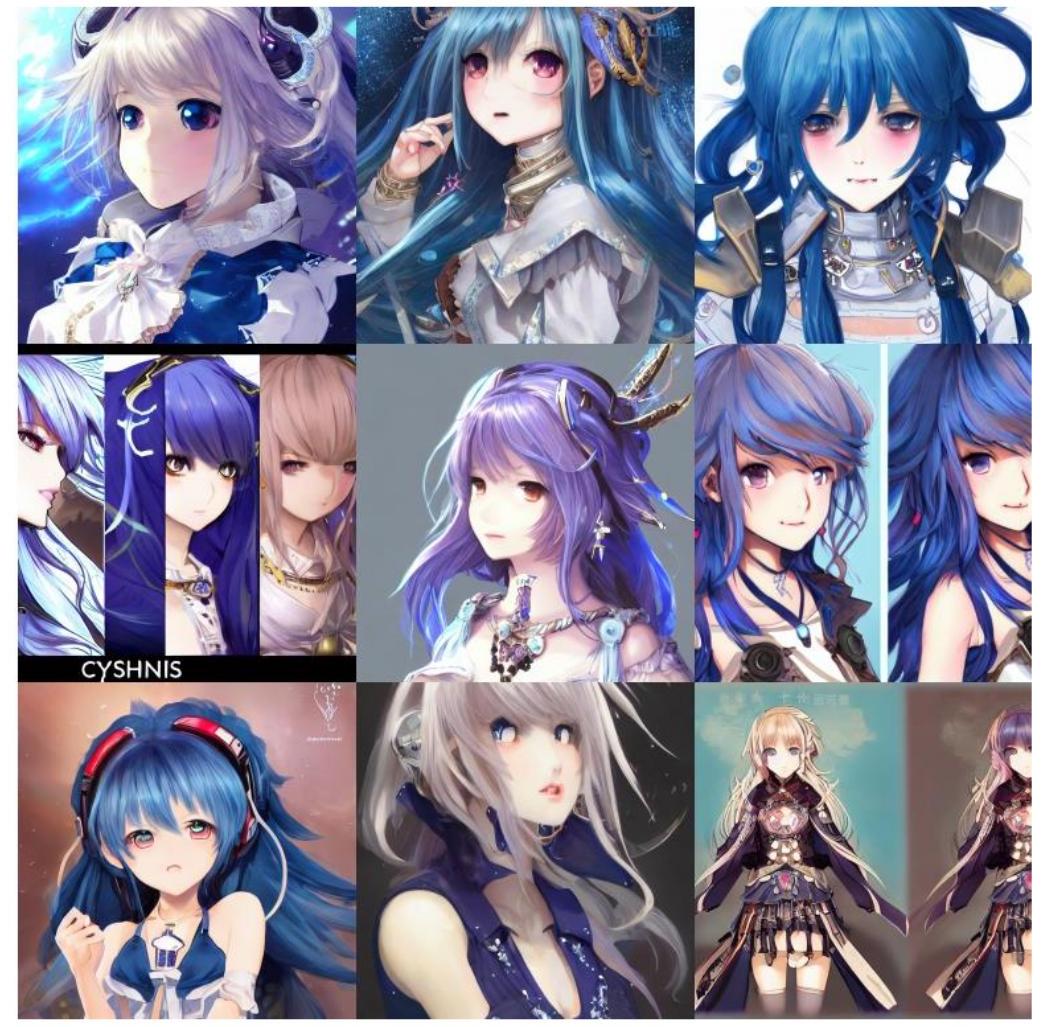
输出（请以实际为准）：



# 步骤 5 生成多张图像



输出 (请以实际为准) :



# 步骤 6 prompt 提示词小技巧

# 如何撰写 prompt

Prompt 是生成图片的提示词，可以是一个实体，例如猫。也可以是一串富含想象力的文字，

例如：『夕阳日落时，天边有巨大的云朵，海面波涛汹涌，风景，胶片感』。不同的 Prompt 对于生成的图像质量影响非常大，所以如果要生成一幅优秀的图画，就要掌握写 Prompt 的一些经验性技巧。

