2025

BRICS SKILLS COMPETITION

(BRICS+ FUTURE SKILLS & TECH CHALLENGE)

Application of AI Technology

BRICS-FS-56

Test Project

(International Final_Onsite)

May, 2025

Contents

1. Introduction

1.1 Event Name

Application of AI technology Skill of the 2025 BRICS Skills Competition (BRICS+ Future Skills Challenge). Code of Skill: BRICS-FS-56.

1.2 Competition Format

Individual Championship

1.3 Event Description

The application of artificial intelligence technology event of the 2025 BRICS Skills Competition will be held, it focuses on the basic theories, technical applications and practical skills of artificial intelligence, aiming to assess students' abilities in data processing, algorithm design, programming implementation, model training, etc. This competition is mainly aimed at majors related to the new generation of information technology. The specific modules examined include four modules: image processing t, application of machine learning, application of deep learning, and application development of NLP. Achieve the goal of cultivating internationalized, highly skilled, future-oriented technical and skilled talents. The competition is provided with a competition environment and assessment system by a professional artificial intelligence skills competition platform. Contestants complete the task assessment through offline methods. The international finals of this competition are individual matches.

2. Participants and Competition Content

The participants of this skills competition are full-time students currently enrolled in higher vocational colleges and technical colleges. They should use Python and be based on mainstream AI frameworks at home and abroad such as OpenCV,

TensorFlow, and PyTorch. Using classic machine learning algorithms, open-source algorithms of computer vision, convolutional neural networks (CNN), recurrent neural networks (RNN), long short-term memory networks (LSTM), and other technologies, complete the application and development of models related to OpenCV image processing, machine learning, deep learning, natural language processing, and other modules. The assessment content is as follows:

Module A: Image Processing Technology

Using OpenCV, the basic operations of images, image processing, and the extraction and analysis of image features are accomplished. Including but not limited to reading and display, image format conversion, acquisition of basic image attributes, image saving, image filtering, color space conversion, edge detection, etc.

Module B: Application of Machine Learning Algorithms

It mainly examines the application of classic algorithms of machine learning, data preprocessing, feature engineering, model selection and optimization, model evaluation and other knowledge contents, including but not limited to data understanding and preprocessing, feature engineering, model selection and optimization, model evaluation and verification, etc.

Module C: Application of Deep Learning Technology

The main focus is on the development of models in the fields of image classification, object detection, and semantic segmentation, based on deep learning frameworks such as TensorFlow and Pytorch. Deep learning technologies include but not limited to dataset invocation, data preprocessing, deep network construction, model training, model testing, and model application.

Module D: Application Development of Natural Language Processing

Based on practical problems, it examines text classification, sentiment analysis, machine translation, question-answering systems, text generation, etc. Including but not limited to text cleaning, word segmentation and annotation, feature extraction of bag-of-words models, model selection and optimization, evaluation and validation, etc.

3. Time Allocation and Competition Tasks

3.1 Time Allocation

The competition consists of four modules, including Module A: Image Processing Technology. Module B: Application of Machine Learning Algorithms. Module C: Application of Deep Learning Technology. Module D: Development of Natural Language Processing Applications. Participants are required to complete the quiz within 6 hours, and the quiz time for each module will be allocated by the participants themselves.

3.2 Competition Tasks

Module A: Image Processing Technology

This module mainly based on the OpenCV, completes basic image operations, regenerates classic image processing algorithms, such as edge detection, histogram equalization, filtering and denoising, etc., and verifies their robustness on different types of image data.

Project 1: Graphic detection

Case Illustration：

With the development of computer vision technology, the recognition and measurement of target objects in images have been widely applied in multiple fields such as industrial inspection, smart metering, and automatic control. Traditional measurement methods have high human participation, low efficiency and are prone to errors, while image processing technology can meet the requirements of non-contact, automated and batch measurement.

