2025

金砖国家职业技能大赛（金砖国家未来技能和技术挑战赛）

人工智能技术应用

BRICS-FS-56

样题(国际总决赛)

2025 年 05 月

目录

1 赛项简介

1.1 赛项名称

2025 金砖国家职业技能大赛（金砖国家未来技能挑战赛）人工智能技术应用赛项，赛项编号 BRICS-FS-56。

1.2 参赛形式

单人赛。

1.3 赛项描述

2025 年金砖国家职业技能大赛人工智能技术应用赛项围绕人工智能基础理论、技术应用和实操能力展开，旨在考察学生的数据处理、算法设计、编程实现、模型训练等能力。本赛项主要面向新一代信息技术相关专业开展，考察模块具体包括图像处理技术、机器学习算法应用、深度学习技术应用、自然语言处理应用开发四个模块。实现培养国际化、高技能、未来技术技能型人才的目标。竞赛由专业的人工智能技能竞赛平台提供竞赛环境和考核系统，选手通过线下方式完成任务考核。本赛项国际总决赛为线上或线下单人赛。

2 参赛对象及竞赛内容

本赛项的参赛对象为高职院校全日制在籍学生以及技师学院在籍学生，采用 Python 语言，基于开源的 OpenCV、TensorFlow、PyTorch 等国内外主流 AI 框架，使用机器学习经典算法、开源计算机视觉算法、卷积神经网络 CNN、循环神经网络 RNN、长短时记忆网络 LSTM 等技术，完成涉及 OpenCV 图像处理、机器学习、

深度学习、自然语言处理等相关模块的算法应用及应用开发，考核内容如下：

模块 A：图像处理技术

使用 OpenCV 开源计算机视觉库，完成图像的基础操作、图像处理和图像特征提取与分析。包括但不限于读取与显示、图像格式转换、图像基本属性获取、图像保存、图像滤波、颜色空间转换、边缘检测等。

模块 B：机器学习算法应用

主要考察机器学习经典算法的应用、数据预处理、特征工程、模型选择与优化、模型评估等知识内容，包括但不限于数据理解与预处理、特征工程、模型选择与优化、模型评估与验证等，

模块 C：深度学习技术应用

主要考察基于 TensorFlow、Pytorch 等深度学习框架，使用深度学习相关技术，完成在图像分类、目标检测、语义分割领域的模型开发，包括但不限于数据集调用、数据预处理、深度网络搭建、模型训练、模型测试、模型应用等。

模块 D：自然语言处理应用开发

以实际问题展开，考察文本分类、情感分析、机器翻译、问答系统、文本生成等。包括但不限于文本清洗、分词与标注、词袋模型特征提取、模型选择与优化、评估与验证等。

3 竞赛内容和时间要求

3.1 竞赛模块和时间要求

人工智能技术应用赛项共 4 个模块，包括模块 A：图像处理技术。模块 B：机器学习算法应用。模块 C：深度学习技术应用。模块 D：自然语言处理应用开发。要求选手在 6 个小时内完成答题，各个模块的答题时间由参赛选手自行分配。

3.2 任务内容

模块 A：图像处理技术

是以 OpenCV 开源计算机视觉库为主，完成图像基本操作，复现经典图像处理算法，如边缘检测、直方图均衡化、滤波去噪等，并验证其在不同类型图像数据上的鲁棒性。本模块包括以下 2 个项目。

项目 1：图形检测

案例说明：

随着计算机视觉技术的发展，图像中目标物体的识别与测量已广泛应用于工业检测、智慧计量、自动化控制等多个领域。传统测量方法人工参与度高、效率低、易出错，而图像处理技术可实现非接触式、自动化、批量化的测量需求。

硬币作为典型的规则圆形物体，是图像识别中常用于教学与实验的对象。通过对图像中硬币的自动识别与位置分析，可进一步实现物体识别、计数、定位与距离估算等功能。

```
import cv2
import numpy as np
第 1 题：使用 opencv 读取图片,赋值给 img（1 分）
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
#复制原图，并使用中值滤波进行降噪
o = img.copy()
o = cv2.medianBlur(o, 5)
```

第 2 题：将图像 o 从彩色图像变成单通道灰度图像（1 分）

上传此部分代码

***

上传此部分代码

第 3 题：使用 opencv 的霍夫变换检测图像中的圆环（2 分）

上传此部分代码

***

上传此部分代码

第 4 题：将检测得到的数据四舍五入成整数（1 分）

上传此部分代码

***

上传此部分代码

#打印出得到的检测结果（圆心横坐标、纵坐标和圆半径）

print(circles[0])