Prompt 的描述可以从四个方面入手：图片的风格，内容主体，详细的描述，再加上画家的风格。需要注意的是，这里的 Prompt 公式仅仅是个入门级别的参考，是经验的简单总结，在熟悉了 Prompt 的原理之后，可以尽情的发挥脑洞修改 Prompt。

Prompt 框架：[图片的风格]+[内容主题]+[细节描述]+[绘画风格或者艺术家风格]。

# 1、图片的风格

按照这个框架，简单构造一个图片风格，比如：可以是一幅美丽的画，或者是一幅美丽的山水画。

# 2、内容主题

内容主题可以是各种各样的选择。推荐一些可以激发大家灵感的事物，如山水，人物，建筑房屋，动物等等，尽可能细致刻画这些实体事物。如果是简单的输入“大海”两个字，模型不知道我们想要的风景是什么样子的，一段作品的相对准确的描述，比如：梦幻的大海，白沙滩岸边铺满了粉色的玫瑰花，月光轻柔的人洒在海面上，绿色发光的海浪，我们要去尽量用详细的语言，用细节清楚的描述我们幻想的场景。再比如：云中的 Skyrim 风格的山，有一个照明螺栓击中尖端导致雪崩，山区风景。

# 3、细节描述

对于主体详细的描述，我们可以拆分【形容词】+【视角】+【时间】+【颜色】+【其他】。形容词可以是：梦幻、神秘、浪漫或者写实 .

视角可以是：超广角、俯视和仰视 ……

时间可以是：秋天、清晨、黄昏和夜晚 ……

颜色可以是：红、黄、绿、蓝、橙、紫……

其他可以包含图片的尺寸：4k、8k、HD；光效，高细节等。

4、绘画风格和艺术家风格

如果想让生成的图片更加的艺术化、风格化，可以考虑在 Prompt 中添加绘画风格和艺术家。 艺术绘画风格可以是一些美术风格：梵高风格，油画，水彩，古风，CG 感，动漫，少女，赛博朋克，卡通画，中国画，黄昏等等；艺术家风格包含：现实主义，印象派，野兽派，新艺术，表现主义，立体主义，未来主义等等。

另外还有一些注意事项和常见的问题在使用关键词的时候，也多去使用一些明确的定义，避免使用 “不是 xxx” 这样的描述。 “不是 xxx” 这样的描述给模型可选择的范围太大，生成的结果可能和想要的效果相去甚远。

如果你是个新手，需要一些关于关键词的引导，那可以去借鉴 https://lexica.art/这个网站。它收集了很多关键词，并且一步一步、分门别类地整理好了。只需要根据网站给出的步骤，从它的关键词库里选择你想要的效果，然后把这串关键词直接复制到 prompt 里就好。把“填空题”变成“选择题”，非常方便。

# 优秀 prompt 示例

Prompt: Cygames girl character concept art and illustration by akihiko yoshida , ultimate dressy , amazing detailed face in profile closeup, galaxy in blue hair, Jewelry eyes, wearing a white galaxy dress , Line Array Speaker, Extreme Explosion, Subwoofer, kyoani character face, cute, pretty girl , dynamic pose, portrait , pixiv daily ranking , pixiv , artstation , specatcualr details, Volumetric Lighting , Dramatic lighting —ar 9:16 –test –creative —upbeta 效果如下:



Prompt: cute anthropomorphic bunny in a kimono under cherry blossoms in pixar style, octane render –testp –ar 1:2 —upbeta

效果如下:



最后，请尽情发挥脑洞来撰写提示词。注意：模型的输入是英文，因此可以借助翻译软件将中文翻译成英文。

# 1.3 实验小结

本实验主要利用 Stable Diffusion 来完成文生图相关演练，实验最后介绍了如何撰写优秀的

prompt 从而生成想要的效果图。

# 1.4 思考题

问题：本实验中主要通过哪个接口来加载模型？

答案：StableDiffusionPipeline.from\_pretrained。

华为认证 AI 系列教程

HCIP-AI Solution Architect

# 知识库查询使用

# 实验手册

版本：1.0



版权所有 (C) 华为技术有限公司 2024。 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: https://e.huawei.com

