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(省级/区域选拔赛)

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。本赛项省级选拔赛为线下单人赛。

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 4 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：OpenCV 图像基本处理

案例说明：

该项目基于 OpenCV 库，围绕经典 Lena 图像展开一系列基础图像处理操作，涵盖图像的读取与显示、颜色空间转换（如灰度化）、图像滤波（均值、高斯、中值滤波）、边缘检测（Canny 算法）、阈值处理等核心功能，旨在通过具体案例展示 OpenCV 在图像处理中的基本应用方法与算法效果，帮助初学者理解并掌握图像处理的基础知识与技术流程。请在下列***处编程，完成整个案例开发。

第 1 题：导入 OpenCV 库。1 分

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

```
import numpy as np
from matplotlib import pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

第 2 题：使用相对路径 './data/lena.jpg'读取 lena.jpg 图像。1 分

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

if image is None:
    print("无法加载图像，请检查路径是否正确")
    exit()
```

第 3 题：显示原始图像。1 分

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

plt.title('原始图像')
plt.axis('off')
```

第 4 题：将图像转换为灰度图像。1 分

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

plt.imshow(gray_image, cmap='gray')
plt.title('灰度图像')
```

```
plt.axis('off')
```

第 5 题：使用 Canny 算法对图像进行边缘检测，阈值范围设置为 100-200。1 分

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

plt.imshow(edges, cmap='gray')
plt.title('边缘检测')
plt.axis('off')
```

第 6 题：使用高斯模糊对图像进行滤波，高斯核大小设置为 5×5。1 分

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

plt.imshow(cv2.cvtColor(blurred, cv2.COLOR_BGR2RGB))
plt.title('高斯模糊')
plt.axis('off')
```

第 7 题：对图像进行阈值处理，阈值设置为大小 127。2 分

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

```
plt.imshow(thresh, cmap='gray')
plt.title('阈值处理')
plt.axis('off')
```

第 8 题:将处理后的图像分别命名为 gray_image.jpg,edges.jpg,blurred.jpg,thresh.jpg。保存在 data 目录下。2 分

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

项目 2：OpenCV 行人检测

案例说明：

该项目是一个基于 OpenCV 的行人检测系统，通过加载预训练的 HOG（方向梯度直方图）和 SVM（支持向量机）分类器模型，对输入的静态图像进行行人检测。该项目无需自定义函数，直接调用 OpenCV 内置工具实现快速、简单的行人检测功能，适用于快速验证和演示计算机视觉技术在行人检测中的应用。 请在下列***处编程，完成整个案例开发。（共 10 分）

注意：已有代码请勿修改，编写代码时注意将***删掉，将指定的答案按要求上传至竞赛平台的答题框内。

第 1 题：导入 numpy 库并简写为 np。1 分

```
import cv2
###### 上传此部分代码 ######
***
```

上传此部分代码

第 2 题：读取行人图像，图象在 data 文件夹下。1 分

上传此部分代码

***

上传此部分代码

```
if image is None:
    print("无法加载图像，请检查路径是否正确")
    exit()
```

第 3 题：调用 OpenCV 内置函数，初始化 HOG 描述符和行人检测器。2 分

上传此部分代码

***

```
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

上传此部分代码

第 4 题：调整图像大小以提高检测速度，大小为 640×480。1 分

上传此部分代码

***

上传此部分代码

第 5 题：调用 hog 特征检测器，并设置适当参数，检测行人。2 分

上传此部分代码

***

上传此部分代码

第 6 题：调用函数绘制检测到的行人矩形框。1 分

上传此部分代码

***

上传此部分代码

第 7 题：显示结果，并命名图像名字为 Pedestrian Detection。2 分

上传此部分代码

***

上传此部分代码

cv2.waitKey(0)

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：基于 KNN 和 face_recognition 的人脸别

案例说明：

人脸识别，是在图像和视频中检测人脸，并与人脸数据库中进行实时对比，从而实现身份识别。本案例是基于 KNN 算法和 face_recognition 库，完成人脸识别，该库是简单的基于 python 的人脸识别库，是一个基于 C++开源库 dlib 中的人脸识别模块。其核心功能是将图像中的人脸进行编码，然后与已有的人脸编码进行比对，从而进行人脸识别。请在下列***处编程，完成人脸识别案例开发。

（14 分）

注意：已有代码请勿修改，填写代码时注意将***删掉，将指定的答案上传至竞赛平台的答题框内。

第 1 题：从 sklearn 中导入 neighbors 模块。1 分

