中国研究生创新实践系列大赛

“华为杯”第十八届中国研究生

数学建模竞赛

学　　校	华东理工大学
参赛队号	21102510017
队员姓名	1. 刘艳芳
	2. 佟雁羚
	3. 段利均

“华为杯”第十八届中国研究生数学建模竞赛

题　目　空气质量二次预测模型的构建分析

摘　　要：

随着人类日常生产活动以及自然过程的发生，某些物质进入大气中，经过一段时间的累积，到达足够的浓度，就有可能危害人类的健康甚至危害生态环境。因此，需要构建空气质量预测模型来提前预测可能发生的大气污染并采取相应管控措施。典型的预测模型通常基于 WRF-CMAQ 模拟体系，但是受限于当前模拟条件和污染物生成机理理论的研究，这些预测模型结果并不理想。因此，可以参考污染物浓度实测数据进行二次预测，本文在第一问中首先进行数据查看，按照规则计算 AQI 和首要污染物，发现污染物浓度和气象条件之间具有一定关联；在第二问中使用 KNN 插值法对缺失值进行拟合，使用 3σ 原则剔除异常值，使用皮尔逊相关系数和折线趋势图等分析气象条件对污染物的影响程度；在问题三中，本文考虑了监测站点间个性与共性的关系，即站点间符合同一的污染物变换规律且站点间污染物变化的规律受所在区域人文自然因素的影响，基于此提出了 GAN-Stacked-BiLSTM-MLP 模型建模适用于 A、B、C 三个监测点的二次预报数学模型；在第四问中，本文考虑到了监测站点间的时空相关性问题，提出了时空注意力预测模型 PGAN 对不同站点的污染物浓度进行精准预测。

针对问题一：首先进行数据查看，包括缺失值和异常值分析，使用了 3σ 原则和箱型图来进行异常值的探索和可视化；然后计算臭氧最大 8 小时滑动平均值，作为计算臭氧的 IAQI 值的基础；接下来根据 AQI 和首要污染物的定义计算并划分等级。

针对问题二：首先处理数据，剔除异常值，使用最近邻等多种插值方式进行拟合；然后使用皮尔逊相关系数分析气象因素之间的相关性、污染物之间的相关性以及二者之间的相关性；接下来通过绘制折线趋势图探索气象因素对污染物的影响，使用不同的时间粒度，查看气象因素和污染物浓度之间的相关性；最终得到气象分类和各类的特征。

针对问题三：对于跨站点多元污染物浓度通用预报模型。首先，本文从时序信号预测问题为切入点，选用改进的长短期记忆网络 LSTM 模型对由污染物浓度及其相关的气象数据构成的时序信号进行建模；其次，考虑到污染物浓度与气象数据间存在多元影响的复杂关系以及预测任务属性的多元性特点，对单变量的 LSTM 模型进行扩展，并采用迭代形式

预测未来三日的六种污染物浓度变化情况；最后，对于跨站点通用模型的目标，本文使用对抗生成网络 GAN 对站点间客观存在的差异性特点进行消除，使得模型在跨站点预测中拥有较强的泛化性能。本文提出了一种 GAN-Stacked-BiLSTM-MLP，适用于 A、B、C 三个监测点的二次预报数学模型，用来预测未来三天 6 种常规污染物单日浓度值。

针对问题四：对于多站点污染物浓度联合预测问题，我们从时空两个维度进行研究。时间维度，我们考虑到了污染物的动态连续性；空间维度，我们考虑到了污染物的空间动态扩散运动。本研究问题中，我们关注时空因素，并提出了一种污染物图注意网络（PGAN）来预测网络节点上不同位置前方时间步长的污染物浓度状况。PGAN 采用编码器-解码器架构，其中编码器和解码器均由多个时空注意块组成，以模拟时空因素对污染状况的影响。编码器对输入数据特征进行编码，解码器预测输出序列。在编码器和解码器之间，应用 transformer 注意力层来转换编码的输入数据特征，以生成未来时间步长的序列表示作为解码器的输入。Transformer 注意力机制对历史和未来时间步长之间的直接关系进行建模，这有助于缓解预测时间步长之间的错误传播问题。

关键词：污染物浓度预测；AQI；向量自回归模型 VAR；长短期记忆网络 LSTM；对抗生成网络 GAN；PCA；多元变量线性回归模型；时空注意力模型 PGAN；时空相关性

目 录

一. 问题重述

1.1 问题背景

随着我国以火力发电为代表的电力、热力生产和供应业等产业的蓬勃发展，以及居民生活、交通和城市化的快速发展，以 SO_2，NO_2，$PM_{2.5}$ 和 PM_{10} 等为代表的大气污染物排放日益增长。如图 1.1，以二氧化硫的排放情况为例，可以看到，来自工业源的 SO_2 排放量最高，其次是生活源，最后是集中式污染治理。虽然我国已经进行了多年的生态治理，但是 2019 年我国的 SO_2 排放总量仍然高达 457.3 万吨[1]。大量的污染气体排放，导致空气质量下降，对我国居民及生态环境造成巨大的危害。根据《2019 全球空气状况中文报告》[2]，大气污染会加重心脏病、慢性呼吸道疾病、肺部相关疾病的患病率，全球大约有 490 万人因空气污染过早死亡，同时也造成人均预期寿命缩短了 20 个月。为此，迫切需要建立空气质量预测模型来提前预测大气污染情况，并采取相应的管控措施。

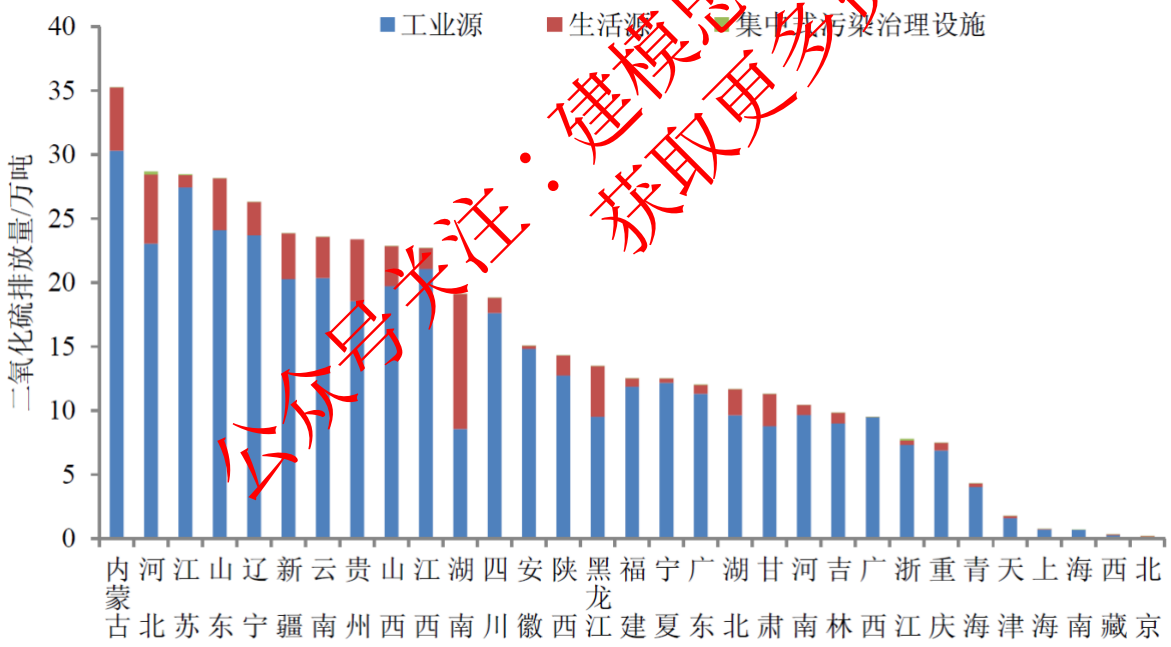

图 1.1 2019 年各地区二氧化硫排放情况

当前的空气质量预测模型主要基于 WRF-CMAQ 模拟体系，而该体系对模拟的气象场及排放清单非常敏感，同时臭氧作为《环境空气质量标准》（GB3095-2012）中唯一的二次污染物，其生成机理非常复杂且并不完全明晰，如图 1.2 城市（区域）O3 部分污染成因概念图所示[5]。因此，单纯基于 WRF-CMAQ 模型得到的空气质量预测结果并不理想。考虑到实测近地表风速、风向及压强等气象条件，以及过去污染物浓度实测数据对空气质量的预测有一定的帮助，所以可以结合特定气象监测点及空气质量监测点的检测数据进行二次建模，以提高预测的准确性。

