### 案例 3：基于 CNN-LSTM 模型的城市区域交通流量预测

* **问题背景**：城市交通拥堵已成为制约城市发展的重要问题，精准的交通流量预测是智能交通系统的核心功能之一。交通流量具有明显的时空特性，某一区域的交通流量不仅与该区域历史流量相关，还与相邻区域的流量密切相关，同时随时间动态变化。
* **问题描述**：某市交通管理部门需要对城市核心商圈及周边 5 平方公里范围内的 100 个路口未来 24 小时的交通流量进行预测。要求模型能够同时考虑不同路口之间的空间关联（如主干道与支路的流量影响）和时间变化规律（如早晚高峰的流量特征），为交通信号配时优化和拥堵疏导提供依据。
* **数据情况**：提供过去 1 年的各路口每 15 分钟的交通流量数据（单位：辆 / 15 分钟），同时提供路口的位置关系图、道路等级、周边建筑功能（商业区、住宅区、办公区等）、节假日及天气数据。数据存在因设备故障导致的部分时段缺失。

### 案例 3：CNN-LSTM 模型城市区域交通流量预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.preprocessing import MinMaxScaler  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Flatten  from sklearn.metrics import mean\_squared\_error  import os  # 数据加载与预处理  def load\_traffic\_data(file\_path):  data = pd.read\_csv(file\_path, parse\_dates=['time'], index\_col='time')  return data.values # 形状为 (时间步, 路口数量)  traffic\_data = load\_traffic\_data('traffic\_flow.csv')  scaler = MinMaxScaler(feature\_range=(0, 1))  traffic\_scaled = scaler.fit\_transform(traffic\_data)  # 构建时空序列数据  def create\_spatio\_temporal\_data(data, time\_steps=8, pred\_steps=1):  X, y = [], []  for i in range(len(data) - time\_steps - pred\_steps + 1):  X.append(data[i:i+time\_steps, :]) # 时空特征 (时间步, 路口数)  y.append(data[i+time\_steps:i+time\_steps+pred\_steps, :]) # 预测目标  return np.array(X), np.array(y).squeeze()  time\_steps = 8 # 8个15分钟，共2小时  X, y = create\_spatio\_temporal\_data(traffic\_scaled, time\_steps)  # 划分训练集和测试集  train\_size = int(len(X) \* 0.8)  X\_train, X\_test = X[:train\_size], X[train\_size:]  y\_train, y\_test = y[:train\_size], y[train\_size:]  # 重塑输入形状 (样本数, 时间步, 特征数)  X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], X\_train.shape[2])  X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], X\_test.shape[2])  # 构建CNN-LSTM模型  model = Sequential()  # CNN层提取空间特征  model.add(Conv1D(filters=64, kernel\_size=3, activation='relu', input\_shape=(time\_steps, X\_train.shape[2])))  model.add(MaxPooling1D(pool\_size=2))  model.add(Conv1D(filters=32, kernel\_size=2, activation='relu'))  # LSTM层提取时间特征  model.add(LSTM(50, return\_sequences=False))  model.add(Dense(y\_train.shape[1])) # 输出每个路口的预测值  model.compile(optimizer='adam', loss='mse')  # 模型训练  history = model.fit(X\_train, y\_train, epochs=30, batch\_size=32, validation\_data=(X\_test, y\_test), verbose=1)  # 模型预测  y\_pred = model.predict(X\_test)  # 反归一化  y\_pred\_actual = scaler.inverse\_transform(y\_pred)  y\_test\_actual = scaler.inverse\_transform(y\_test)  # 评估模型  mse = mean\_squared\_error(y\_test\_actual.flatten(), y\_pred\_actual.flatten())  print(f'时空模型MSE: {mse}')  # 可视化第一个路口的预测结果  plt.figure(figsize=(12, 6))  plt.plot(y\_test\_actual[:, 0], label='实际流量')  plt.plot(y\_pred\_actual[:, 0], label='预测流量')  plt.legend()  plt.savefig('traffic\_prediction.png')  plt.show()  # 保存模型  model.save('cnn\_lstm\_traffic.h5') |