```
import math

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

import os
import os.path
import pickle
from PIL import Image, ImageDraw
import face_recognition
from face_recognition.face_recognition_cli import
image_files_in_folder
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

第 2 题：创建两个空列表 X 和 y，分别用于存放训练集图片和训练标签。2 分

上传此部分代码

***

上传此部分代码

第 3 题：使用相对路径设置训练图片的路径，赋值给变量 train_dir，图像存放于 data 文件夹下的 train 文件中。2 分

上传此部分代码

***

上传此部分代码

第 4 题：使用 face_recognition 库的相关函数，获取图像中人脸的边界框，赋值给变量 face_bounding_boxes。1 分

```
# 循环获取训练集图片
for class_dir in os.listdir(train_dir):
    for img_path in image_files_in_folder(os.path.join(train_dir, class_dir)):
        # 加载图片文件，其实是 numpy 数组
        image = face_recognition.load_image_file(img_path)

        # 获取人脸检测框
        ###### 上传此部分代码 ######
        ***
        ###### 上传此部分代码 ######
```

```
            # 多个人物或者 0 个人物不处理
            if len(face_bounding_boxes) != 1:
                if verbose:
                    print("Image {} not suitable for training:
{}".format(img_path, "Didn't find a face" if len(face_bounding_boxes) <
1 else "Found more than one face"))
            else:
                # 将编码后的人脸图像添加到训练集中
                X.append(face_recognition.face_encodings(image,
known_face_locations=face_bounding_boxes)[0])
                y.append(class_dir)
```

第 5 题：创建 KNN 分类器，n_neighbors=3，algorithm='ball_tree'，其余参数默认即可。其余参数默认即可。2 分

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

第 6 题：使用训练集训练模型 knn_clf。2 分

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

第 7 题:将模型保存在 data 文件夹下,模型命名为'trained_knn_model.clf'。

2 分

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
with open(model_save_path, 'wb') as f:
    pickle.dump(knn_clf, f)
if knn_clf is None:
    with open(model_save_path, 'rb') as f:
        knn_clf = pickle.load(f)
```

第 8 题：取到人脸的四个顶点坐标，并画出框，在框的下面画出一个小矩形，把预测的名字放在里面，如果是未知的人，名字为'unkonwn'。2 分

```
# 取到人脸的四个顶点坐标，并画出框，在框的下面画出一个小矩形，把预测的名字放在里面，如果是未知的人，名字为'unkonwn'。请在适当位置补齐代码
distance_threshold=0.6
for image_file in os.listdir("data/test"):
    full_file_path = os.path.join("data/test", image_file) # 完整的测试图片路径
        X_img = face_recognition.load_image_file(full_file_path)
        X_face_locations = face_recognition.face_locations(X_img)
        pil_image = Image.open(full_file_path)
        draw = ImageDraw.Draw(pil_image)
```

```
        faces_encodings = face_recognition.face_encodings(X_img,
known_face_locations=X_face_locations)
        closest_distances = knn_clf.kneighbors(faces_encodings,
n_neighbors=1) # 距离计算
        print(closest_distances)
        are_matches = [closest_distances[0][i][0] <= distance_threshold
for i in range(len(X_face_locations))]

        for pred, loc, rec in zip(knn_clf.predict(faces_encodings),
X_face_locations, are_matches):
    ###### 上传此部分代码 ######
            ***
    ###### 上传此部分代码 ######

            draw.rectangle(((left, top), (right, bottom)), outline=(0,
0, 255)) # 画框
            name = pred.encode("UTF-8")
            if not rec:
                name = 'unknown'
            text_width, text_height = draw.textsize(name)
            draw.rectangle(((left, bottom - text_height - 10), (right,
bottom)), fill=(0, 0, 255), outline=(0, 0, 255)) #显示文字的框
            draw.text((left + 6, bottom - text_height - 5), name,
```