图 1.2 城市（区域）O3 污染成因概念图

1.2 问题解析

从上述背景中不难得出，随着气象条件日渐复杂，变化规律的显著性降低，单纯依靠地理和气象场模拟得到气象场数据的 WRF 难以为 CMAQ 提供有效的数据支持，同时在污染物形成机理不十分明晰的情况下，基于物化反应模拟污染物变化的 CMAQ 得到的结果并不十分准确。为此本文结合历史实际气象条件和历史污染物浓度实测数据，使用机器学习（包括深度学习）的方法进行二次建模。目的是通过给出的监测点的数据构建适用于多个站点的预测模型，同时考虑相邻监测点间空间影响，保证 AQI 预报值的最大相对误差尽可能小，同时首要污染物预测准确率尽可能高。主要包括以下几个问题：

问题 1：按照指定方法计算监测点的实测 AQI 和首要污染物。首先查看数据，包括数据异常和缺失情况；然后计算臭氧最大 8 小时滑动平均，以此作为计算臭氧 IAQI 值的基准；最后按照 AQI 的计算规则得到 AQI 值和首要污染物。这一过程中发现臭氧和气象条件之间存在一定的关联，为第二问打下基础。

问题 2：首先对数据进行预处理，使用线性插值和拉普拉斯等多种差值方法对缺失值进行填充，使用 3σ 法则剔除异常值；然后使用皮尔斯相关系数和折线趋势图查看气象变量之间的关系、污染物浓度之间的关系以及二者之间的相关性，并以此对气象条件进行划分，找出每类的特性。

问题 3：建立适用于多站点的二次空气质量预测模型。首先，污染物本身的扩散而言，其浓度变化在时间尺度上是缓慢的，如何建模这种主流时序信号变化是解决问题三的基础。其次，污染物本身以及其他气象因素是相互关联的，一次污染物与大气中已有组分或几种一次污染物之间经过一系列化学或光化学反应而生成的与一次污染物性质不同的新污染

物质，如何建模多种因素间的关联性是本章解决以上问题的关键。最后，不同站点间的信息具备可能的个性，也即局部自然气候、区域地形和人类活动及其对气象数据的影响也会对站点间的污染物预测造成干扰，也要充分考虑站点间客观存在的差异对预报的影响。

问题 4：建立区域协同多站点的二次空气质量预测模型。动态空间相关性，监测站点网络中传感器之间污染物浓度的相关性随时间显着变化。如何动态选择相关传感器的数据来预测目标传感器的长期污染状况是一个具有挑战性的问题。非线性时间相关性，传感器处的污染物浓度可能会剧烈而突然地波动，从而影响不同时间步长之间的相关性。随着时间的推移，如何对非线性时间相关性进行自适应建模仍然是一个挑战。对错误传播的敏感性，从长远来看，每个时间步中的小错误可能会在对未来进行更远的预测时放大。这种误差传播使得对遥远未来的预测极具挑战性。

二. 模型假设及符号说明

2.1 模型假设

建模过程中，为了简化问题排除无关因素的干扰，进行了以下假设：

（1） 假设给定的监测点气象数据和污染物浓度数据真实有效；

（2） 假设大气污染不会在短期内出现剧烈变化

（3） 假设现有的气象数据和历史实测数据足以进行有效的预测

（4） 假设监测站点的污染物浓度受局部人文自然条件影响

（5） 假设监测站点的位置真实性可靠，地理位置精确

2.2 符号说明

本文使用的部分符号及其意义如表 2-1 所示。

表 2-1 符号说明

符号	符号说明
Wxi	自变量权重
DL	数据权重
e	误差率
α	分类器权重
pk	第 k 个类的概率
Gini	基尼指数
fk	第 k 颗树的函数
l(yi,yi^)	损失函数
Ω(fk)	正则项函数
∂	一阶导数
ɤ	函数权重
ƛ	函数权重
Obj*	模型目标函数
V	监测站点
E	监测站点之间的直线连接
A	监测站点见的连接矩阵
X	输入数据
P	输入时间序列长度

Q	输出时间序列长度
L	PGAN 网络层数
f	激活函数
K	多头注意力的个数
Wi	线性层参数
bi	线性层偏置

三. 问题一：计算实测 AQI 和首要污染物

3.1 数据概览

3.1.1 数据基本信息

本次建模所使用的数据主要包括三个部分，分别是：监测点 A 空气质量预报基础数据、监测点 B、C 空气质量预报基础数据以及监测点 A1、A2、A3 空气质量预报基础数据。这三部分数据都是以站点形式进行组织，其基础统计情况如下表 3.1 所示。

表 3-1 6 个站点的数据统计信息

	一次预报数据	实测数据
数据采集时间	A1、A2、A3 站点：2020/7/23——2021/7/13 A、B、C 站点：2020/7/23——2021/7/15	2019/4/16——2021/7/13
样本个数	A、A1、A2、A3 站点：25344 B、C 站点：25416	820 左右 （每个站点略有差异）
变量类型及个数	气象一次预报数据：15 个	逐小时污染物浓度数据：6 个 逐小时气象实测数据：5 个 逐日污染物浓度数据：6 个

其中，气象一次预报数据主要包括：近地 2 米温度（℃）、地表温度（K）、比湿（kg/kg）、湿度（%）、近地 10 米风速（m/s）、近地 10 米风向（°）、雨量（mm）、云量、边界层高度（m）、大气压（Kpa）、感热通量（W/m²）、潜热通量（W/m²）、长波辐射（W/m²）和地面太阳能辐射（W/m²）等。污染物浓度数据主要包括：SO2、NO2PM10、PM2.5、O3 和 CO 等的小时平均浓度(mg/m³)数据和逐日检测数据，如图 3-1 所示。

此外，由于样本变量较多，这里分别选择气象一次预报数据、逐小时污染物浓度数据以及逐日污染物浓度数据中的一部分，查看其数据趋势，如图 3-2、3-3 及 3-4 所示，从图中不难发现：

① 从图 3-2 单月一次气象预报数据中可以看到，预报数据具有非常明显的周期性规律

② 图 3-3 单月污染物浓度数据则没有这么明显的规律性，但是也符合时序数据逐渐变化的特征。更重要的是，所有污染物浓度的变化趋势基本相同，例如：图中 2020-9-1 到 2020-9-5 之间，SO2 和其他物种污染物浓度变化趋势基本一致，这说明，这些污染物变量之间有一定的相关影响。

③ 最后，在图 3-4 逐日污染物浓度图像中可以看到，有很明显的异常值和缺失值的情况出现，同时，与图 3-3 相同的是，所有污染物浓度变化一致。即短期的逐小时污染物浓度变化和长期的逐日污染物浓度变化都反映出，污染物变量彼此之间具有很大的相关性，变化趋势基本一致。