Coins, as typical regular circular objects, are often used in teaching and experiments in image recognition. Through the automatic recognition and position analysis of coins in the image, functions such as object recognition, counting, positioning and distance estimation can be further realized.

```
import cv2
import numpy as np
```

Question 1: Use opencv to read the image and assign it to img（1 point）

Upload this part of the code

***

Upload this part of the code

#Copy the original image and use median filtering for noise reduction

o = img.copy()

o = cv2.medianBlur(o, 5)

Question 2: Transform image o from a color image to a single-channel grayscale image（1 point）

Upload this part of the code

***

Upload this part of the code

Question 3: Use the Hough transform of opencv to detect rings in an image（2 points）

Upload this part of the code

***

Upload this part of the code

Question 4: Round the detected data to an integer（1 point）

Upload this part of the code

***

Upload this part of the code

#Print out the obtained test results (the horizontal and vertical coordinates of the center of the circle and the radius of the circle)

print(circles[0])

Question 5: Draw all the detected coins in the original image img, and the drawing color should be red（2 points）

Upload this part of the code

***

Upload this part of the code

Question 6: Color the coin in the upper right corner green（2 points）

Upload this part of the code

***

Upload this part of the code

Question 7: Calculate the distances between the centers of all coins and print out the maximum distance（2 points）

Upload this part of the code

***

Upload this part of the code

Question 8: Show the image after the above operation（1 point）

Upload this part of the code

***

Upload this part of the code

Project 2: Statistics of Platform Numbers

Case Illustration:

In railway transportation, traffic dispatching and urban infrastructure management, accurately identifying and counting the number of platforms is of great value for transport capacity planning, resource dispatching and safety supervision. The traditional approach relies on manual inspection or manual marking, which has problems of low efficiency and easy misjudgment.

This project automatically recognizes the platform (or bus stop sign, landmark) patterns that appear in image or video frames through the Template Matching algorithm in OpenCV, achieving precise counting without manual intervention, and providing solution support for the integrated application of intelligent transportation systems and image recognition technology.

```
import cv2
   image = cv2.imread("image.png")  # Read the original image
templ = cv2.imread("templ1.png")  # Read the template image
```

Question 1: Take the height, width and number of channels of the template image（1 point）

Upload this part of the code

***

Upload this part of the code

Question 2: Use the matchTemplate function of OpenCV to find the position of the template image in the image. The matching method: Use the normalized correlation coefficient matching method（2 points）

Upload this part of the code

```
***
###### Upload this part of the code ######
station_Num = 0   # The number of platforms for initializing the express rail is 0
for y in range(len(results)):   # Traverse the rows of the result array
    for x in range(len(results[y])):   # Traverse the columns of the result array
        if results[y][x] > 0.99:
            # Question 3: Draw a blue rectangular border at the position where the match is successful（2 points）
            ###### Upload this part of the code ######
            ***
            ###### Upload this part of the code ######

#           Question 4: Count the number of platforms that have successfully matched（2 points）
            ###### Upload this part of the code ######
            ***
            ###### Upload this part of the code ######

Question 5: Write red text in the upper left corner of the image，to display“the numbers of stations: [The number of stations (specific quantity)]”（1 point）
###### Upload this part of the code ######
***
###### Upload this part of the code ######
cv2.imshow("result", image)
cv2.waitKey()
cv2.destroyAllWindows()
```

Module B: Application of Machine Learning Algorithms.

Contestants need to use Python and mainstream machine learning libraries (such as Scikit-learn, TensorFlow Lite, Pandas, etc.) to complete the full-process development from data preprocessing, feature engineering to model training and evaluation.

Project 3: PCA Data dimension reduction

Case Illustration:

With the development of artificial intelligence, the dimension of data is getting higher and higher. Direct modeling in data with high feature dimensions (such as images, texts, and medical data) may face problems such as dimension disaster, waste of computing resources, and model overfitting. Principal Component Analysis (PCA) is the most commonly used linear dimension reduction algorithm, which can effectively extract the most informative part of the data and achieve compression, visualization and feature optimization.