```
fill=(255, 255, 255, 255)) # 文字内容，名字
        del draw
        plt.ion()
        plt.imshow(pil_image)
        plt.pause(2)
```

项目 4：鸢尾花图像分类

案例说明：

该项目是一个基于鸢尾花数据集的分类任务，系统考察了机器学习从数据预处理到模型部署的全流程：通过标准化特征、划分训练测试集，实现逻辑回归、随机森林和支持向量机三种模型并进行评估，利用网格搜索优化随机森林超参数，最终输出分类性能指标（如准确率、混淆矩阵）并可视化结果，同时保存优化后的模型以支持后续部署应用，全面覆盖了数据清洗、模型训练、调优、评估及工程化等核心技能点。请在下列***处编程，完成案例开发。 （16 分）

注意：已有代码请勿修改，填写代码时注意将***删掉，将指定的答案上传至竞赛平台的答题框内。

第 1 题：调用 sklearn 中的以下模块，load_iris，SVC，RandomForestClassifier，LogisticRegression。2 分

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
```

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

    from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
    from sklearn.pipeline import Pipeline
    import matplotlib.pyplot as plt
    import seaborn as sns
    import joblib
```

第 2 题：加载鸢尾花数据集。2 分

```
    ###### 上传此部分代码 ######
    ***
    ###### 上传此部分代码 ######
    X = iris.data
    y = iris.target
    feature_names = iris.feature_names
    target_names = iris.target_names
```

第 3 题：数据预处理，标准化特征。2 分

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

第 4 题：划分训练集和测试集，训练集：测试集=7:3。2 分

上传此部分代码

***

上传此部分代码

第 5 题：定义分类模型和参数，模型分别采用支持向量机、随机森林和逻辑回归算法。3 分

上传此部分代码

***

上传此部分代码

第 6 题：训练和评估模型。2 分

上传此部分代码

***

上传此部分代码

```
        print(f"{name} Accuracy: {accuracy:.2f}")
        print(classification_report(y_test, y_pred,
target_names=target_names))
        print("Confusion Matrix:")
        print(confusion_matrix(y_test, y_pred))
        print("\n")
    # 使用网格搜索优化随机森林模型
    param_grid = {
```

```
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

rf_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier())
])

grid_search = GridSearchCV(rf_pipeline, param_grid, cv=5,
scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best parameters for Random Forest:")
print(grid_search.best_params_)
print("Best accuracy for Random Forest:
{:.2f}".format(grid_search.best_score_))
```

第 7 题：使用最佳模型进行预测。2 分

```
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
y_pred_best = best_rf.predict(X_test)
```

```
    print("Optimized Random Forest Accuracy:
{:.2f}".format(accuracy_score(y_test, y_pred_best)))
```

第 8 题：保存模型。1 分

```
    }###### 上传此部分代码 ######
    ***
    }###### 上传此部分代码 ######
    # 可视化混淆矩阵
    cm = confusion_matrix(y_test, y_pred_best)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=target_names, yticklabels=target_names)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix for Optimized Random Forest')
    plt.show()
```

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 2 个项目。

项目 5：CIFAR-10 图像分类

案例说明：

本案例是基于深度学习的图像分类方法，该方法也是目前热门的研究方向。本案例使用 Keras 框架搭建 CNN 实现对 CIFAR-10 数据集的图像分类，属于 10 分类问题。采用 tensorflow 和 keras 内置的数据集,CIFAR-10 数据集包含 60000 张图像，其中训练集包含 50000 张图像，测试集包含 10000 张图像，共有十个类别：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车。图像形状为 32×32×3，即尺寸为 32×32 像素的 3 通道彩色图像。请在下列***处编程，完成整个案例开发。（共 15 分）

注意：已有代码请勿修改，编写代码时注意将***删掉，将指定的答案按要求上传至竞赛平台的答题框内。

第 1 题：导入库 tensorflow 库，并简写为 tf。（1 分）

