import numpy as np  
import pandas as pd  
from keras.models import Sequential  
from keras.layers import Dense, Dropout  
from keras.layers import Conv1D, MaxPooling1D, Flatten  
from keras.optimizers import Adam  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
import matplotlib.pyplot as plt  
# 加载数据  
def load\_data(csv\_file):  
 data = pd.read\_csv(csv\_file)  
 X = data.iloc[:, :-1].values  
 y = data.iloc[:, -1:].values  
 return X, y  
  
# 构建模型  
  
  
# def build\_classification\_model(input\_shape, num\_labels):  
# model = Sequential()  
# model.add(Dense(128, activation='relu', kernel\_initializer='uniform', input\_shape=input\_shape))  
# model.add(Dropout(0.2))  
# model.add(Dense(256, activation='relu', kernel\_initializer='uniform'))  
# model.add(Dropout(0.2))  
# model.add(Dense(512, activation='relu', kernel\_initializer='uniform'))  
# model.add(Dropout(0.2))  
# model.add(Dense(256, activation='relu', kernel\_initializer='uniform'))  
# model.add(Dropout(0.2))  
#  
# # 根据分类问题的类型选择激活函数  
# if num\_labels == 2: # 二分类问题  
# model.add(Dense(1, activation='sigmoid', kernel\_initializer='uniform'))  
# loss\_function = 'binary\_crossentropy'  
# else: # 多分类问题  
# model.add(Dense(num\_labels, activation='softmax', kernel\_initializer='uniform'))  
# loss\_function = 'categorical\_crossentropy'  
#  
# # 编译模型  
# model.compile(loss=loss\_function, optimizer='adam', metrics=['accuracy'])  
# return model  
  
  
from tensorflow.keras.models import Sequential,Model,load\_model  
from tensorflow.keras.layers import Dense, Dropout, Conv1D, Flatten, MaxPooling1D, GlobalMaxPooling1D,LSTM,Reshape  
from tensorflow.keras.layers import Layer, Dense, Activation, Permute, Lambda  
import tensorflow.keras.backend as K  
  
class SelfAttention(Layer):  
 def \_\_init\_\_(self, \*\*kwargs):  
 super(SelfAttention, self).\_\_init\_\_(\*\*kwargs)  
  
 def build(self, input\_shape):  
 self.W = self.add\_weight(name='attention\_weight',  
 shape=(input\_shape[-1], 1),  
 initializer='random\_normal',  
 trainable=True)  
 self.b = self.add\_weight(name='attention\_bias',  
 shape=(input\_shape[1], 1),  
 initializer='zeros',  
 trainable=True)  
 super(SelfAttention, self).build(input\_shape)  
  
 def call(self, x):  
 e = K.tanh(K.dot(x, self.W) + self.b)  
 e = K.exp(e)  
 a = e / K.sum(e, axis=1, keepdims=True)  
 output = x \* a  
 return output  
  
 def compute\_output\_shape(self, input\_shape):  
 return input\_shape  
  
  
def build\_model(input\_shape, num\_labels):  
 model = Sequential()  
 # Input layer  
 model.add(Conv1D(filters=8, kernel\_size=80, strides=20, activation='relu', input\_shape=input\_shape))  
 model.add(Dropout(0.2))  
 # 添加自注意力层  
 model.add(SelfAttention())  
 model.add(Flatten()) # 或者使用 GlobalMaxPooling1D()  
 model.add(Dense(num\_labels, activation='softmax'))  
 model.summary()  
 model.compile(loss='categorical\_crossentropy', optimizer=Adam(learning\_rate=0.001), metrics=['accuracy'])  
 return model  
  
# 假设输入数据形状为 (100, 1)，即序列长度为 100，输入通道数为 1  
# 假设有 10 个类别  
# input\_shape = (100, 1)  
# num\_labels = 10  
#  
# model = build\_model(input\_shape, num\_labels)  
# model.summary()  
  
  
  
# 主函数  
def main():  
 csv\_file = 'E:\\libs\\3sample.csv'  
 X1, y1= load\_data(csv\_file)  
 csv\_file = 'E:\\libs\\data1245.csv'  
 X, y = load\_data(csv\_file)  
 #scaler = StandardScaler()  
 #X\_scaled = scaler.fit\_transform(X)  
  
 # X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
 X\_train, X\_val, y\_train\_, y\_val\_ = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
 print(y\_train\_[1],y\_train\_[2],y\_train\_[3])  
  