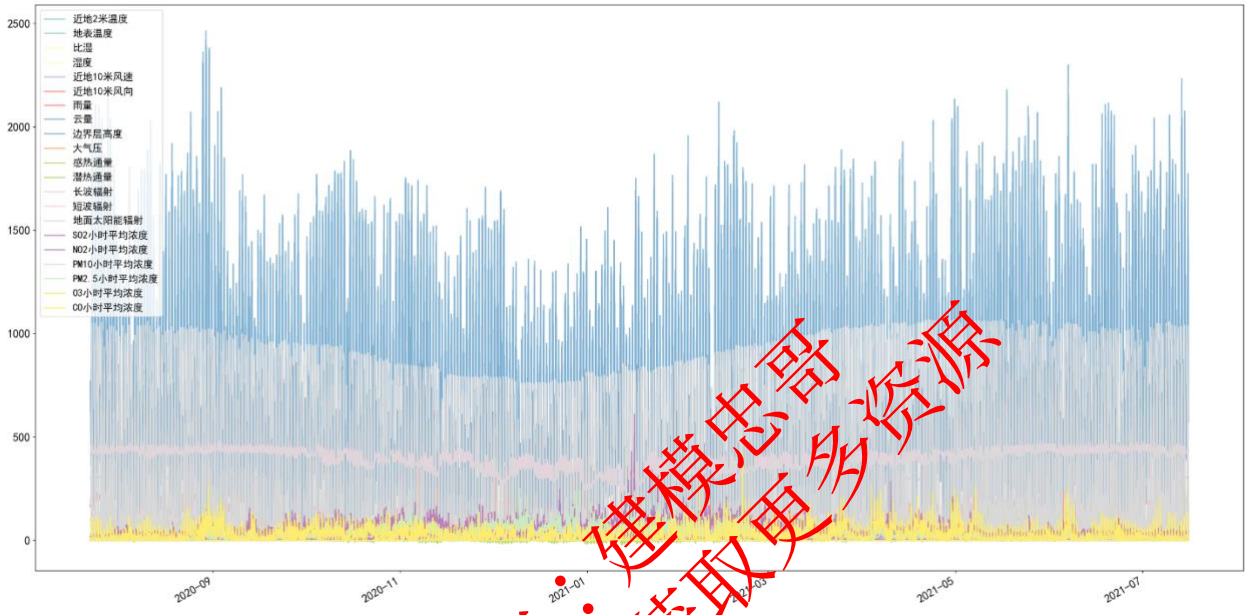

图 3-1 数据整体图

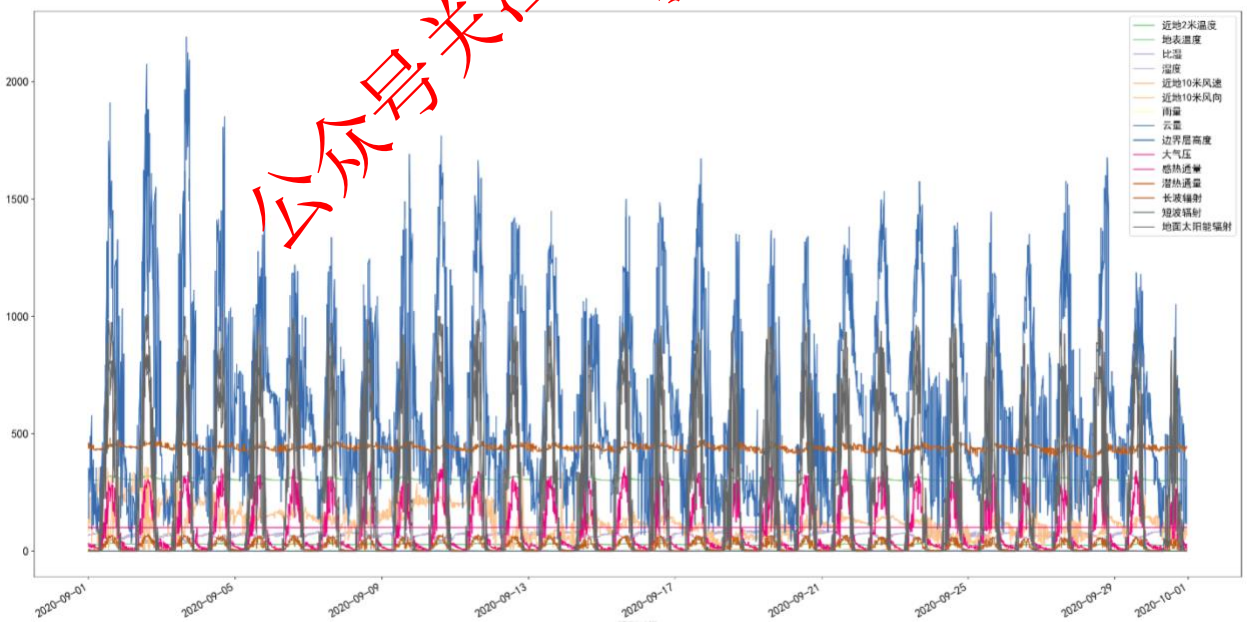

图 3-2 单月一次气象预报数据

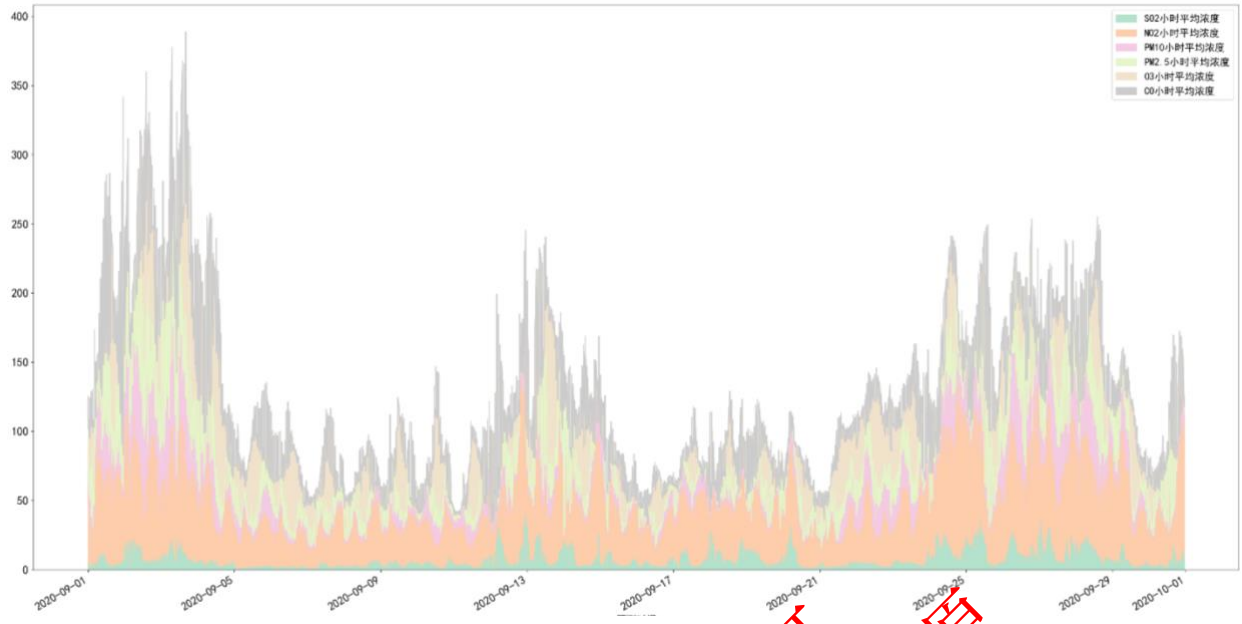

图 3-3 单月逐小时污染物浓度数据

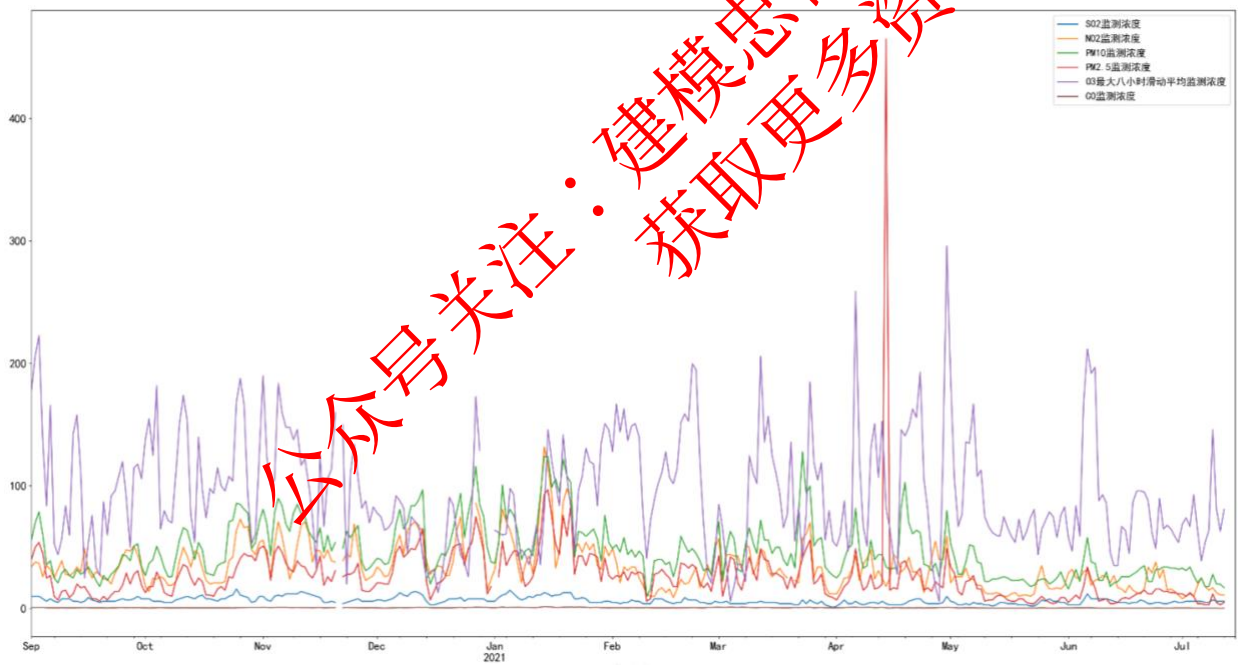

图 3-4 逐日污染物浓度数据

3.1.2 数据缺失查看

在使用数据之前，需要确认数据的完整性，查看是否存在缺失值，同时在查看数据基本信息时，已经发现存在有缺失值。故这里以附件 1 监测点 A 空气质量预报基础数据中的监测点A逐日污染物浓度实测数据工作表为例，查看缺失值，其他数据也以类似方式进行。

直接调用 Python 的分析缺失值的库 missingno 进行查看，如图 3-5 所示，从左到右，可以看到，PM10 这一污染物浓度缺失的数据最多，大约缺失了 1%左右，其余缺失情况并不十分严重。

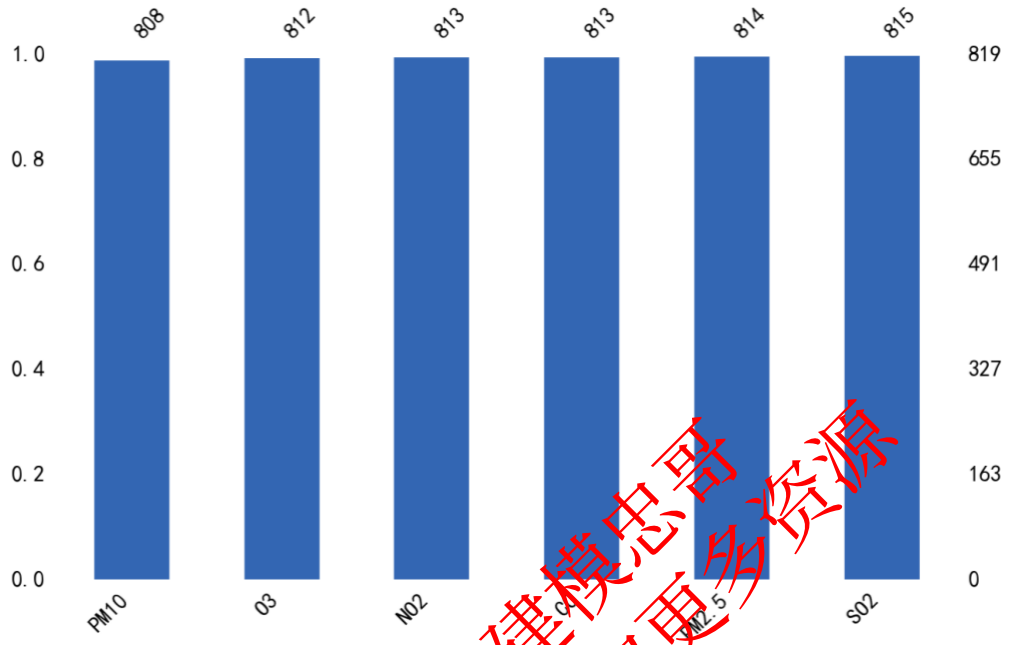