```
import warnings
warnings.filterwarnings('ignore')
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
```

```
import matplotlib.pyplot as plt
from tensorflow.keras import models, layers
```

第 2 题：从 keras 内置的数据集模块中加载 cifar-10 数据集。（1 分）

变量名命名要求如下：

训练集图像：x_train

训练集标签：y_train

测试集图像：x_test

测试集标签：y_test

上传此部分代码

***

上传此部分代码

第 3 题：将训练集图像和测试集图像归一化到 0~1 之间。（1 分）

上传此部分代码

***

上传此部分代码

第 4 题：请使用正向索引的方法，展示测试集中第 8 张图像的像素。（0.5）

上传此部分代码

***

上传此部分代码

第 5 题：使用 y_test 的 shape 属性，展示测试集标签的形状。注意：将代码及输出结果上传至答题框。（0.5 分）

上传此部分代码及输出结果

***

上传此部分代码及输出结果

第 6 题：使用 matplotlib 库展示训练集的前 9 张图像，画板大小为 5×5，分为 3 行 3 列。（1 分）

```
class_names = ['airplne', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'forg', 'horse', 'ship', 'truck']

###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i], cmap = plt.cm.binary)
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```

第 7 题：使用 keras 的 models 类和 layers 类，按照下列要求搭建卷积神经网络模型。

（1）调用 models 类中的顺序式 API 搭建模型，模型命名为 model。（1 分）

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

（2）使用 model.add()函数向 model 中添加第 1 卷积层。（1 分）

要求：

滤波器个数为 32，

卷积核大小为 3×3，

激活函数为 relu，

输入数据形状为(32,32,3)。

上传此部分代码

***

上传此部分代码

（3）使用 model.add()函数向 model 中添加第 1 最大池化层，池化核大小为 2×2，步长为 2。（1 分）

上传此部分代码

***

上传此部分代码

```
model.add(layers.Conv2D(64, 3, activation = 'relu'))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Conv2D(64, 3, activation = 'relu'))
model.add(layers.MaxPooling2D(2, 2))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation = 'relu'))
```

（4）添加 Dropout 层，使每个神经元以 30%的概率随机失活，抑制模型过拟合。（1 分）

上传此部分代码

***

上传此部分代码

（5）添加分类层，以全连接层作为分类层，本项目是 10 分类，并设置激活函数为 softmax。（1 分）

上传此部分代码

***

上传此部分代码

第 8 题:编译模型,设置模型的优化器为 adam,损失函数为稀疏交叉熵损失,评估指标为 accuracy。（2 分）

########## 上传此部分代码 ##########

***

########## 上传此部分代码 ##########

第 9 题：调用训练函数，对模型进行训练 10 代，要求每训练 1 代，使用测试集对模型进行 1 次评估。（2 分）

########## 上传此部分代码 ##########

***

########## 上传此部分代码 ##########

第 10 题：使用测试集评估模型。注意：将代码及输出结果上传至答题框。（1 分）

上传此部分代码及输出结果

***

上传此部分代码及输出结果

项目 6：图像语义分割

案例说明：

本案例使用 Oxford-IIIT 数据集，完成宠物图像语义分割。该宠物数据集是 37 个类别的宠物图像数据集，其中有犬类 25 类，猫类 12 类，每个类别大约有 200 张图像。图像在比例，姿势和照明方面有很大的差异。该数据集由图像、图像所对应的标签、以及对像素逐一标记的掩码组成。掩码其实就是给每个像素的标签。每个像素分别属于以下三个类别中的一个：