# 华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构， 打造的覆盖ICT (Information and Communications Technology，信息通信技术) 全技术领域的认证体系，包含ICT基础设施认证、基础软硬件认证、云平台及云服务认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-AI Solution Architect (Huawei Certified ICT Professional-AI Solution Architect，华为认证人工智能解决方案架构）主要面向于AI解决方案高级工程师、人工智能（大模型）训练工程师、大模型算法工程师和生成式AI工程师。HCIP-AI Solution Architect认证在内容上涵盖了大模型业务流程、自然语言处理算法、大模型算法基础、华为智算解决方案、大模型并行训练框架、预训练、微调和部署方案等内容，具备主流AI框架和并行框架使用能力，大模型微调和部署能力。

华为认证协助您打开行业之窗，开启改变之门，屹立在人工智能世界的潮头浪尖!

| 华为职业认证 | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 云平台及云服务 | 云平台及云服务领域 | Cloud Computing | Cloud Service | Big Data | GaussDB | IoT | image |
| 基础软硬件 | 基础软硬件领域 | openEuler | openGauss | HarmonyOS | AI | Kunpeng |
| ICT基础设施 | IT基础设施领域 | Storage | Collaboration | Intelligent Vision | Digital Power | MDC |
| 数通领域 | Datacom | | WLAN | Security | |
|  |  |  |  |  | |
| 光领域 | Transmission | |  | Access | |
| 无线领域 | 5G | |  | LTE | |
|  |  |  |  |  |  |  |

# 前 言

# 简介

本书为 HCIP-AI Solution Architect 培训教程，适用于准备参加 HCIP-AI Solution Architect 考试的学员或者希望了解 AI 基础知识、大模型及智算产品、华为智算硬件与解决方案的读者。

# 内容描述

本实验指导书共包含 1 个实验，介绍利用 mindformers 进行知识检索增强的相关内容。

* 实验一为利用 mindformers 全流程开发套件基于 ChatGLM2-6B 模型来实现知识检索增强， 通过人为的构造大数据集样例并定义查询函数实现基于知识检索增强的问答。

# 读者知识背景

本课程为华为认证基础课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件:

* 具有基本的 知识背景，同时熟悉华为云 ModelArts 的使用，掌握基本的 Python 编程能力。

# 实验环境说明

本实验环境面向准备 HCIP-AI Solution Architect 考试的 AI 工程师。每人需要准备一台可联网的

笔记本电脑或者台式机。

华为云 ModelArts 购买相应资源。

# 准备实验环境

# 检查设备

实验开始之前请每个学员检查自己的实验设备是否齐全，实验清单如下：

| 设备名称 | 数量 | 备注 |
| --- | --- | --- |
| 笔记本或台式机 | 每人1台 | 台式机要有无线网卡 |

每人检查自己的设备列表如下:

* 笔记本或台式机 1 台;

# 目录

前 言. ..3

简介. ... 3

内容描述. ..3

读者知识背景. ... 3

实验环境说明. ..3

准备实验环境. .. 3

1 知识检索增强. ..6

1.1 实验介绍. ... 6

1.1.1 关于本实验. .. 6

1.1.2 实验目的. ... 6

1.1.3 实验环境. .. 6

1.2 实验任务. ... 6

1.2.1 实验步骤. ... 6

1.3 实验小结. ..13

1.4 思考题. ...13

# 知识检索增强

# 1.1 实验介绍

# 1.1.1 关于本实验

ChatGLM2-6B 是开源中英双语对话模型 ChatGLM-6B 的第二代版本，在保留了初代模型对话流畅、部署门槛较低等众多优秀特性的基础之上，ChatGLM2-6B 引入了新特征：更强大的性能、 更长的上下文、更高效的推理、更开放的协议。