图 3-5 站点 A 逐日污染物浓度数据缺失情况

另外，可以看到，NO2 和 CO 的缺失数量相同，但是所处的缺失样本可能不同。因此，进行污染物缺失情况关联性分析，如图 3-6 的热力图所示，不难看出，SO2 和 CO 这一数据缺失比较分散，其他数据缺失相对较为集中，结合对应的站点 A 逐小时污染物浓度数据可知，逐日数据并不是逐小时数据的求和平均得到，例如：2020/12/26 的逐小时数据，24 个值中缺失了 5 个，但是 2020/12/26 的逐日数据是一个缺失值。

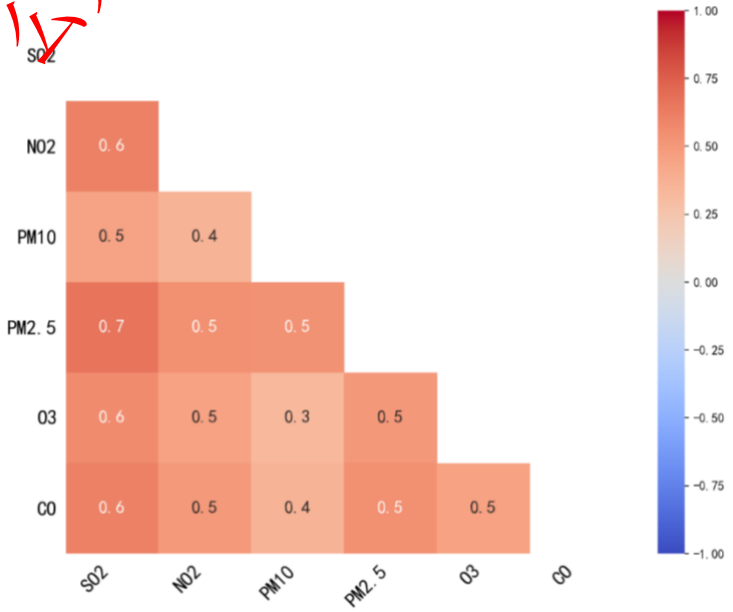

图 3-5 站点 A 逐日污染物浓度数据缺失关联

3.1.3 数据异常分析

首先使用 3σ 原则对数据进行异常值分析。3σ 原则：又称为拉依达准则，具体来说，就是先假设一组检测数据只含有随机误差，对原始数据进行计算处理得到标准差，然后按一定的概率确定一个区间，认为误差超过这个区间的就属于异常值。由于本实验变量个数较多，这里仅以附件 1 中逐日污染物浓度为例，展示其使用 3σ 原则分析后的结果。如表 3-2 所示。其中变量的四分位数的值从左到右依次表示下四分位数，中位数和上四分位数。