第 5 题：在原图 img 中绘制所有检测到的硬币，绘制颜色为红色（2 分）

上传此部分代码

***

上传此部分代码

第 6 题：将右上角的硬币，绘制颜色为绿色（2 分）

上传此部分代码

***

上传此部分代码

第 7 题：计算所有硬币圆心之间的距离，并打印出最大距离（2 分）

上传此部分代码

***

上传此部分代码

第 8 题：展示上述操作后的图像（1 分）

上传此部分代码

***

上传此部分代码

项目 2：站台数量统计

案例说明

在铁路运输、交通调度和城市基础设施管理中，准确识别和统计站台数量对于运力规划、资源调度和安全监管具有重要价值。传统方式依赖人工巡视或手动标注，存在效率低、易误判的问题。

本项目通过 OpenCV 中的模板匹配算法（Template Matching），自动识别图像或视频帧中出现的站台（或站牌、标志物）图案，实现非人工干预的精准计数，为智能交通系统和图像识别技术的融合应用提供方案支持。

```
import cv2
image = cv2.imread("image.png")  # 读取原始图像
templ = cv2.imread("templ1.png")  # 读取模板图像
```

第 1 题：取模板图像的高度、宽度和通道数（1）

上传此部分代码

***

上传此部分代码

第 2 题:使用 OpenCV 的 matchTemplate 函数,在图像中查找模板图像的位置,匹配的方法：使用归一化相关系数匹配法（2）

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
station_Num = 0  # 初始化快轨的站台个数为 0
for y in range(len(results)):  # 遍历结果数组的行
    for x in range(len(results[y])):  # 遍历结果数组的列
        if results[y][x] > 0.99:
            # 第 3 题：在匹配成功的位置绘制蓝色矩形边框（2）
            ###### 上传此部分代码 ######
            ***
            ###### 上传此部分代码 ######

#           第 4 题:统计匹配成功的站台数量（2）
            ###### 上传此部分代码 ######
            ***
            ###### 上传此部分代码 ######
第 5 题:在图像左上角位置写上红色文字，显示“the numbers of stations:[站点数量(具体数量)]”（1）
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
cv2.imshow("result", image)
cv2.waitKey()
cv2.destroyAllWindows()
```

模块 B：机器学习算法应用

机器学习算法应用，选手需运用 Python 及主流机器学习库（如 Scikit-learn、TensorFlow Lite、Pandas 等）完成从数据预处理、特征工程到模型训练评估的全流程开发。本模块包括以下两个项目。

项目 3：PCA 数据降维

案例说明

随着人工智能的发展，数据维度越来越高。在特征维度较高的数据（如图像、文本、医学数据）中直接进行建模可能面临维度灾难、计算资源浪费、模型过拟合等问题。主成分分析（PCA） 是最常用的线性降维算法，能有效提取数据中最有信息量的部分，实现压缩、可视化与特征优化。

本项目以两类典型数据集为研究对象：鸢尾花数据集和人脸图像数据集

```
import matplotlib.pyplot as plt
# 第 1 题:从 scikit-learn 中导入 load_iris 函数，用于加载鸢尾花数据集（1）
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
from sklearn.decomposition import PCA
iris = load_iris()
y = iris.target
X = iris.data
X.shape
import pandas as pd
pd.DataFrame(X)
```

```
pca = PCA(n_components=2)
pca = pca.fit(X)
X_dr = pca.transform(X)
X_dr.shape
colors = ['red', 'black', 'orange']
iris.target_names
plt.figure()
for i in [0, 1, 2]:
#第 2 题：用 Matplotlib 的 scatter 函数绘制一个散点图（2）
    ###### 上传此部分代码 ######
    ***
    ###### 上传此部分代码 ######
    ,X_dr[y == i, 1]
    ,alpha=0.7#透明度
    ,c=colors[i]
    ,label=iris.target_names[i])
# 第 3 题：显示图例，标明不同类别。（1）
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
plt.title('PCA of IRIS dataset')
plt.show()
```

探索降维后的数据

```
pca.explained_variance_
```

```
#查看降维后每个新特征向量所占的信息量占原始数据总信息量的百分比
pca.explained_variance_ratio_


pca.explained_variance_ratio_.sum()
X.var(axis=0)
X.var(axis=0).sum()
#累积可解释方差贡献率曲线
pca_line = PCA().fit(X)
pca_line.explained_variance_ratio_
pca_line.explained_variance_
pca_line.explained_variance_.sum()
4.22824171/pca_line.explained_variance_.sum()
import numpy as np
pca_line = PCA().fit(X)
# 第 4 题:使用绘制一个折线图。横坐标是 [1,2,3,4]，纵坐标是
np.cumsum(pca_line.explained_variance_ratio_)（2）
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######


#第 5 题：将 x 轴的刻度设置为 1,2,3,4（1）
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
```