类别 1：像素是宠物的一部分；

类别 2：像素是宠物的轮廓；

类别 3：以上都不是/外围像素。

请在下列***处补全代码，完成案例开发。 （15 分）

注意：已有代码请勿修改，填写代码时注意将***删掉，将指定的答案上传至竞赛平台的答题框内。

第 1 题：导入 matplotlib 库的子库 pyplot，并简写为 plt。1 分

```
import tensorflow as tf
import os
from tensorflow_examples.models.pix2pix import pix2pix
import tensorflow_datasets as tfds
tfds.disable_progress_bar()
```

```
from IPython.display import clear_output


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


import warnings
warnings.filterwarnings('ignore')
```

第 2 题：使用 tensorflow_datasets 的函数加载数据集，数据放在“/data/oxford/”目录下。2 分

```
###### 上传此部分代码 ######
dataset, info = ***
###### 上传此部分代码 ######
def normalize(input_image, input_mask):
    input_image = tf.cast(input_image, tf.float32)/128.0 - 1
    input_mask -= 1
    return input_image, input_mask
def load_image_train(datapoint):
    input_image = tf.image.resize(datapoint['image'], (128, 128))
    input_mask = tf.image.resize(datapoint['segmentation_mask'],
(128, 128))
    if tf.random.uniform(()) > 0.5:
        #按水平（从左向右）随机翻转图像
```

```
            input_image = tf.image.flip_left_right(input_image)
            input_mask = tf.image.flip_left_right(input_mask)
        input_image, input_mask = normalize(input_image, input_mask)
        return input_image, input_mask
    def load_image_test(datapoint):
        input_image = tf.image.resize(datapoint['image'], (128, 128))
        input_mask = tf.image.resize(datapoint['segmentation_mask'],
(128, 128))
        input_image, input_mask = normalize(input_image, input_mask)
        return input_image, input_mask
```

第 3 题：设置超参数，BATCH_SIZE 设置为 64，BUFFER_SIZE 设置为 1000。2 分

```
    TRAIN_LENGTH = info.splits['train'].num_examples
    ###### 上传此部分代码 ######
    BATCH_SIZE = ***
    BUFFER_SIZE = ***
    ###### 上传此部分代码 ######

    STEPS_PER_EPOCH = TRAIN_LENGTH // BATCH_SIZE
    #使用 map 方法将此函数应用于数据集中的每个项
    train = dataset['train'].map(load_image_train,
num_parallel_calls=tf.data.experimental.AUTOTUNE)
    test = dataset['test'].map(load_image_test)
```

```
    #对数据进行打乱分批处理
    train_dataset =
train.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat()
    train_dataset =
train_dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
    test_dataset = test.batch(BATCH_SIZE)
```

第 4 题：定义一个可视化函数，查看数据集一例图像及其所对应的掩码。要求：画布大小为 15×15，图像上方展示相应的标签。包括：'Input Image','True Mask', 'Predicted Mask'。2 分

```
    def display(display_list):
        ###### 上传此部分代码 ######
        ***
        ###### 上传此部分代码 ######
    for image, mask in train.take(1):
        sample_image, sample_mask = image, mask
    display([sample_image, sample_mask])
    OUTPUT_CHANNELS = 3
```