表 3-2 污染物浓度数据异常值

变量名	3σ 原则得到的异常值	四分位数
SO2	[17,20,17,17,17,18,19,17]	[5,6,9]
NO2	[96,111,113,97,98,115,100,132,117,92, 98]	[20,29,41]
PM10	[118,115,116,120,131,125,143,116,124,124,132,128]	[27,38,56]
PM2.5	[93,97,465]	[11,21,32.75]
O3	[255, 259, 262, 259, 296]	[61,87,128.25]
CO	[1.4, 1.4, 1.5, 1.4]	[0.6,0.7,0.8]

除了使用 3σ 原则判断异常值，还可以使用箱型图（以四分位数作为依据）来判断异常值，如图 3-6 所示，可以发现，PM10 的异常值最为密集，同时，使用超过四分位数上界得到的异常值与采用 3σ 原则得到的异常值基本一致。

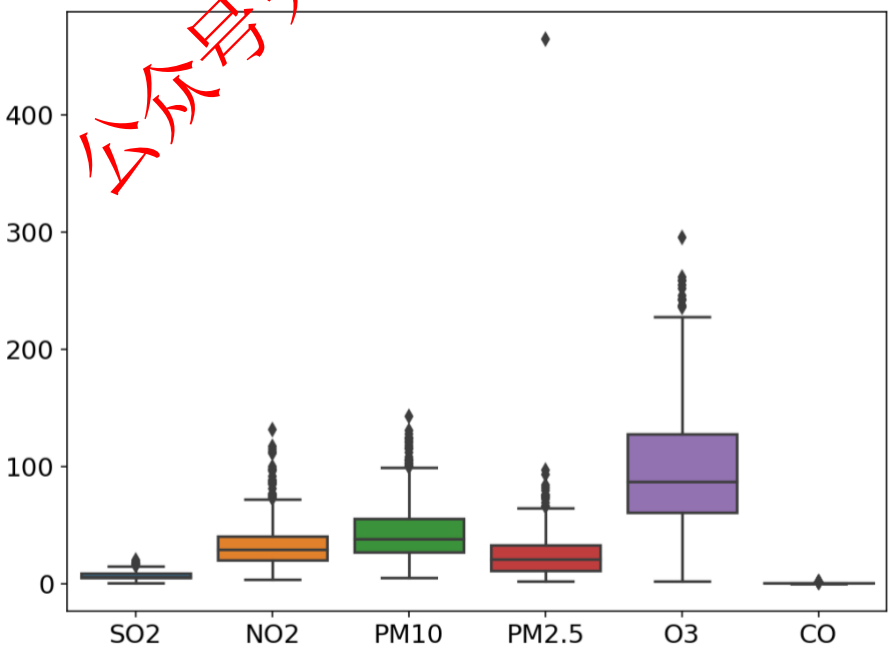

图 3-6 站点 A 逐日污染物浓度数据箱型图

为了更直观的看到异常值的分布情况，这里以 NO2 为例，绘制了其异常点的位置，如图 3-7 所示，其中 x 轴表示样本标号，y 轴表示 NO2 污染物浓度。

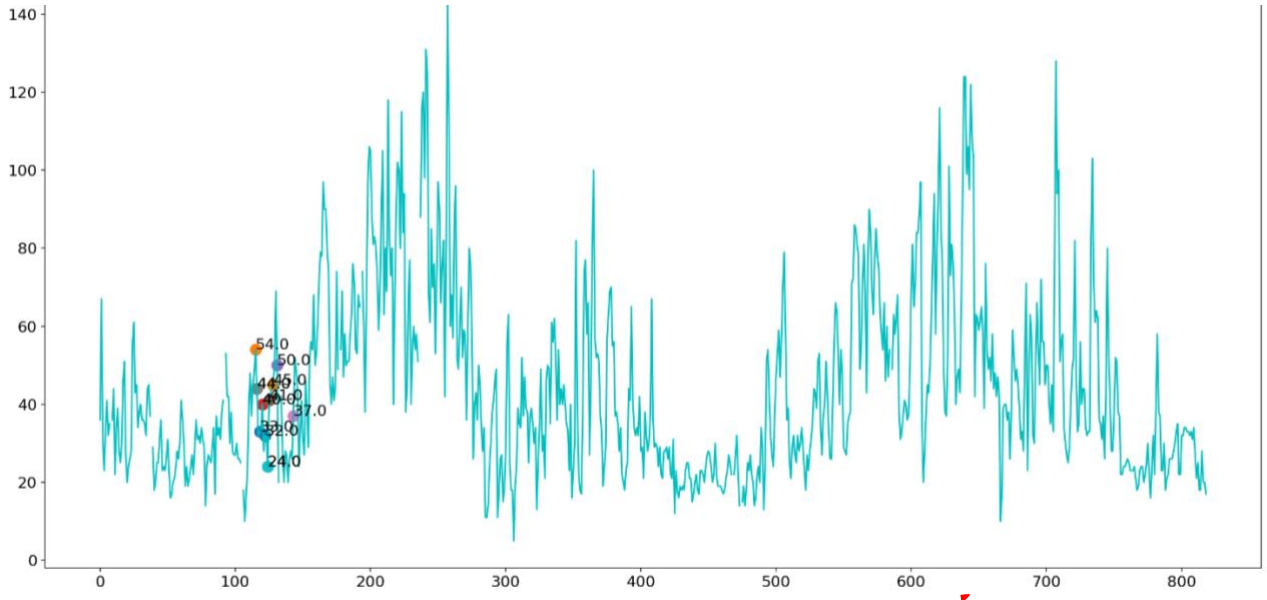

图 3-7 站点 A 浓度物 NO2 异常值分布图

3.2 实测 AQI 和首要污染物计算

3.2.1 计算方法说明

空气质量指数（AQI）是从多种污染物浓度的空气质量分指数（IAQI）中取最大值得到的，其可以用来判别空气质量等级，按照数值的不同，可以划分为：

① AQI 在[0,50]时，空气质量等级为优；

② AQI 在[51,100]时，空气质量等级为良；

③ AQI 在[101,150]时，空气质量等级为轻度污染；

④ AQI 在[151,200]时，空气质量等级为中度污染；

⑤ AQI 在[201,300]时，空气质量等级为重度污染；

⑥ AQI 在[301,+∞)时，空气质量等级为严重污染；

其中，各项污染物的空气质量分指数（IAQI）的计算按照如下公式进行：

$$IAQI_P = \frac{IAQI_{Hi} - IAQI_{Lo}}{BP_{Hi} - BP_{Lo}} \cdot (C_P - BP_{Lo}) + IAQI_{Lo}$$

式中各符号含义如下：

$IAQI_P$	污染物P的空气质量分指数，**结果进位取整数**；
C_P	污染物P的质量浓度值；
BP_{Hi}，BP_{Lo}	与C_P相近的污染物浓度限值的高位值与低位值；
$IAQI_{Hi}$, $IAQI_{Lo}$	与BP_{Hi},BP_{Lo}对应的空气质量分指数。

有一点需要额外注意的是，臭氧（O3）数据给出的是逐小时和逐日数据，但是需要进一步计算其最大 8 小时滑动平均指标，即 8 时到 24 时这一时间段内所有的 8 小时滑动平均值的最大值，作为臭氧这一污染物的 IAQI 计算依据。使用以下公式计算：

$$C_{O_3} = \max_{t=8,9,\ldots,24}\{\frac{1}{8}\sum_{i=t-7}^{t} c_i\}$$

其中c_i为臭氧在某日$t-1$时至t时的平均污染物浓度。

污染物浓度限值及空气质量分指数级别可以查询表 3-3 得到，需要注意的是，当臭氧（O_3）的最大 8 小时滑动平均浓度值高于 800 $\mu g/m^3$，其他其余污染物浓度高于 IAQI=500 对应的限制时，不再进行其 IAQI 的计算。