```
plt.xlabel("number of components after dimension reduction")
plt.ylabel("cumulative explained variance ratio")
plt.show()
```

最大似然估计自选超参数

```
pca_mle = PCA(n_components="mle")
```

第 6 题:使用 pca_mle 模型实例拟合数据 X（2）

上传此部分代码

***

上传此部分代码

```
X_mle = pca_mle.transform(X)
X_mle.shape
#可以发现，mle 为我们自动选择了 3 个特征
pca_mle.explained_variance_ratio_
pca_mle.explained_variance_ratio_.sum()
```

按信息量占比选超参数

```
pca_f = PCA(n_components=0.97
          ,svd_solver="full"
         )
pca_f = pca_f.fit(X)
X_f = pca_f.transform(X)
pca_f.explained_variance_ratio_
```

人脸识别中属性 components_的运用

第 7 题:从 scikit-learn 中导入 fetch_lfw_people 函数，用于加载 LFW(人脸图像数据集)（1）

```
###### 上传此部分代码 ######

***

###### 上传此部分代码 ######

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

import numpy as np

faces = fetch_lfw_people(min_faces_per_person=60)

faces.images.shape

# 1348 是图像的个数

X = faces.data

fig, axes = plt.subplots(3,8

                         ,figsize=(8,4)

                         ,subplot_kw = {"xticks":[],"yticks":[]} #
不要显示坐标轴                 )

# 第 8 题:把人脸图像数据集中的前若干张图片，显示在之前用
plt.subplots() 创建的子图中，图片显示为灰度图。（2）

###### 上传此部分代码 ######

***

###### 上传此部分代码 ######

#原本有 2900 维，我们现在来降到 150 维

pca = PCA(150).fit(X)

V = pca.components_

V.shape

fig, axes = plt.subplots(3,8,figsize=(8,4),subplot_kw =
```

```
{"xticks":[],"yticks":[]})
```

第 9 题:将 PCA 降维后得到的主成分向量（特征脸）重新还原成图像的形状，并绘制在子图网格中，以灰度图的形式展示（2）

上传此部分代码

***

上传此部分代码

```
faces = fetch_lfw_people(min_faces_per_person=60)
faces.images.shape
faces.data.shape
X = faces.data
pca = PCA(150)
X_dr = pca.fit_transform(X)
X_dr.shape
X_inverse = pca.inverse_transform(X_dr)
X_inverse.shape
fig, ax = plt.subplots(2,10,figsize=(10,2.5)
                       ,subplot_kw={"xticks":[],"yticks":[]}
                       )
for i in range(10):
    ax[0,i].imshow(faces.images[i,:,:],cmap="binary_r")
    ax[1,i].imshow(X_inverse[i].reshape(62,47),cmap="binary_r")
```

项目 4：数据预处理及数据的特征工程

案例说明

在现代机器学习项目中，数据预处理与特征工程已成为影响模型效果的核心要素。无论是结构化表格数据、时间序列数据，还是用户行为数据，原始数据通常存在缺失值、异常值、噪声、非结构化字段等问题，直接影响模型训练效果。本案例聚焦于数据建模前的“前处理”阶段，旨在提高选手对数据的理解能力、处理能力和特征构造能力，为后续的预测建模打下坚实基础。

```
import pandas as pd
import numpy as np

# 防止部分警告
import warnings
warnings.filterwarnings("ignore")

# 数据可视化
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 数据的标签处理
from sklearn.preprocessing import LabelEncoder

#卡方检验
from sklearn.feature_selection import chi2
```

```
from sklearn.feature_selection import SelectKBest
```

第 1 题：从名为 'first_round_training_data.csv' 的 CSV 文件中读取数据（1 分）

```
###### 上传此部分代码 ######
```

***

```
###### 上传此部分代码 ######
```

1、数据探索（数据 EDA）

第 2 题：显示数据的基本信息，比如行数、列数、每列的类型、非空值数量（1 分）

```
###### 上传此部分代码 ######
```

***

```
###### 上传此部分代码 ######
train_data.head()
```

数据离散性和连续性分析

第 3 题：获取除了 'Quality_label' 列之外的所有列名，存入变量 col_name（2 分）

```
###### 上传此部分代码 ######
```

***

```
###### 上传此部分代码 ######
Notdlts_count = []
for i in col_name:
    # 计算非重复值的个数
    Notdlts = len(train_data[i].drop_duplicates())/6000
    Notdlts_count.append(Notdlts)
```

```
plt.plot(col_name, Notdlts_count, c='r')
plt.title('非重复值的总数计算')                    # 标题
plt.xlabel('列名')                              # x 轴 的轴名
plt.ylabel('非重复数据在全数据上的占比')  # y 轴 的轴名
plt.xticks(rotation=45)                         # 旋转 x 轴的刻度名
plt.show()
# 提取出全部的特征
unit = train_data.drop([ 'Quality_label'], 1)
```

数据的分布差异