第 5 题：设置网络输出通道数为 3，使用预训练模型 MobileNetV2 作为基础模型，并去掉基础模型的 top 层。3 分

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

```
    layer_names = [
        'block_1_expand_relu',   # 64x64
        'block_3_expand_relu',   # 32x32
        'block_6_expand_relu',   # 16x16
        'block_13_expand_relu',  # 8x8
        'block_16_project',      # 4x4
    ]
    layers = [base_model.get_layer(name).output for name in layer_names]
    # 创建特征提取模型
    down_stack = tf.keras.Model(inputs=base_model.input,
outputs=layers)
    #不允许训练这些层
    down_stack.trainable = False
    up_stack = [
        pix2pix.upsample(512, 3),  # 4x4 -> 8x8
        pix2pix.upsample(256, 3),  # 8x8 -> 16x16
        pix2pix.upsample(128, 3),  # 16x16 -> 32x32
        pix2pix.upsample(64, 3),   # 32x32 -> 64x64
    ]
    def unet_model(output_channels):
        # 这是模型的最后一层
        last = tf.keras.layers.Conv2DTranspose(
            output_channels, 3, strides=2,
            padding='same', activation='softmax')  #64x64 -> 128x128
```

```
    inputs = tf.keras.layers.Input(shape=[128, 128, 3])
    x = inputs
    # 在模型中降频取样
    skips = down_stack(x)
    x = skips[-1]
    skips = reversed(skips[:-1])
    # 升频取样然后建立跳跃连接
    for up, skip in zip(up_stack, skips):
        x = up(x)
        concat = tf.keras.layers.Concatenate()
        x = concat([x, skip])
    x = last(x)
    return tf.keras.Model(inputs=inputs, outputs=x)
```

第 6 题：编译模型，设置优化器为 adam，损失函数为稀疏交叉熵损失函数，评估指标为准确率。1 分

```
model = unet_model(OUTPUT_CHANNELS)
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
def create_mask(pred_mask):
    pred_mask = tf.argmax(pred_mask, axis=-1)
    pred_mask = pred_mask[..., tf.newaxis]
    return pred_mask[0]
```

```
    def show_predictions(dataset=None, num=1):
        if dataset:
            for image, mask in dataset.take(num):
                pred_mask = model.predict(image)
                display([image[0], mask[0], create_mask(pred_mask)])
        else:
            display([sample_image,
sample_mask,create_mask(model.predict(sample_image[tf.newaxis, ...]))
])
    class DisplayCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs=None):
            clear_output(wait=True)
            show_predictions()
            print ('\nSample Prediction after epoch
{}\n'.format(epoch+1))
```

第 7 题：训练模型，训练 2 代，并从测试集中划分出一部分充当验证集，每训练一代验证一次，调用可视化函数展示训练过程，并将训练数据存储在 model_history 中。1 分

```
    EPOCHS = 2
    VAL_SUBSPLITS = 5
    VALIDATION_STEPS =
info.splits['test'].num_examples//BATCH_SIZE//VAL_SUBSPLITS
```

```
###### 上传此部分代码 ######

***

###### 上传此部分代码 ######
```

第 8 题：绘制训练集和测试集的损失函数变化曲线。2 分

```
###### 上传此部分代码 ######

***

###### 上传此部分代码 ######



epochs = range(EPOCHS)


plt.figure()

plt.plot(epochs, loss, 'r', label='Training loss')

plt.plot(epochs, val_loss, 'bo', label='Validation loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss Value')

plt.ylim([0, 1])

plt.legend()

plt.show()
```

第 9 题：使用测试集完成模型预测。1 分

```
###### 上传此部分代码 ######
```

***

上传此部分代码

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 7：酒店评价情感分析

案例说明：

数据来源于一个较大规模的酒店评论语料，规模为 10000 条，是从携程旅行中收集整理而成。本案例选用了其中的 6000 条评论，其中正负评论各 3000 条。情感词典主要包括四类：通用情感词、程度副词、否定词、领域词。请在下列***处编程，完成案例开发。（20 分）

注意：已有代码请勿修改，填写代码时注意将***删掉，将每题的答案上传至竞赛平台的答题框内。

```
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import re
import jieba
from gensim.models import KeyedVectors
import warnings
```

```
warnings.filterwarnings("ignore")

import bz2

import warnings

warnings.filterwarnings('ignore')

cn_model = KeyedVectors.load_word2vec_format('./data/embeddings/sgns.zhihu.bigram',
                                             binary=False,
unicode_errors="ignore")
```

第 1 题：请用 cn_model 展示“人工智能”这个词的词向量长度，并赋值给 embedding_dim。1 分

```
###### 上传此部分代码 ######

***

###### 上传此部分代码 ######


print('词向量长度为{}'.format(embedding_dim))
```