此外，关于首要污染物，其基于 AQI 进行判断，其判断规则如下：

① $AQI \leq 50$时，没有首要污染物（空气质量评级为优）

② $AQI > 50$时，IAQI 值最大的为首要污染物。若最大值有两项或两项以上，则对应的污染物同为首要污染物。

表 3-3 污染物浓度限值及空气质量分指数级别

序号	指数或污染物项目	空气质量分指数及对应污染物浓度限值								单位
0	空气质量分指数（IAQI）	0	50	100	150	200	300	400	500	-
1	一氧化碳（CO）24 小时平均	0	2	4	14	24	36	48	60	mg / m³
2	二氧化硫（SO_2）24 小时平均	0	50	150	475	800	1600	2100	2620	
3	二氧化氮（NO_2）24 小时平均	0	40	80	180	280	565	750	940	
4	臭氧（O_3）最大 8 小时滑动平均	0	100	160	215	265	800	-	-	μg / m³
5	粒径小于等于 10μm颗粒物（PM_{10}）24 小时平均	0	50	150	250	350	420	500	600	
6	粒径小于等于 2.5μm颗粒物（$PM_{2.5}$）24 小时平均	0	35	75	115	150	250	350	500	

3.2.2 计算结果

首先计算 2020 年 8 月 25 日到 2020 年 8 月 28 日这四日的臭氧（O3）最大 8 小时滑动平均值，并与其他污染物浓度的日监测数据组合在一起，形成计算 AQI 的基础数据，计算结果表 3-4 所示：

表 3-4 计算三日 AQI 所需的污染物浓度数据

监测日期	SO2	NO2	PM10	PM2.5	O3	CO
2020-08-25	8	12	27	11	112.25	0.5
2020-08-26	7	16	24	10	92.125	0.5

2020-08-27	7	31	37	23	169.125	0.6
2020-08-28	8	30	47	33	200.75	0.7

根据以上污染物浓度数据计算各污染物的空气质量分指数 IAQI，计算结果如表 3-5 所示，可以看到，其中 O3 的 IAQI 分指数在四天内均是最高的。

表 3-5 各污染物的空气质量分指数

监测日期	$IAQI_{SO2}$	$IAQI_{NO2}$	$IAQI_{PM10}$	$IAQI_{PM2.5}$	$IAQI_{O3}$	$IAQI_{CO}$
2020-08-25	8	15	27	16	61	13
2020-08-26	7	20	24	15	47	13
2020-08-27	7	39	37	33	109	15
2020-08-28	8	38	47	48	138	18

在各污染物的空气质量分指数 IAQI 的基础上，得到 AQI，并参照规定和 AQI 等级划分得到最终的计算结果，并没有出现污染指数特别大的情况，所以都可以进行常规的计算和分类，结果如表 3-6 所示：

表 3-6 各污染物的空气质量分指数

监测日期	地点	AQI 计算	
		AQI	首要污染物
2020/8/25	监测点 A	良	O3(IAQI=61)
2020/8/26	监测点 A	优	无首要污染物 (IAQI=47)
2020/8/27	监测点 A	轻度污染	O3(IAQI=109)
2020/8/28	监测点 A	轻度污染	O3(IAQI=138)

从上述结果可以看出，对于监测点 A 来说，连续四日的首要污染物都是 O3，而 O3 是六项污染物中唯一的二次污染物，这说明监测点 A 附近的环境和产业可能有助于 O3 的形成和传播。为此，对监测点 A 的逐日污染物浓度进行分析，绘制不同周期的趋势变化图（暂时不考虑对缺失值进行插值，插值和异常值处理在问题 2 中处理），如下图 3-8 所示：

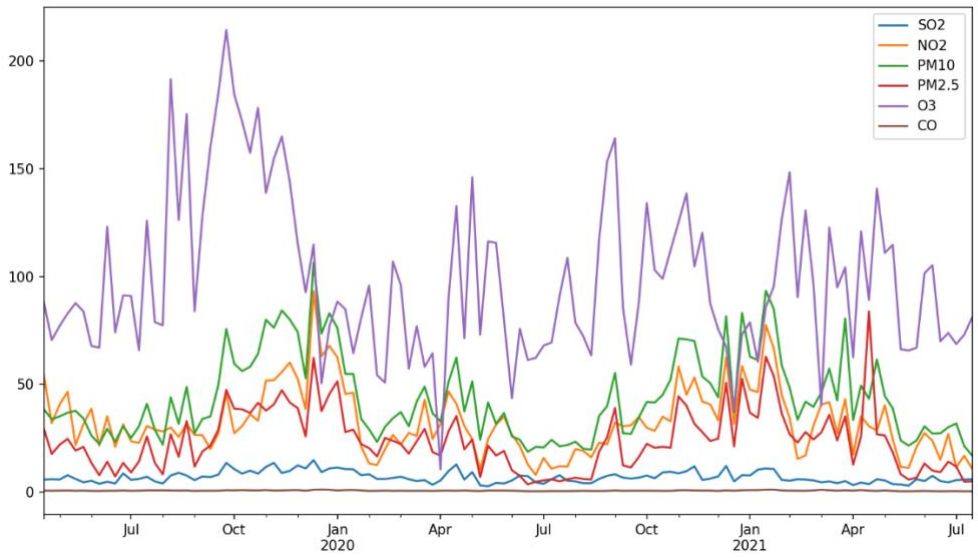

图 3-8 污染物浓度周平均数据

从上图可以看出，对于各污染物的周平均数据来说，O3 依然整体上都是浓度最高的污染物，此外，第二高的是 PM10，最低的是 CO 和 SO2。为了降低偶发误差的干扰，可以尝试查看污染物浓度月平均折线趋势图，如图 3-9 所示。

不难发现，在周平均数据中所呈现的污染物浓度在月平均数据中显得差距更为明显，O3 在 2019 年 4 月到 2021 年 7 月这段时间内，污染物浓度一直是最高的，基本都是首要污染物，这也验证了我们上面得到的 2020 年 8 月 25 日到 2020 年 8 月 28 日之间首要污染物都是 O3 的计算结果。

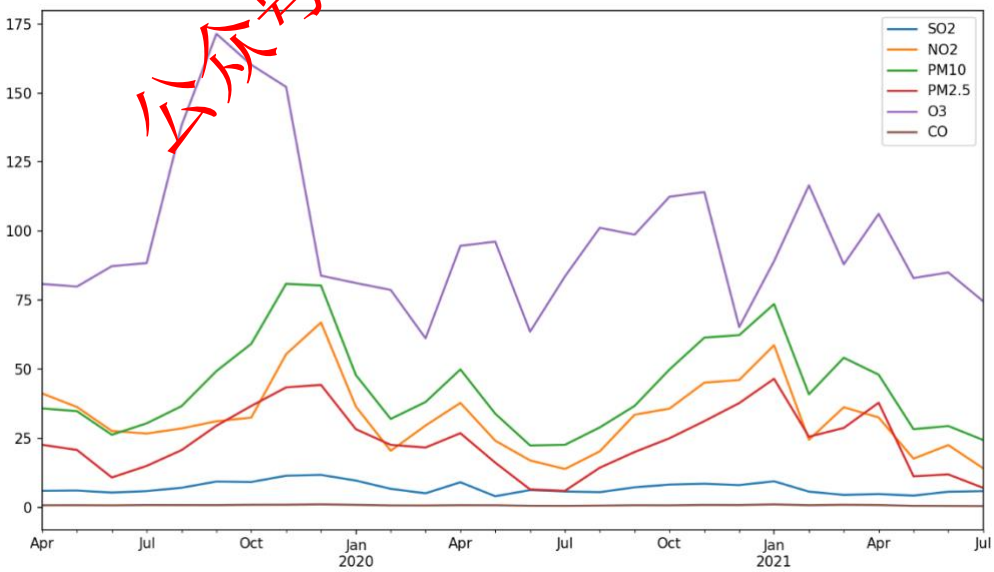

图 3-9 污染物浓度月平均数据

四. 问题二：分析气象条件对污染物影响及其特征

4.1 数据预处理

4.1.1 研究方法

由 3.1 节的数据缺失查看和数据异常值分析可知，数据存在一定的质量问题为此，需要对原始数据进行处理，主要包括缺失值处理，异常值处理及样本确定三个步骤，这三个步骤常采取的方法总结如下。