This project takes two typical datasets as the research objects: the iris dataset and the face image dataset

```
import matplotlib.pyplot as plt
# Question 1: Import the load_iris function from scikit-learn for loading the iris dataset（1 points）
###### Upload this part of the code ######
***
###### Upload this part of the code ######
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]:
#Question 2: Use the scatter function of Matplotlib to draw a scatter plot（2 points）
    ###### Upload this part of the code ######
```

```
        ***
        ###### Upload this part of the code ######
        ,X_dr[y == i, 1]
        ,alpha=0.7#Transparency
        ,c=colors[i]
        ,label=iris.target_names[i])
    # Question 3: Display the legend and indicate different categories.（1 points）
    ###### Upload this part of the code ######
    ***
    ###### Upload this part of the code ######
    plt.title('PCA of IRIS dataset')
    plt.show()
    Explore the data after dimensionality reduction
    pca.explained_variance_
    #Check the percentage of information occupied by each new feature vector after
dimensionality reduction in the total information of the original data
    pca.explained_variance_ratio_
      pca.explained_variance_ratio_.sum()
    X.var(axis=0)
    X.var(axis=0).sum()
    Cumulative explainable variance contribution rate curve
    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)
    # Question 4: Use to draw a line graph. The horizontal coordinate is [1,2,3,4],
The vertical coordinate is np.cumsum(pca_line.explained_variance_ratio_)（2 points）
    ###### Upload this part of the code ######
    ***
    ###### Upload this part of the code ######
```

```
#Question 5: Set the scale of the X-axis to1,2,3,4（1 points）
###### Upload this part of the code ######
***
###### Upload this part of the code ######
plt.xlabel("number of components after dimension reduction")
plt.ylabel("cumulative explained variance ratio")
plt.show()
Maximum likelihood estimation with self-selected hyperparameters
pca_mle = PCA(n_components="mle")
# Question 6: Fit the data X using instances of the pca_mle model（2 points）
###### Upload this part of the code ######
***
###### Upload this part of the code ######
X_mle = pca_mle.transform(X)
X_mle.shape

#It can be found that mle automatically selects three features for us

pca_mle.explained_variance_ratio_
pca_mle.explained_variance_ratio_.sum()
Select the hyperparameters according to the proportion of information volume
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_
The Application of Attribute components_ in Face Recognition
# Question 7: Import the fetch_lfw_people function from scikit-learn for loading the LFW(Face image dataset)（1 points）
###### Upload this part of the code ######
***
```

```
###### Upload this part of the code ######
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 is the number of images
X = faces.data
fig, axes = plt.subplots(3,8
                         ,figsize=(8,4)
                         ,subplot_kw                    =
{"xticks":[],"yticks":[]}                    )
```

Question 8: Display the first several images in the face image dataset in the subplots previously created using plt.subplots(), and show the images as grayscale plots.（2 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
#Originally, there were 2,900 dimensions. Now let us reduce it to 150 dimensions
pca = PCA(150).fit(X)
V = pca.components_
V.shape
fig, axes = plt.subplots(3,8,figsize=(8,4),subplot_kw = {"xticks":[],"yticks":[]})
```

Question 9: Restore the principal component vector (feature face) obtained after PCA dimensionality reduction to the shape of the image and plot it in the subgraph grid, presenting it in the form of a grayscale image（2 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######


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")
```

Project 4: Data preprocessing and feature engineering of data

Case Illustration:

In modern machine learning projects, data preprocessing and feature engineering have become the core elements that affect the model performance. Whether it is structured table data, time series data, or user behavior data, the original data usually has problems such as missing values, outliers, noise, and unstructured fields, which directly affect the training effect of the model. This case focuses on the 'preprocessing' stage before data modeling, aiming to enhance the contestants' understanding, processing and feature construction capabilities of the data, and lay a solid foundation for subsequent predictive modeling.

```
import pandas as pd
import numpy as np

# Prevent partial warnings
import warnings
warnings.filterwarnings("ignore")

# Data visualization
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# Label processing of data
from sklearn.preprocessing import LabelEncoder

#Chi-square test
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
```

Question 1: Read data from the CSV file named 'first_round_training_data.csv'（1 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
```

1.Data Exploration (Data EDA)

Question 2: Display the basic information of the data, such as the number of rows, the number of columns, the type of each column, and the number of non-null values（1 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
train_data.head()
```

Question 3: Obtain all column names except the 'Quality_label' column and store them in the variable col_name（2 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
Notdlts_count = []
for i in col_name:
    # Calculate the number of non-repetitive values
    Notdlts = len(train_data[i].drop_duplicates())/6000
    Notdlts_count.append(Notdlts)
```