```
# 遍历列名
for i in col_name:
    plt.hist(unit[i], bins=20)
    plt.title('%s 平均分割取值范围计数统计图'%i)
    plt.xlabel('%s 范围'%i)
    plt.ylabel('值在该范围的个数')
    plt.show()
```

数据的离散程度--看数据的标准差

第 4 题：获取数据框 unit 的所有列名，存入 col_name（2）

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
```

第 5 题：计算每列的标准差（std），并转置后取出对应值，存入 col_std（1）

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
plt.plot(col_name, col_std, c='red')  # 作图
plt.title('列 - 标准差')     # 标题
plt.xlabel('列名')           # x 轴 的轴名
plt.ylabel('标准差')         # y 轴 的轴名
plt.xticks(rotation=90)      # 旋转 x 轴的刻度名
plt.show()
```

数据的标签处理

```
lb = LabelEncoder()


train_data["Quality_label"] =
lb.fit_transform(train_data["Quality_label"])
```

第 6 题：对 unit 数据框中 col_name 对应的所有列的数值，进行四次方根运算（即每个值的 1/4 次方）（1）

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
# 遍历列名
for i in col_name:
    plt.hist(unit[i], bins=20)
    plt.title('%s 平均分割取值范围计数统计图'%i)
    plt.xlabel('%s 范围'%i)
```

```
    plt.ylabel('值在该范围的个数')
    plt.show()
```

去除数据的标准差

```
plt.plot(col_name, col_std**(1/4), c='g')
plt.plot(col_name, 10*np.ones((1,20))[0], c='m', linestyle="--")
plt.title('列 - 标准差')
plt.xlabel('列名')
plt.ylabel('标准差')
plt.xticks(rotation=90)
plt.legend(['标准差','等高线：10'])
plt.show()
```

第 7 题:对 unit 数据框中 col_name 所对应的列的每个数值,进行对数变换(即计算 log(值+1))，把变换后的结果存回原位置。（2）

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
# 计算变换后的 标准差(std)
col_log_std = unit[col_name].describe().T['std']

plt.plot(col_name, col_log_std, c='red')
plt.title('列 - 标准差')
plt.xlabel('列名')
plt.ylabel('标准差')
plt.xticks(rotation=90)
```

```
plt.show()
```

第 8 题：对 unit 数据框中的每一列 i，进行归一化处理，将每个数值转换到 0 到 1 之间。（2）

```
###### 上传此部分代码 ######
```

***

```
###### 上传此部分代码 ######
```

特征选择

第 9 题：使用 SelectKBest 方法，基于卡方检验（chi2），选择前 14 个最重要的特征。（2）

```
###### 上传此部分代码 ######
```

***

```
###### 上传此部分代码 ######
```

第 10 题：用 fit 方法在 unit 特征和对应的目标标签 train_data['Quality_label'] 上进行特征筛选。（2）

```
###### 上传此部分代码 ######
```

***

```
###### 上传此部分代码 ######
# 打印卡方检验值
print(fit.scores_)
train = pd.DataFrame(fit.transform(unit),columns=['V{0}'.format(i) for i in range(1, 15)])
train.head()
```

模块 C：深度学习技术应用

基于卷积神经网络、循环神经网络等，以 TensorFlow、Pytorch 等为框架，

搭建深度神经网络，并训练得到深度模型，完成图像识别、图像分类、语义分割等应用开发。包括以下 1 个项目。

项目 5：猫狗图像分类

案例说明

图像分类是计算机视觉领域的基础任务之一，广泛应用于安防监控、医疗诊断、自动驾驶等场景。猫狗识别作为图像分类的经典入门案例，能够帮助学习者理解图像数据的预处理流程、卷积神经网络的基本结构，以及模型训练与评估的完整流程。 本项目旨在通过深度学习方法，训练一个能够自动识别猫和狗的图像分类模型。