  
  
 xuhao={'0':0,'1':1,'10':2,'100':3, '103':4, '104':5, '105':6, '106':7, '107':8, '110':9, '111':10, '116':11, '117':12, '12':13, '13':14, '14':15, '16':16, '2':17, '20':18, '24':19, '29':20, '3':21, '30':22, '32':23, '35':24, '36':25, '38':26, '39':27, '41':28, '42':29, '43':30, '46':31, '48':32, '51':33, '53':34, '54':35, '55':36, '56':37, '63':38, '68':39, '69':40, '70':41, '71':42, '72':43, '73':44, '77':45, '8':46, '81':47, '84':48, '85':49, '9':50, '97':51, '98':52}  
 #print(X\_train)  
 #print(y\_train)  
 y\_train=[[0 for j in range(53)] for i in range(len(y\_train\_))]  
 for i in range(len(y\_train\_)):  
 y\_train[i][xuhao[str(y\_train\_[i][0])]]=1  
 y\_train = np.array(y\_train, dtype=float)  
 y\_val = [[0 for j in range(53)] for i in range(len(y\_val\_))]  
 for i in range(len(y\_val\_)):  
 y\_val[i][xuhao[str(y\_val\_[i][0])]] = 1  
 y\_val = np.array(y\_val, dtype=float)  
  
 #model = build\_classification\_model((num\_features,), num\_labels)#(num\_features,), num\_labels  
 model = build\_model((7062,1), 53)  
 model.fit(X\_train, y\_train, epochs=100, batch\_size=8, validation\_data=(X\_val, y\_val), verbose=1)  
 #model = load\_model("conv1d+sf-att 100epoch.h5")  
 # middle1 = Model(inputs=model.get\_layer('conv1d').input, outputs=model.get\_layer('conv1d').output)  
 # middle2=Model(inputs=model.get\_layer('conv1d').input, outputs=model.get\_layer('self\_attention').output)  
 # middle = Model(inputs=model.get\_layer('conv1d').input, outputs=model.get\_layer('flatten').output)  
 #  
 #  
 # result = middle.predict(X\_train)[2]  
 # print(result.shape)  
 # hengzhou = np.array([\_ \* 0.1 for \_ in range(5600)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\4\\flatten2.jpg")  
 # plt.clf()  
 # result = middle.predict(X\_train)[3]  
 # print(result.shape)  
 # hengzhou = np.array([\_ \* 0.1 for \_ in range(5600)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\4\\flatten3.jpg")  
 # plt.clf()  
  
 # result = middle.predict(X\_train)[0][:, 1]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-2.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 2]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-3.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 3]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-4.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 4]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-5.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 5]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-6.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 6]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-7.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 7]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-8.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 8]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-9.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 9]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-10.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 10]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-11.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 11]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-12.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 12]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-13.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 13]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-14.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 14]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-15.jpg")  
 # plt.clf()  
 #  
 # result = middle.predict(X\_train)[0][:, 15]  
 # print(result.shape)  
 # hengzhou = np.array([\_ for \_ in range(350)])  
 # plt.scatter(hengzhou, result, s=1)  
 # plt.savefig("E:\\libs\\explainable\\3\\conv1d-16.jpg")  
 # plt.clf()  
 #  
 # # 评估模型  
 X\_test,y\_test\_=load\_data("E:\\libs\\data36.csv")  
  
 y\_test = [[0 for j in range(53)] for i in range(len(y\_test\_))]  
 for i in range(len(y\_test\_)):  
 y\_test[i][xuhao[str(y\_test\_[i][0])]] = 1  
 #scaler = StandardScaler()  
 #X\_test = scaler.fit\_transform(X\_test)  
 y\_pred = model.predict(X\_test)  
 #  
 # #print(y\_test,y\_pred)  
 acc=0  
 for i in range(len(y\_test)):  
 if np.argmax(y\_test[i])==np.argmax(y\_pred[i]):  
 acc+=1  
 print("acc:",round(acc/len(y\_test),4))  
  
 #pd.DataFrame(y\_test).to\_excel('CARS\_300sample\_23wei -5label-actual\_values.xlsx', index=False)  
 #pd.DataFrame(y\_pred).to\_excel('CARS\_300sample\_23wei -5label-predicted\_values.xlsx', index=False)  
 model.save('conv1d+sf-att 100epoch.h5') # 保存模型到文件  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()