第 2 题：请使用 cn_model 的函数，计算并展示“饮料”和“果汁”的相似度。1 分

```
###### 上传此部分代码 ######

***
```

```
###### 上传此部分代码 ######

cn_model.most_similar(positive='美丽',topn=5)

#读入数据集，分好训练样本和训练标签

train_texts_orig = []

# 文本所对应的 labels，也就是标记

train_target = []

with open("./data/positive_samples.md", "r", encoding="utf-8") as f:

    lines = f.readlines()

    for line in lines:

        dic = eval(line)

        train_texts_orig.append(dic["text"])

        train_target.append(dic["label"])
```

第 3 题：请读取 negative_samples.md 样本，并把内容保存到 train_texts_orig 中，把标签保存到 train_target 中。2 分

```
with open("./data/negative_samples.md", "r", encoding="utf-8") as f:

    lines = f.readlines()

    for line in lines:

        dic = eval(line)

###### 上传此部分代码 ######
```

***

上传此部分代码

导入 tensorflow 的接口，构建模型

from tensorflow.keras.models import Sequential

import tensorflow as tf

from tensorflow.keras.layers import Dense, GRU, Embedding, LSTM, Bidirectional

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.optimizers import RMSprop

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlateau

第 4 题：使用结巴分词对训练集每句话进行分词处理。2 分

进行分词和 tokenize

train_tokens = []

for text in train_texts_orig:

text = re.sub("[\s+\.\!\/_,$%^*(+\"\']+|[+——！，。？、~@#￥%……&*

（）]+", "",text)

结巴分词

上传此部分代码

***

上传此部分代码

cut_list = [i for i in cut]

for i, word in enumerate(cut_list):

try:

cut_list[i] = cn_model.key_to_index[word]

except KeyError:

cut_list[i] = 0

train_tokens.append(cut_list)

第 5 题：请获取所有句子长度，并赋值给 num_tokens。1 分

上传此部分代码

***

上传此部分代码

```
num_tokens = np.array(num_tokens)
len(num_tokens)
```

第 6 题：请计算并展示所有句子的平均长度。1 分

```
np.max(num_tokens)
###### 上传此部分代码 ######
***
###### 上传此部分代码 ######
plt.hist(np.log(num_tokens), bins = 100)
plt.xlim((0,10))
plt.ylabel('number of tokens')
plt.xlabel('length of tokens')
plt.title('Distribution of tokens length')
plt.show()
# 假设 tokens 长度的分布为正态分布，则 max_tokens 这个值可以涵盖 95%左右的样本
max_tokens = np.mean(num_tokens) + 2 * np.std(num_tokens)
max_tokens = int(max_tokens)
max_tokens
# 取 tokens 的长度为 236 时，大约 95%的样本 tokens 长度在 236 附近
```

```
# 我们对长度不足的进行 padding，超长的进行修剪
np.sum( num_tokens < max_tokens ) / len(num_tokens)
#定义由标签返回原文本的函数，方便 Debug
def reverse_tokens(tokens):
    text = ''
    for i in tokens:
        if i != 0:
            text = text + cn_model.index_to_key[i]
        else:
            text = text + ' '
    return text
reverse = reverse_tokens(train_tokens[0])
# 经过 tokenize 再恢复成文本
# 可见标点符号都没有了
reverse
# 原始文本
train_texts_orig[0]
# 只使用前 50000 个词
num_words = 50000
# 初始化 embedding_matrix，之后在 keras 上进行应用
```

```
embedding_matrix = np.zeros((num_words, embedding_dim))
# embedding_matrix 为一个 [num_words，embedding_dim] 的矩阵
# 维度为 50000 * 300
for i in range(num_words):
    embedding_matrix[i,:] = cn_model[cn_model.index_to_key[i]]
embedding_matrix = embedding_matrix.astype('float32')
# 检查 index 是否对应，
# 输出 300 意义为长度为 300 的 embedding 向量一一对应
np.sum(cn_model[cn_model.index_to_key[333]] == embedding_matrix[333] )
# # embedding_matrix 的维度，
## 这个维度为 keras 的要求，后续会在模型中用到
embedding_matrix.shape
# 进行 padding 和 truncating， 输入的 train_tokens 是一个 list
# 返回的 train_pad 是一个 numpy array
train_pad = pad_sequences(train_tokens, maxlen=max_tokens,
                          padding='pre', truncating='pre')
# 超出五万个词向量的词用 0 代替
train_pad[train_pad>=num_words ] = 0
# 可见 padding 之后前面的 tokens 全变成 0，文本在最后面
train_pad[33].shape
```