1）缺失值处理的方法

个案剔除法——作为最直接也是最简单的缺失数据处理方法，是很多统计软件（如 SPAA 和 SAS）默认的缺失值处理方法。在这种方法中，如果任何一个变量含有缺失值，就把对应的样本直接剔除。其优点在于：当缺失值比例较小时（根据数据量的大小，比例在5%~20%之间），这一方法十分有效。其缺点也非常明显：由于其以减少样本量来获取信息的完备，丢弃了大量不完整样本数据中的信息，在样本量较小的情况下，删除少量对象就足以严重影响到数据的客观性和结果的正确性；因此，当缺失数据所占比例较大，特别是当缺失数据非随机分布时，这种方式可能会导致数据发生偏离，从而得出错误的结论。

均值替换法——当存在缺失的数据很重要又很多时，就不宜采取个案剔除法。均值替换法作为个案剔除法的一种解决方法，将变量的属性分为数值型和非数值型来分别处理。对于数据型，使用该变量在其他样本的取值的平均值来填充；对于非数值型，则根据统计学中的众数原理，使用该变量在其他样本中取值次数最多的值来填充。其优点在于：简便快速。其缺点在于：容易产生有偏估计，此外，由于是建立在完全随机缺失(MCAR)的假设之上，会造成变量的方差和标准差变小。

回归替换法——首先需要选择若干个预测缺失值的自变量，然后建立回归方程估计缺失值，即使用缺失数据的条件期望值对缺失值进行填充。与前几种插补方法相比，该方法更加充分的利用了数据信息。其缺点在于：虽然是一个无偏估计，但是却容易忽略随机误差，低估标准差和其他未知性质的测量值，同时这一问题会随着缺失信息的增多而变得更加严重；此外，这一方法的前提是假设缺失值所在变量和其他变量存在线性关系，而很多时候这种关系是不存在的。

2) 异常值处理的方法

剔除异常值——有两种常见的剔除异常数的方法：拉依达准则和格拉布斯准则。对于前者，其假设一组检测数据只含有随机误差，对其进行计算处理得到标准偏差，按一定概率确定一个区间，认为凡超过这个区间的误差，就不属于随机误差而是粗大误差，含有该误差的数据应予以剔除。这种判别处理原理及方法仅局限于对正态或近似正态分布的样本数据处理，它是以测量次数充分大为前提的，当测量次数少的情形用准则剔除粗大误差是

不够可靠的。因此，在测量次数较少的情况下，最好不要选用该准则。对于后者，格拉布斯准则是以正态分布为前提的，理论上较严谨，使用也方便。

平滑处理——数据平滑主要是为了去除数据中的噪声，常见的有基于最小二乘法的数据平滑算法及指数平滑算法等

标准化（归一化）——常见的数据规范方法有：最小-最大规范化、Z-score 规范化及小数定标规范化等。

4.1.2 数据处理

1) 缺失值处理

根据 3.1.2 部分的数据缺失情况可知，PM10 这一污染物的逐日污染物浓度数据缺失情况最严重，故以 PM10 为例，选用了包括线性插值，时间插值，填充插值，最近邻插值等的多种插值方法，根据最后的拟合效果选择了最近邻插值作为处理缺失值的插值方法，插值结果和原数据对比如图 4-1、4-2 和 4-3 所示。

观察可以得到以下结论，

1. PM10 的缺失值大概有①2019-05 右侧一处，②和③2019-07 右侧两处，以及④2019-09 附近一处，⑤2019-11 到 2020-01 中间一处。
2. 对于填充插值和线性插值来说：在①、②、③处，线性插值表现更为平滑；而在④和⑤处，由于数据差距过大，二者表现都不是非常理想。
3. 对于时间插值和最近邻插值来说：时间差值整体比最近邻插值更为平滑，但是最近邻插值更能反映出与原始数据相近的数据抖动。

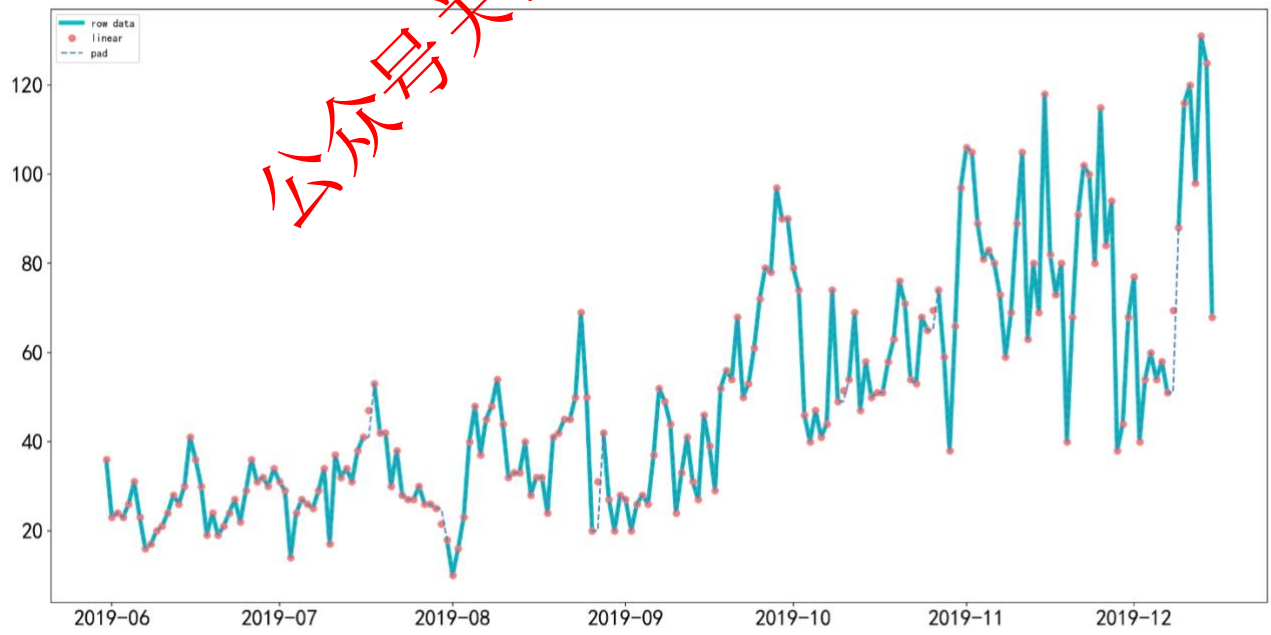

图 4-1 插值结果和原数据对比 1a

在图 4-1 中，蓝色粗线是插值前的数据，橙色点线是线性插值的结果，深蓝色虚线是填充插值结果，可以看到几处明显的虚线，就是填充的缺失值部分。另外，关于线性插值和填充插值详细的对比图可以参考图 4-2。