```
import tensorflow as tf
import os
data_dir = './datasets'
train_cats_dir = data_dir + '/train/cats/'
train_dogs_dir = data_dir + '/train/dogs/'
test_cats_dir = data_dir + '/valid/cats/'
test_dogs_dir = data_dir + '/valid/dogs/'
```

第 1 题：统计目录 train_cats_dir 中的文件（或子文件夹）个数（2）

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
# 构建训练数据集
train_cat_filenames = tf.constant([train_cats_dir + filename for
```

```
filename in os.listdir(train_cats_dir)])
train_dog_filenames = tf.constant([train_dogs_dir + filename for
filename in os.listdir(train_dogs_dir)])
train_filenames = tf.concat([train_cat_filenames, train_dog_filenames],
axis=-1)

# cat 0  dog :1
train_labels = tf.concat([
    tf.zeros(train_cat_filenames.shape, dtype=tf.int32),
    tf.ones(train_dog_filenames.shape, dtype=tf.int32)],
    axis=-1)

train_filenames
train_labels
def _decode_and_resize(filename, label):
    image_string = tf.io.read_file(filename)            # 读取原始文件
    image_decoded = tf.image.decode_jpeg(image_string)  # 解码 JPEG 图
片
#第 2 题：将图片调整到 256×256 像素，并归一化（2）
    ###### 上传此部分代码 ######
    ***
    ###### 上传此部分代码 ######
    return image_resized, label
img,label =
```

```
_decode_and_resize(tf.constant('./datasets/train/cats/cat.0.jpg'),tf.
constant(0))
import matplotlib.pyplot as plt
plt.imshow(img.numpy())
#构建训练集


def _decode_and_resize(filename, label):
    image_string = tf.io.read_file(filename)            # 读取原始文件
    image_decoded = tf.image.decode_jpeg(image_string)  # 解码 JPEG 图
片
    image_resized = tf.image.resize(image_decoded, [256, 256]) / 255.0
    return image_resized, label



batch_size = 32
train_dataset = tf.data.Dataset.from_tensor_slices((train_filenames,
train_labels))
train_dataset = train_dataset.map(
    map_func=_decode_and_resize,
    num_parallel_calls=tf.data.experimental.AUTOTUNE)
# 取出前 buffer_size 个数据放入 buffer，并从其中随机采样，采样后的数据
用后续数据替换
train_dataset = train_dataset.shuffle(buffer_size=23000)
```

第 3 题：将训练集重复 3 次。（2）

上传此部分代码

***

上传此部分代码

第 4 题：将训练集划分成固定大小的批次。（2）

上传此部分代码

***

上传此部分代码

第 5 题：提前加载下一个批次的数据，优化训练性能。（2）

上传此部分代码

***

上传此部分代码

```
# 构建测试数据集
test_cat_filenames = tf.constant([test_cats_dir + filename for filename
in os.listdir(test_cats_dir)])
test_dog_filenames = tf.constant([test_dogs_dir + filename for filename
in os.listdir(test_dogs_dir)])
test_filenames = tf.concat([test_cat_filenames, test_dog_filenames],
axis=-1)
test_labels = tf.concat([
    tf.zeros(test_cat_filenames.shape, dtype=tf.int32),
    tf.ones(test_dog_filenames.shape, dtype=tf.int32)],
    axis=-1)
```

```
test_dataset = tf.data.Dataset.from_tensor_slices((test_filenames,
test_labels))
test_dataset = test_dataset.map(_decode_and_resize)
test_dataset = test_dataset.batch(batch_size)


class CNNModel(tf.keras.models.Model):
    def __init__(self):
        super(CNNModel, self).__init__()
        self.conv1 = tf.keras.layers.Conv2D(32, 3, activation='relu')
        self.maxpool1 = tf.keras.layers.MaxPooling2D()
        self.conv2 = tf.keras.layers.Conv2D(32, 5, activation='relu')
        self.maxpool2 = tf.keras.layers.MaxPooling2D()
#       第 6 题：展平操作，把多维特征图变成一维向量。（2）
        ###### 上传此部分代码 ######
        ***
        ###### 上传此部分代码 ######


#       第 7 题:定义一个包含 64 个神经元的全连接层,使用 ReLU 激活函数(2)
        ###### 上传此部分代码 ######
        ***
        ###### 上传此部分代码 ######


#       第 8 题：定义一个包含 2 个神经元的全连接层，使用 Softmax 激活函数
（2）
```

```
        ###### 上传此部分代码 ######
        ***
        ###### 上传此部分代码 ######

    def call(self, x):
        x = self.conv1(x)
        x = self.maxpool1(x)
#       第 9 题：经过第二个卷积层 conv2（2）
        ###### 上传此部分代码 ######
        ***
        ###### 上传此部分代码 ######
        x = self.maxpool2(x)
        x = self.flatten(x)
        x = self.d1(x)
        x = self.d2(x)
        return x

learning_rate = 0.001
model = CNNModel()

loss_object = tf.keras.losses.SparseCategoricalCrossentropy()
#label 没有 one-hot

# 第 10 题：创建一个 Adam 优化器对象（2）
```

```
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######

# 第 11 题：创建了一个用于统计训练损失的指标对象（2）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######

train_accuracy =
tf.keras.metrics.SparseCategoricalAccuracy(name='train_accuracy')

test_loss = tf.keras.metrics.Mean(name='test_loss')
test_accuracy =
tf.keras.metrics.SparseCategoricalAccuracy(name='test_accuracy')


@tf.function
def train_step(images, labels):
    with tf.GradientTape() as tape:
#       第 12 题：通过模型 model(images) 进行预测（2）
        ###### 上传此部分代码 ######
                ***
        ###### 上传此部分代码 ######
```

```
#         第 13 题：计算预测值与真实标签 labels 之间的损失 loss。（2）
          ###### 上传此部分代码 ######
                  ***
          ###### 上传此部分代码 ######
      gradients = tape.gradient(loss, model.trainable_variables)
      optimizer.apply_gradients(zip(gradients,
model.trainable_variables))

      train_loss(loss)
      train_accuracy(labels, predictions)

def test_step(images, labels):
      predictions = model(images)
      t_loss = loss_object(labels, predictions)

      test_loss(t_loss)
      test_accuracy(labels, predictions)


EPOCHS=2
for epoch in range(EPOCHS):
      # 在下一个 epoch 开始时，重置评估指标
      train_loss.reset_states()
```

```
    train_accuracy.reset_states()
    test_loss.reset_states()
    test_accuracy.reset_states()

    for images, labels in train_dataset:
#       第 14 题：调用定义的训练函数，对这批数据进行训练（2）
        ###### 上传此部分代码 ######
                ***
        ###### 上传此部分代码 ######

    for test_images, test_labels in test_dataset:
#       第 15 题:调用函数 test_step(),对当前批次的测试数据进行处理(2)
        ###### 上传此部分代码 ######
                ***
        ###### 上传此部分代码 ######

    template = 'Epoch {}, Loss: {}, Accuracy: {}, Test Loss: {}, Test
Accuracy: {}'
    print(template.format(epoch + 1,
                          train_loss.result(),
                          train_accuracy.result() * 100,
                          test_loss.result(),
                          test_accuracy.result() * 100
                         ))
```