准备 target 向量，前 2000 样本为 1，后 2000 为 0

train_target = np.array(train_target)

第 7 题：请使用 train_test_split 划分训练集和验证集，其中训练集占比 80%, 并且设置随机种子。2 分

变量命名要求：

训练数据：X_train

训练标签：y_train

测试数据：X_test

测试标签：y_test

上传此部分代码

***

上传此部分代码

第 8 题：构建神经网络模型，第一层为 embedding 层，之后的层为 bilstm 层或 lstm 层，也可以自己设定，最后为全连接层。4 分

上传此部分代码

***

上传此部分代码

第 9 题：编译模型，设置损失函数为二元交叉熵损失。2 分

```
    ###### 上传此部分代码 ######

    ***

    ###### 上传此部分代码 ######

    model.summary()

    path_checkpoint = './data/sentiment_checkpoint.keras'

    checkpoint = ModelCheckpoint(filepath=path_checkpoint, monitor='val_loss',
                                 verbose=1,
save_weights_only=True,
                                 save_best_only=True)

    try:
        model.load_weights(path_checkpoint)
    except Exception as e:
        print(e)


    earlystopping = EarlyStopping(monitor='val_loss', patience=5, verbose=1)


    lr_reduction = ReduceLROnPlateau(monitor='val_loss',
                                     factor=0.1,        min_lr=1e-8,
patience=0,
```

verbose=1)

callbacks = [

earlystopping,

checkpoint,

lr_reduction

]

第 10 题：使用训练集训练模型。epochs=10，batch_size=128，并调用回调函数。2 分

上传此部分代码

***

上传此部分代码

第 11 题：评估测试集准确率。2 分

上传此部分代码

***

上传此部分代码

print('Accuracy:{0:.2%}'.format(result[1]))

```
    def predict_sentiment(text):
        print(text)
        # 去标点
        text = re.sub("[\s+\.\!\/_,$%^*(+\"\']+|[+——！，。？、~@#￥%……&*（）]+", "",text)
        # 分词
        cut = jieba.cut(text)
        cut_list = [ i for i in cut ]
        # tokenize
        for i, word in enumerate(cut_list):
            try:
                cut_list[i] = cn_model.key_to_index[word]
                if cut_list[i] >= 50000:
                    cut_list[i] = 0
            except KeyError:
                cut_list[i] = 0
        # padding
        tokens_pad = pad_sequences([cut_list], maxlen=max_tokens,
                                   padding='pre', truncating='pre')
        # 预测
```

```
        result = model.predict(x=tokens_pad)
        coef = result[0][0]
        if coef >= 0.5:
            print('是一例正面评价','output=%.2f'%coef)
        else:
            print('是一例负面评价','output=%.2f'%coef)
test_list = [
        '酒店设施不是新的，服务态度很不好',
        '酒店卫生条件非常不好',
        '床铺非常舒适',
        '房间很凉，不给开暖气',
        '房间很凉爽，空调冷气很足',
        '酒店环境不好，住宿体验很不好',
        '房间隔音不到位',
        '晚上回来发现没有打扫卫生',
        '因为过节所以要我临时加钱，比团购的价格贵'
]
for text in test_list:
        predict_sentiment(text)
```