本实验利用 mindformers 全流程开发套件基于 ChatGLM2-6B 模型来实现大模型的知识增强检索。

# 1.1.2 实验目的

* 掌握如何利用 mindformers 套件来实现常见大模型的推理。
* 掌握知识增强检索推理的应用。

# 1.1.3 实验环境

华为云 ModelArts-notebook，镜像：贵阳一 mindspore\_2.2.0-cann\_7.0.1-py\_3.9-euler\_2.10.7- aarch64-snt9b

储存配置：200G

Mindformers 版本：1.10

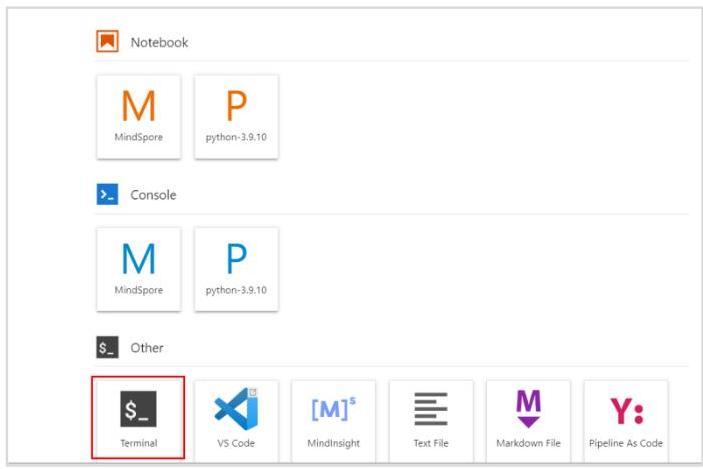
# 1.2 实验任务

# 1.2.1 实验步骤

步骤 1 安装 mindformers

本实验基于 mindformers 1.0 版本完成，首先需要安装 mindformers，如环境中已安装，可跳过本步骤。

打开 Terminal。



# 克隆仓库并安装，代码如下:

git clone -b r1.0 https://gitee.com/mindspore/mindformers.git

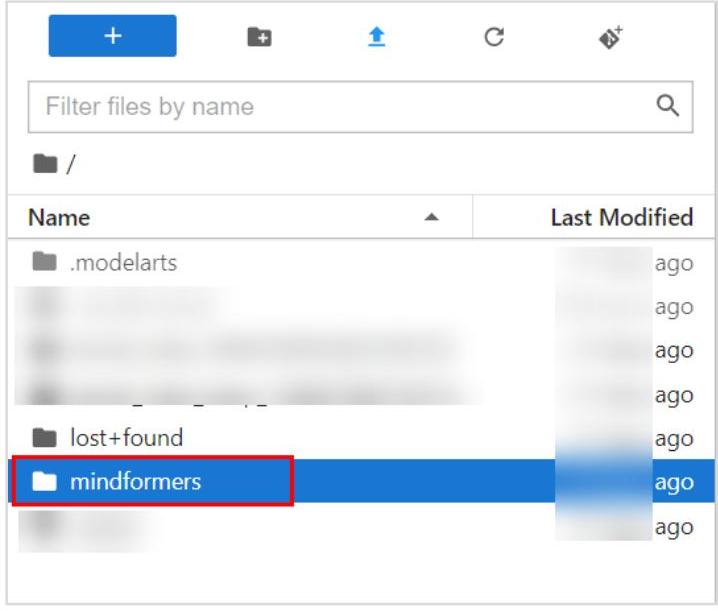
cd mindformers

bash build.sh

输出：

Successfully installed mindformers-1.0.2 mindpet-1.0.3 opencv-python-headless-4.8.1.78 pyarrow-12.0.1 tokenizers-WARNING: You are using pip version 21.0.1; however, version 24.0 is available. You should consider upgrading via the ’/home/ma-user/anaconda3/envs/MindSpore/bin/python3.9 -m pip install – upgrade pip’ command.