模块 D：自然语言处理应用开发

选手需运用 Python 及主流 NLP 工具库（如 NLTK、spaCy、Transformers 等）完成从数据清洗、模型训练到服务端部署的全流程开发；包括以下项目。

项目 6：LSTM 实现新闻分类

案例说明

在信息爆炸的时代，海量的新闻每天都在被生成与传播。为了帮助用户快速获取感兴趣的内容，新闻平台需对新闻文本进行精准分类。传统的机器学习方法难以捕捉文本的语序和上下文关系，因此引入深度学习中的 LSTM 模型能更好地理解新闻语义，实现更准确的分类。

```
import pandas as pd
import numpy as np
from gensim.models import Word2Vec
import tensorflow as tf
import multiprocessing
import jieba
num_cores = multiprocessing.cpu_count()
print(num_cores)
# 设置 TensorFlow 线程配置
tf.config.threading.set_intra_op_parallelism_threads(16)  # 单个操作内部并
```

行线程数

tf.config.threading.set_inter_op_parallelism_threads(16) # 不同操作间并行线程数

train = pd.read_csv('./cnews/train.tsv',sep='\t',header=None,names=['label','content'])

val = pd.read_csv('./cnews/dev.tsv',sep='\t',header=None,names=['label','content'])

test = pd.read_csv('./cnews/test.tsv',sep='\t',header=None,names=['label','content'])

第 1 题：定义了一个函数 sample_by_label，从数据集中为每个类别随机抽取最多 500 个样本（1）

上传此部分代码

***

上传此部分代码

def val_by_label(df, n=100):

return df.groupby('label').apply(lambda x: x.sample(n=min(n, len(x)), random_state=42)).reset_index(drop=True)

def test_by_label(df, n=100):

```
    return df.groupby('label').apply(lambda x: x.sample(n=min(n, len(x)), random_state=42)).reset_index(drop=True)


# 第 2 题：对 train 数据集调用 sample_by_label 函数，从每个类别中随机抽取最多 500 个样本（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######


val = val_by_label(val)
test = test_by_label(test)
train.shape
val.shape
test.shape
#第 3 题：获取 train 数据集中第一行对应的 content 列的内容（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######
train.shape
def content_cut(x):
```

```
#第 4 题：对输入文本 x 进行中文分词（1）
    ###### 上传此部分代码 ######
        ***
    ###### 上传此部分代码 ######
    x = " ".join(x)
    return x
train['content'] = train['content'].map(lambda x: content_cut(x))
val['content'] = val['content'].map(lambda x: content_cut(x))
test['content'] = test['content'].map(lambda x: content_cut(x))
# 第 5 题：将 train、val 和 test 三个数据集沿着行（row）方向合并，形成一个大的数据集 df（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######
df['content_len'] = df['content'].map(lambda x:len(x.split(" ")))
np.percentile(df['content_len'].values,80)
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(20,10))
# 第 6 题：使用折线图显示 df 数据集中 content_len 列的值。每个点用红色
```