```
    plt.plot(col_name, Notdlts_count, c='r')
    plt.title('Calculation of the total number of non-repetitive values') # Title
    plt.xlabel('Column name')                                  # The name of the X-axis
    plt.ylabel('The proportion of non-repetitive data in the full data')  # The name of the Y-axis
    plt.xticks(rotation=45)                                    # Rotate the scale name of the X-axis
    plt.show()
    # Extract all the features
    unit = train_data.drop([ 'Quality_label'], 1)
    The distribution differences of the data
    # Traverse the column names
    for i in col_name:
        plt.hist(unit[i], bins=20)
        plt.title('%s Count statistics chart of the average segmentation value range'%i)
        plt.xlabel('%sRange'%i)
        plt.ylabel('The number of values within this range')
        plt.show()
    The degree of data dispersion - Look at the standard deviation of the data
    Question 4: Obtain all the column names of the data frame unit and store them in col_name（2 points）
    ###### Upload this part of the code ######
    ***
    ###### Upload this part of the code ######
    Question 5: Calculate the standard deviation (std) of each column, transpose and retrieve the corresponding value, and store it in col_std（1 points）
    ###### Upload this part of the code ######
    ***
    ###### Upload this part of the code ######
    plt.plot(col_name, col_std, c='red')  # Make a drawing
    plt.title('Column - Standard deviation')       # Title
    plt.xlabel('Column name')                 # The name of the X-axis
```

```
plt.ylabel('Standard deviation')              # The name of the Y-axis
plt.xticks(rotation=90)          # Rotate the scale name of the X-axis
plt.show()
Label processing of data
lb = LabelEncoder()


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

Question 6: Perform the fourth root operation (i.e., 1/4 of each value) on the values of all columns corresponding to col_name in the unit data frame.（1 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
# Traverse the column names
for i in col_name:
    plt.hist(unit[i], bins=20)
    plt.title('%s Count statistics chart of the average segmentation value range'%i)
    plt.xlabel('%sRange'%i)
    plt.ylabel('The number of values within this range')
    plt.show()
Remove the standard deviation of the data
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('Column - Standard deviation')
plt.xlabel('Column name')
plt.ylabel('Standard deviation')
plt.xticks(rotation=90)
plt.legend(['Standard deviation','Contour lines：10'])
plt.show()
```

Question 7: Perform a logarithmic transformation on each value of the column corresponding to col_name in the unit data frame (i.e., calculate log(value +1)), and store the transformed result back to the original position.（2 points）

```
###### Upload this part of the code ######
```

```
***
###### Upload this part of the code ######
# Calculate the standard deviation after transformation(std)
col_log_std = unit[col_name].describe().T['std']

plt.plot(col_name, col_log_std, c='red')
plt.title('Column - Standard deviation')
plt.xlabel('Column name')
plt.ylabel('Standard deviation')
plt.xticks(rotation=90)
plt.show()
```

Question 8: For each column i in the unit data frame, perform normalization processing and convert each value to between 0 and 1.（2 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
```

Feature selection

Question 9: Using the SelectKBest method and based on the chi-square test (chi2), select the top 14 most important features.（2 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
```

Question 10: Use the fit method to conduct feature screening on the unit feature and the corresponding target label train_data['Quality_label'].（2 points）

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
# Print the chi-square test value
print(fit.scores_)
train = pd.DataFrame(fit.transform(unit),columns=['V{0}'.format(i) for i in range(1, 15)])
train.head()
```

Module C: Application of Deep Learning Technology

Based on convolutional neural networks, recurrent neural networks, etc., and with TensorFlow, Pytorch, etc. as frameworks, deep neural networks were built and deep models were trained to complete the application development such as image recognition, image classification, and semantic segmentation.