此时在 notebook 页面左侧可以看到克隆的 mindformers 仓。



双击进入该文件夹，新建一个 ipynb 文件（可以重命名为 rag-glm2），后续步骤请在该文件中运行。

| image |  | 国 | 金 | C |  |
| --- | --- | --- | --- | --- | --- |
|  | Notebook | è |  |  | Q |
|  | Terminal |  |  |  |  |
|  | Launcher |  |  | - | Last Modified |
|  |  |  |  |  | ago |
|  | .gitee |  |  |  | ago |
|  | .ipynb\_checkpoints |  |  |  | ago |
|  | .jenkins |  |  |  | ago |
|  | build |  |  |  | ago |
|  | chat\_web |  |  |  | ago |
|  | checkpoint\_download |  |  |  | ago |
|  | configs |  |  |  | ago |
|  | II docs |  |  |  | ago |

# 步骤 2 设置 proxy

# 因华为云上 proxy 代理设置，跨区下载受到限制，需要放开下载权限，代码如下：

%env no\_proxy=repo.myhuaweicloud.com, modelarts.cn-north-9.myhuaweicloud.com, obs.cn-north- 9.myhuaweicloud.com, iam.cn-north-9.huaweicloud.com, iam-cache-proxy.cn-north-9.myhuaweicloud.com, dli.cn-north-9.myhuaweicloud.com,127.0.0.1, localhost,172.16.\*

%env NO\_PROXY=repo.myhuaweicloud.com, modelarts.cn-north-9.myhuaweicloud.com, obs.cn-north- 9. myhuaweicloud.com, iam.cn-north-9. huaweicloud.com, iam-cache-proxy.cn-north-9.myhuaweicloud.com, dli.cn-

# 步骤 3 基于大模型推理能力的问答

Mindformers 中可以使用 AutoClass 接口，通过模型名称获取相应的 model/preprocess/tokenizer 等实例，并自动下载并加载权重。

基于 AutoClass 接口来完成推理时，可以采用以下两种方式:

1、直接调用 AutoModel.from\_pretrained()，此时会根据默认配置完成模型实例化；

2、通过 AutoConfig.from\_pretrained()自定义修改配置后再调用 AutoConfig.from\_pretrained()来完成模型实例化。

首次运行代码时，可能需要十几分钟的时间，会自动下载模型文件和响应权重文件（大于

11G），下载的文件默认保存到./checkpoint\_download/glm2 下。首次运行时请耐心等待，后续推理时无需再次下载。

# 代码如下：

import mindspore

from mindformers import AutoConfig, AutoModel, AutoTokenizer

import warnings

warnings.filterwarnings("ignore")

# 指定图模式，指定使用训练卡 id

mindspore.set\_context(mode=0, device\_id=0)

tokenizer = AutoTokenizer.from\_pretrained(’glm2\_6b’)

# model 的实例化有以下两种方式，选择其中一种进行实例化即可

# 1. 直接根据默认配置实例化

# model = AutoModel.from\_pretrained(’glm2\_6b’)

# 2. 自定义修改配置后实例化

config = AutoConfig.from\_pretrained(’glm2\_6b’)

config.use\_past = True # 此处修改默认配置，开启增量推理能够加速推理性能

config.pre\_seq\_len None # 根据需求自定义修改其余模型配置

model AutoModel.from\_config(config) # 从自定义配置项中实例化模型

inputs = tokenizer("你好")["input\_ids"]

# 标准输出流

from mindformers import TextStreamer

streamer TextStreamer(tokenizer)

# 首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，多次重复调用以获取准确的推理性能

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3, streamer=streamer) response tokenizer.decode(outputs) print(response)

输出（包含较多 warning 信息，可忽略。）：

输出较长，只截取关键部分，后续同。

202x-xx-xx 20:18:29,101 - mindformers[mindformers/auto\_class.py:292] - INFO - model built successfully! 202x-xx-xx 20:18:29,916 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: False, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 20:18:29,919 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

[WARNING] GE\_ADPT(152244, ffffbdbcd0b0, python):202x-xx-xx-20:19:09.482.608

[mindspore/ccsrc/transform/graph\_ir/convert.cc:885] ConvertAllNode] Parameter

@kernel\_graph\_1:param\_input\_position cannot found in toposort lists.