的星号（*）标记（1）

上传此部分代码

***

上传此部分代码

```
plt.axhline(y=np.mean(df['content_len'].tolist()),color="black",)
plt.axhline(y=np.percentile(df['content_len'].values,90),color="peru")
plt.axhline(y=np.percentile(df['content_len'].values,98),color="orange")
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['font.family'] = ['sans-serif']
```

第 7 题：设置 Matplotlib 图表的字体为“黑体（SimHei）”（1）

上传此部分代码

***

上传此部分代码

第 8 题：统计 train 数据集中每个类别（label）的样本数量，并将结果存储在 count_class（1）

上传此部分代码

***

上传此部分代码

```
plt.figure(figsize=(20,8))
```

```
    class_bar = plt.bar(x=count_class.index, height=count_class.tolist(),width=0.4,color='lightcoral')
    plt.xticks(fontsize=20)
    plt.yticks(fontsize=20)
    for bar in class_bar:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2, height+1, str(height), ha="center", va="bottom",fontsize=20)


    plt.ylabel("Sample Count",fontsize=25)
    plt.xlabel("类别名称",fontsize=25)
    import os
    # 第 9 题：定义一个变量 file_name，存放 Word2Vec 模型的路径（1）
    ###### 上传此部分代码 ######
            ***
    ###### 上传此部分代码 ######
    if not os.path.exists(file_name):
        model = Word2Vec([document.split(' ')for document in df['content'].values],
                         vector_size=200,
                         window=5,
```

```
                                    epochs=10,
                                    workers=11,
                                    seed=2018,
                                    min_count=2)

            model.save(file_name)
        else:
            model = Word2Vec.load(file_name)
        print("add word2vec finished....")
        tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=50000,
lower=False,filters="")
        tokenizer.fit_on_texts(df['content'].tolist())

        train_ = tokenizer.texts_to_sequences(train['content'].values)
        val_ = tokenizer.texts_to_sequences(val['content'].values)
        test_ = tokenizer.texts_to_sequences(test['content'].values)
        train_ = tf.keras.preprocessing.sequence.pad_sequences(train_, maxlen=800)
        val_ = tf.keras.preprocessing.sequence.pad_sequences(val_, maxlen=800)
        test_ = tf.keras.preprocessing.sequence.pad_sequences(test_, maxlen=800)
```

```
word_vocab = tokenizer.word_index
count = 0


embedding_matrix = np.zeros((len(word_vocab) + 1, 200))
for word, i in word_vocab.items():
    embedding_vector = model.wv[word] if word in model.wv else None
    if embedding_vector is not None:
        count += 1
        embedding_matrix[i] = embedding_vector
    else:
        unk_vec = np.random.random(200) * 0.5
        unk_vec = unk_vec - unk_vec.mean()
        embedding_matrix[i] = unk_vec
train.head()
#label 编码
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
lb = LabelEncoder()
```