Project 5: Image Classification of Cats and Dogs

Case Illustration:

Image classification is one of the fundamental tasks in the field of computer vision and is widely applied in scenarios such as security monitoring, medical diagnosis, and autonomous driving. Cat and dog recognition, as a classic introductory case of image classification, can help learners understand the preprocessing process of image data, the basic structure of convolutional neural networks, as well as the complete process of model training and evaluation. This project aims to train an image classification model capable of automatically recognizing cats and dogs through deep learning methods.

```
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/'
```

Question 1: Count the number of files (or subfolders) in the directory train_cats_dir (2 points)

```
###### Upload this part of the code ######
***
###### Upload this part of the code ######
# Build the training dataset
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)                # Read the original file
        image_decoded = tf.image.decode_jpeg(image_string)    # Decode JPEG
images
    #Question 2: Adjust the image to 256×256 pixels and normalize it（2 points）
        ###### Upload this part of the code ######
        ***
        ###### Upload this part of the code ######
        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())

    def _decode_and_resize(filename, label):
        image_string = tf.io.read_file(filename)              # Read the original file
        image_decoded = tf.image.decode_jpeg(image_string)    # Decode JPEG
images
        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)

    # Take out the previous buffer_size data and put it into the buffer, randomly sample from it, and replace the sampled data with subsequent data
    train_dataset = train_dataset.shuffle(buffer_size=23000)

    Question 3: Repeat the training set three times.（2 points）
    ###### Upload this part of the code ######
    ***
    ###### Upload this part of the code ######
    Question 4: Divide the training set into batches of fixed size.（2 points）
    ######  Upload this part of the code ######
    ***
    ######  Upload this part of the code ######
    Question 5: Pre-load the data of the next batch to optimize the training performance.（2  points）
    ######  Upload this part of the code ######
    ***
    ######  Upload this part of the code ######
    # Build the test data set
    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()
```

Question 6: Flattening operation, converting multi-dimensional feature maps into one-dimensional vectors.（2 points）

```
        ###### Upload this part of the code ######
        ***
        ###### Upload this part of the code ######
```

Question 7: Define a fully connected layer containing 64 neurons and use the ReLU activation function（2 points）

```
        ###### Upload this part of the code ######
        ***
        ###### Upload this part of the code ######
```

Question 8: Define a fully connected layer containing two neurons and use the Softmax activation function（2 points）

```
        ###### Upload this part of the code ######
        ***
        ###### Upload this part of the code ######

    def call(self, x):
        x = self.conv1(x)
        x = self.maxpool1(x)
#       Question 9: After the second convolutional layer conv2（2 points）
```

```
            ###### Upload this part of the code ######
            ***
            ###### Upload this part of the code ######
            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 has no one-hot

    # Question 10: Create an Adam optimizer object（2 points）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code ######

    # Question 11: An index object for calculating training losses has been created（2
points）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code ######

    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:
# Question 12: Make predictions through the model(images)（2 points）
        ###### Upload this part of the code ######
            ***
        ###### Upload this part of the code ######

# Question 13: Calculate the loss between the predicted value and the true value (labels).（2 points）
        ###### Upload this part of the code ######
            ***
        ###### Upload this part of the code ######
        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):
    # At the beginning of the next epoch, reset the evaluation metrics
    train_loss.reset_states()
    train_accuracy.reset_states()
    test_loss.reset_states()
    test_accuracy.reset_states()
```

```
        for images, labels in train_dataset:
    # Question 14: Call the defined training function to train this batch of data（2 points）
            ###### Upload this part of the code ######
                ***
            ###### Upload this part of the code ######

        for test_images, test_labels in test_dataset:
    # Question 15: Call the function test_step() to process the test data of the current batch（2 points）
            ###### Upload this part of the code ######
                ***
            ###### Upload this part of the code ######

        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
                             ))
```

Module D: Application Development of Natural Language Processing

Contestants are required to use Python and mainstream NLP tool libraries (such as NLTK, spaCy, Transformers, etc.) to complete the entire development process from data cleaning, model training to server-side deployment.

Project 6: News Classification Based on LSTM Algorithm

Case Illustration:

In the era of information explosion, a vast amount of news is generated and disseminated every day. To help users quickly obtain the content they are interested in, news platforms need to precisely classify news texts. Traditional machine learning methods have difficulty in capturing the word order and contextual relationship of texts. Therefore, introducing the LSTM model in deep learning can better understand