[WARNING] DEVICE(152244, ffffbdbcd0b0, python):202x-xx-xx-20:19:10.230.304

[mindspore/ccsrc/plugin/device/ascend/hal/hardware/ge\_graph\_executor.cc:1258] RunInitGraph] Can not find

init\_subgraph.kernel\_graph\_1 sub graph, don’t need data init subgraph in INFER mode.

202x-xx-xx 20:19:41,052 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time:

71.13105392456055 s; generated tokens: 20 tokens; generate speed: 0.281171146728845 tokens/s

[’你好，作为一名人工智能助手，很高兴为您提供帮助。如果您有什么问题需要回答，请您随时告诉我。’]

由于首次调用 model.generate()进行推理将包含图编译时间，推理性能显示不准确，可以多次

重复调用以获取准确的推理性能。

调整输入的问题，查看模型推理结果。

代码如下: print(response)

# 采用真实问题

inputs = tokenizer("华为 nova8 用的什么芯片? ")["input\_ids"]

outputs = model.generate(inputs, max\_new\_tokens=20, do\_sample=True, top\_k=3, streamer=streamer)

response tokenizer.decode(outputs)

# 输出:

202x-xx-xx 11:33:05,587 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 11:33:05,589 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

华为 nova8 用的什么芯片? 根据资料，华为 nova8 使用的芯片是联发科天玑 800 芯片。

202x-xx-xx 11:33:06,535 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time: 0.9448192119598389 s; generated tokens: 16 tokens; generate speed: 16.934456663736963 tokens/s [’华为 nova8 用的什么芯片? 根据资料，华为 nova8 使用的芯片是联发科天玑 800 芯片。’]

输出中包含了问答结果，“华为 Nova8 使用的芯片是联发科天玑 800”，但真实情况是华为 Nova8 使用的芯片是麒麟 985。可见对于新的事实性数据，大模型生成的效果不太理想。

步骤 4 构建知识检索增强的数据集样例

在本实例中假设我们有一个字典形式的知识库，内含各版本华为 nova 手机的芯片型号，希望模型可以结合知识库回答问题。知识库内容代码如下：

# 代码如下:

knowledge\_base = {

"华为 nova1 的芯片是?": "骁龙 625",

"华为 nova2 的芯片是?": "麒麟 659",

"华为 nova2s 的芯片是?": "麒麟 960",

"华为 nova3 的芯片是?": "麒麟 970",

"华为 nova6 的芯片是?": "麒麟 990",

"华为 nova7 的芯片是?": "麒麟 985",

"华为 nova8 的芯片是?": "麒麟 985",

"华为 nova9 的芯片是?": "骁龙 778",

"华为 nova10 的芯片是?": "骁龙 778" }

# 步骤 5 定义知识库查询函数

这个函数接受用户的输入文本和知识库，使用 TF-IDF 向量化并计算输入与知识库问题的余弦相似度，以找出最相似的问题，并返回对应的答案。

代码如下:

# # 定义知识查询函数

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

import numpy as np

import jieba

# 分词函数

def jieba\_tokenize(text):

return jieba.lcut(text)

# 初始化向量化器并向量化知识库问题

def initialize\_knowledge\_base(knowledge\_base):

kb\_questions = list(knowledge\_base.keys())

vectorizer = TfidfVectorizer(tokenizer=jieba\_tokenize, stop\_words=None) # 使用 jieba 分词

\_vectors vectorizer.fit\_transform

return vectorizer, \_vectors, \_questions

def query\_knowledge\_base(input\_text, vectorizer, kb\_vectors, kb\_questions, knowledge\_base):

# 使用相同的向量化器转换输入文本

input\_vector = vectorizer.transform([input\_text])

# 计算输入文本和知识库中所有问题的余弦相似度

cosine\_similarities = cosine\_similarity(input\_vector, kb\_vectors).flatten()

# 找到最相似的问题

most\_similar\_index = np.argmax(cosine\_similarities)

most\_similar\_question \_questions[most\_similar\_index]

# 返回最相似问题的答案

return knowledge\_base.get(most\_similar\_question, "No relevant information found.")