第 10 题：使用标签编码器（lb）将 train 数据集中 label 列的类别标签转换成数字标签，并存储在 train_label 中（1）

```
###### 上传此部分代码 ######

        ***

###### 上传此部分代码 ######

val_label = lb.transform(val['label'].values)

test_label = lb.transform(test['label'].values)

content = tf.keras.layers.Input(shape=(800), dtype='int32')

embedding = tf.keras.layers.Embedding(

    name="word_embedding",

    input_dim=embedding_matrix.shape[0],

    weights=[embedding_matrix],

    output_dim=embedding_matrix.shape[1],

    trainable=False)

x = tf.keras.layers.SpatialDropout1D(0.2)(embedding(content))

#编码层

#bi-GRU

#bi-GRU

x = tf.keras.layers.Bidirectional(tf.keras.layers.GRU(200,
return_sequences=True))(x)  #（batch,800,400）

x = tf.keras.layers.Bidirectional(tf.keras.layers.GRU(200,
return_sequences=True))(x)
```

```
#池化层
avg_pool = tf.keras.layers.GlobalAveragePooling1D()(x) #（batch,400）
# 第 11 题：使用全局最大池化层对输入 x 进行池化操作（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######

conc = tf.keras.layers.concatenate([avg_pool, max_pool])

x = tf.keras.layers.Dense(1000)(conc)
# 第 12 题：对 x 张量进行批归一化（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######
x = tf.keras.layers.Activation(activation="relu")(x)
# 第 13 题：添加 Dropout 层，对 x 进行操作，随机丢弃 20%的神经元（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######
```

```
x = tf.keras.layers.Dense(500)(x)

x = tf.keras.layers.BatchNormalization()(x)

x = tf.keras.layers.Activation(activation="relu")(x)

x = tf.keras.layers.Dense(10)(x)

output = tf.nn.softmax(x)



model = tf.keras.models.Model(inputs=content, outputs=output)

train_label

len(train_[0])

train_label = tf.keras.utils.to_categorical(train_label,num_classes=10,dtype='int')

val_label = tf.keras.utils.to_categorical(val_label,num_classes=10,dtype='int')

test_label = tf.keras.utils.to_categorical(test_label,num_classes=10,dtype='int')

train_label.shape

train_label

# 第 14 题：将训练数据和对应的标签转换成一个 TensorFlow 的数据集对象
train_dataset（2）

###### 上传此部分代码 ######

        ***

###### 上传此部分代码 ######

for a,b in train_dataset.take(1):
```

```
    print(a.shape,b.shape)


import tensorflow as tf
from tqdm import tqdm
# 配置参数
learning_rate = 0.001
BATCH_SIZE = 1024  # 根据内存调整
VAL_BATCH_SIZE = 1024
EPOCHS = 2
def configure_dataset(dataset, shuffle=False, batch_size=BATCH_SIZE):
    if shuffle:
        dataset = dataset.shuffle(buffer_size=1000)
    dataset = dataset.batch(batch_size)
    dataset = dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
    # 设置并行处理选项
    options = tf.data.Options()
    options.threading.private_threadpool_size = 16  # 根据 CPU 核心数调整
    options.threading.max_intra_op_parallelism = 16
    dataset = dataset.with_options(options)
```

```
    return dataset

# 准备数据集

# 第 15 题：调用 configure_dataset 函数，对 train_dataset 进行配置，并启用数据的随机打乱（1）

###### 上传此部分代码 ######

        ***

###### 上传此部分代码 ######

val_dataset = configure_dataset(tf.data.Dataset.from_tensor_slices((val_, val_label)),
                                batch_size=VAL_BATCH_SIZE)



# 模型和优化器

# 第 16 题：创建一个分类交叉熵损失函数对象 loss_object（1）

###### 上传此部分代码 ######

        ***

###### 上传此部分代码 ######

optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)



# 指标
```

```
# 第 17 题：创建一个用于记录训练过程中平均损失值的指标对象 train_loss
（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######
train_accuracy = tf.keras.metrics.CategoricalAccuracy(name='train_accuracy')
val_loss = tf.keras.metrics.Mean(name='val_loss')
val_accuracy = tf.keras.metrics.CategoricalAccuracy(name='val_accuracy')


@tf.function
def train_one_step(x, y):
    with tf.GradientTape() as tape:
        predictions = model(x, training=True)
        loss = loss_object(y, predictions)
    gradients = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))
    train_loss.update_state(loss)
    train_accuracy.update_state(y, predictions)


@tf.function
```

```
    def val_one_step(x, y):
        predictions = model(x, training=False)
        v_loss = loss_object(y, predictions)
        val_loss.update_state(v_loss)
    #第 18 题：将真实标签 y 和模型预测的结果 predictions 更新到验证准确率指标 val_accuracy 中（1）
        ###### 上传此部分代码 ######
            ***
        ###### 上传此部分代码 ######


    def train_for_epoch(train_dataset, val_dataset, epoch):
        train_loss.reset_state()
        train_accuracy.reset_state()
        val_loss.reset_state()
        val_accuracy.reset_state()
            # 训练阶段
        for x_batch, y_batch in tqdm(train_dataset, desc=f'Train Epoch {epoch+1}'):
            train_one_step(x_batch, y_batch)
        # 验证阶段
```

```
        for x_batch, y_batch in val_dataset:
            val_one_step(x_batch, y_batch)
        # 打印结果
        print(f"Epoch {epoch+1}, "
              f"Loss: {train_loss.result():.4f}, "
              f"Acc: {train_accuracy.result()*100:.2f}%, "
              f"Val Loss: {val_loss.result():.4f}, "
              f"Val Acc: {val_accuracy.result()*100:.2f}%")
# 训练循环
for epoch in range(EPOCHS):
    train_for_epoch(train_dataset, val_dataset, epoch)
# 第 19 题：打印模型的结构信息（1）
###### 上传此部分代码 ######
        ***
###### 上传此部分代码 ######
model(tf.constant([test_[1]]))
test_label[0]
test_dataset = tf.data.Dataset.from_tensor_slices(test_)
test_dataset = test_dataset.batch(batch_size=256)
predictions=[]
```

```
    for line in test_dataset:
        prediction = model(line)
        predictions.extend(list(np.argmax(prediction.numpy(),axis=1)))
test_.shape
from sklearn.metrics import accuracy_score
test_true = list(np.argmax(test_label,axis=1))
accuracy_score(test_true,predictions)
from sklearn.metrics import classification_report
print(classification_report(test_true,predictions,target_names=list(lb.classes_)))
```