the semantics of news and achieve more accurate classification.

```
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)
# Set the TensorFlow thread configuration
tf.config.threading.set_intra_op_parallelism_threads(16)  # The number of parallel threads within a single operation
tf.config.threading.set_inter_op_parallelism_threads(16)  # The number of parallel threads between different operations
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'])

# Question 1: A function sample_by_label is defined to randomly draw up to 500 samples for each category from the dataset（1 point）

###### Upload this part of the code ######
        ***
###### Upload this part of the code ######

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)

# Question 2: Call the sample_by_label function on the train dataset and randomly draw up to 500 samples from each category（1 point）
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######

val = val_by_label(val)
test = test_by_label(test)
train.shape
val.shape
test.shape
#Question 3: Obtain the content of the content column corresponding to the first row in the train dataset（1 point）
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
train.shape
def content_cut(x):
#Question 4: Perform Chinese word segmentation on the input text x（1 point）
    ###### Upload this part of the code ######
        ***
    ###### Upload this part of the code ######
    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))
# Question 5: Merge the train, val and test datasets along the row direction to form a large dataset df（1）
```

```
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
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))
```

Question 6: Use a line graph to display the values of the content_len column in the df dataset. Each point is marked with a red asterisk (*)（1 point）

```
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
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']
```

Question 7: Set the font of the Matplotlib chart to 'SimHei'（1 point）

```
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
```

Question 8: Count the number of samples for each category (label) in the train dataset and store the results in count_class（1 point）

```
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
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("Category name",fontsize=25)
    import os
    # Question 9: Define a variable file_name to store the path of the Word2Vec
model（1 point）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code ######
    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 Coding
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
lb = LabelEncoder()
# Question 10: Use the label encoder (lb) to convert the category labels of the label column in the train dataset into digital labels and store them in train_label（1 points）
###### Upload this part of the code ######
    ***
###### Upload this part of the code ######
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))

    #Coding layer
    #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)

    #Pooling layer
    avg_pool = tf.keras.layers.GlobalAveragePooling1D()(x) #（batch,400）
    # Question 11: Use the global Max pooling layer to pool the input x（1 point）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code ######

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

    x = tf.keras.layers.Dense(1000)(conc)
    # Question 12: Batch normalization of the x-tensor（1 point）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code ######
    x = tf.keras.layers.Activation(activation="relu")(x)

    # Question 13: Add a Dropout layer, operate on x, and randomly discard 20% of
the neurons（1 point）
    ###### Upload this part of the code ######
            ***
```

```
###### Upload this part of the code ######
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
```

Question 14: Convert the training data and the corresponding labels into a TensorFlow dataset object, train_dataset（2 points）

```
###### Upload this part of the code ######
          ***
###### Upload this part of the code ######
for a,b in train_dataset.take(1):
    print(a.shape,b.shape)

import tensorflow as tf
from tqdm import tqdm

# Configuration parameters
learning_rate = 0.001
BATCH_SIZE = 1024  # Adjust according to memory
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)

        # Set the parallel processing options
        options = tf.data.Options()
        options.threading.private_threadpool_size = 16  # Adjust according to the number of CPU cores
        options.threading.max_intra_op_parallelism = 16
        dataset = dataset.with_options(options)

        return dataset

    # Prepare the dataset
    # Question 15: Call the configure_dataset function to configure train_dataset and enable random shuffling of data（1 point）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code ######
    val_dataset = configure_dataset(tf.data.Dataset.from_tensor_slices((val_, val_label)),
                                    batch_size=VAL_BATCH_SIZE)
        # Model and optimizer
    # Question 16: Create a classification cross-entropy loss function object loss_object（1point）
    ###### Upload this part of the code ######
            ***
    ###### Upload this part of the code######
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
```

```
# Indicator
# Question 17: Create an index object named train_loss for recording the average loss value during the training process（1 point）
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
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)
#Question 18: Update the true label y and the model's predicted results, predictions, to the validation accuracy metric val_accuracy（1 point）
    ###### Upload this part of the code ######
        ***
    ###### Upload this part of the code ######

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()

        # Training stage
        for x_batch, y_batch in tqdm(train_dataset, desc=f'Train Epoch {epoch+1}'):
            train_one_step(x_batch, y_batch)

        # Verification stage
        for x_batch, y_batch in val_dataset:
            val_one_step(x_batch, y_batch)

        # Print the result
        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}%")

# Training cycle
for epoch in range(EPOCHS):
    train_for_epoch(train_dataset, val_dataset, epoch)
# Question 19: Print the structural information of the model（1 point）
###### Upload this part of the code ######
        ***
###### Upload this part of the code ######
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_)))
```