# 步骤 6 定义增强输入文本函数

这个函数使用 query\_knowledge\_base 函数获取输入文本相关的知识库信息，并将这些信息添加到输入文本中，以增强输入，从而提供给模型更丰富的上下文信息。

代码如下：

# 增强输入文本函数

def enhance\_input\_with\_knowledge(input\_text, knowledge\_base):

# 初始化知识库

vectorizer, kb\_vectors, kb\_questions = initialize\_knowledge\_base(knowledge\_base)

knowledge = query\_knowledge\_base(input\_text, vectorizer, kb\_vectors, kb\_questions, knowledge\_base)

if knowledge != "知识库无相关信息.":

# 加入指令性语言，引导模型进行提炼和整合

enhanced\_input = f" {input\_text } 根据信息: ’ {knowledge }’, 总结并回答。"

else:

enhanced\_input input\_text

return enhanced\_input

# 步骤 7 基于知识检索增强的问答

代码如下:

\_inputs = "华为 nova8 用的什么芯片?"

enhanced\_input = enhance\_input\_with\_knowledge(\_inputs, knowledge\_base)

print(’enhanced\_input’, enhanced\_input)

input\_length = len(tokenizer(enhanced\_input)[’input\_ids’])

if input\_length < 0:

raise ValueError("输入问题为空")

inputs = tokenizer(enhanced\_input)[’input\_ids’]

outputs model.generate(inputs, max\_new\_tokens , do\_sample=True, top\_k=3, streamer=streamer)

response tokenizer.decode(outputs)

print(response)

输出:

enhanced\_input 华为 nova8 用的什么芯片? 根据信息: ’麒麟 985’, 总结并回答。

202x-xx-xx 11:43:03,765 - mindformers[mindformers/generation/text\_generator.py:1103] - INFO - Generation Config is: {’max\_length’: 193, ’max\_new\_tokens’: 20, ’num\_beams’: 1, ’do\_sample’: True, ’use\_past’: True, ’temperature’: 1.0, ’top\_k’: 3, ’top\_p’: 1, ’repetition\_penalty’: 1.0, ’encoder\_repetition\_penalty’: 1.0, ’renormalize\_logits’: False, ’pad\_token\_id’: 0, ’bos\_token\_id’: None, ’eos\_token\_id’: 2, ’\_from\_model\_config’: True }

202x-xx-xx 11:43:03,768 - mindformers[mindformers/generation/text\_generator.py:174] - INFO - The generation mode will be \*\*SAMPLE\*\*.

华为 nova8 用的什么芯片? 根据信息: ’麒麟 985’, 总结并回答。 根据提供的信息，华为 nova8 使用的芯片是麒麟 985 。

202x-xx-xx 11:43:04,682 - mindformers[mindformers/generation/text\_generator.py:724] - INFO - total time:

0.9138157367706299 s; generated tokens: 17 tokens; generate speed: 18.60331280798138 tokens/s

["华为 nova8 用的什么芯片？ 根据信息: "麒麟 985"，总结并回答。 根据提供的信息，华为 nova8 使用的芯片是麒麟 985。"]

可见通过知识检索模块，在模型的提示词中加入了麒麟 985 这个信息，模型即可据此信息回答问题。实现基于知识库的问答功能。并且回答的答案是准确的。

# 1.3 实验小结

本实验通过外挂知识库的方式使得大语言模型查询知识内容并基于查询结果给出更准确的推理回答。

# 1.4 思考题

问题：根据输入文本如何在知识库中查找相似的问题答案。

答案：使用余弦相似度计算输入与知识库问题的相似性。