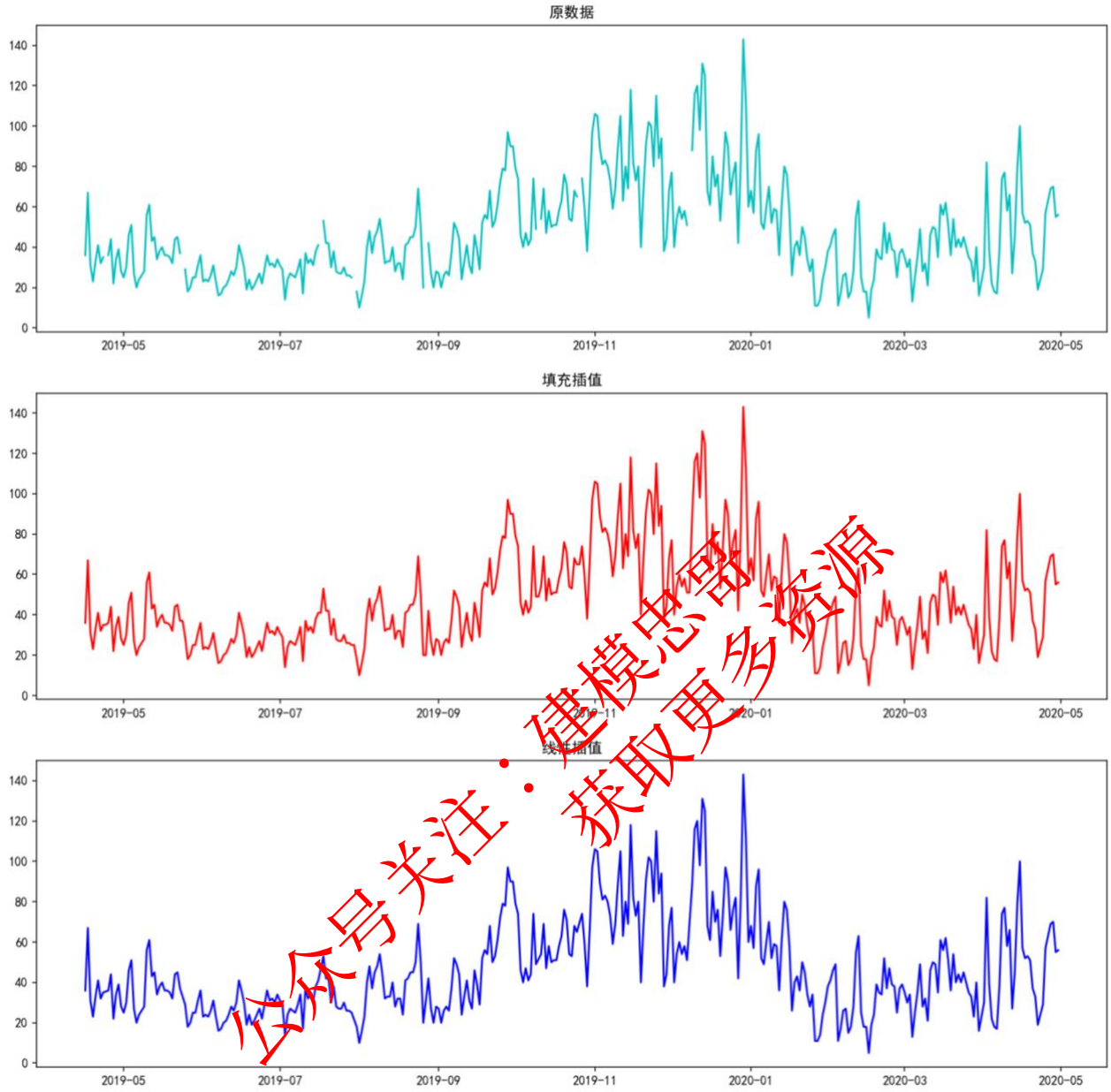

图 4-2 插值结果和原数据对比 1b

除了线性插值和填充插值，还使用了时间插值和最近邻插值，如图 4-3 所示，鉴于整体情况与 4-2 类似，这里就不再附上对应于 4-3 的类似 4-1 的图像了。

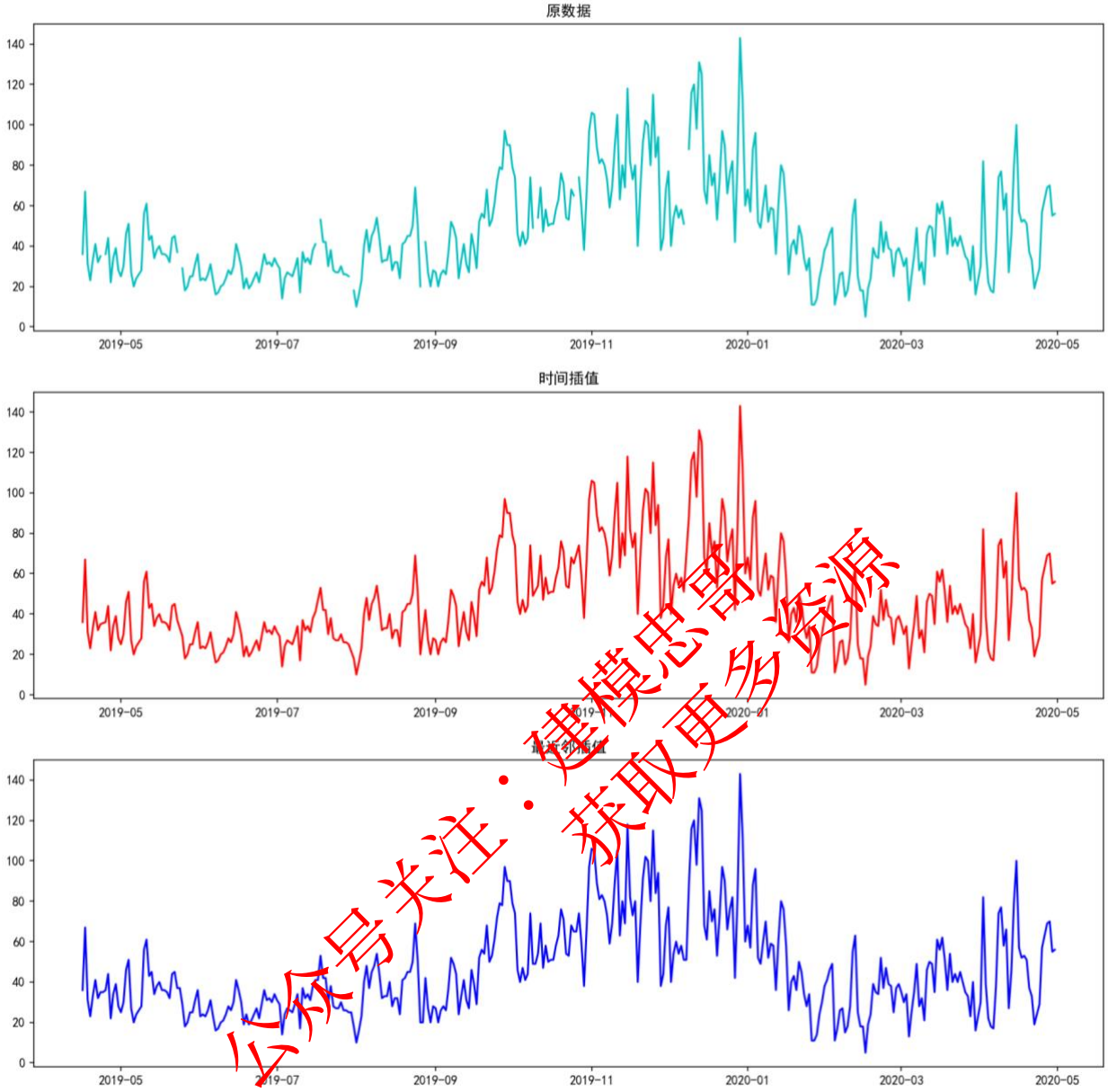

图 4-3 插值结果和原数据对比 2

2) 异常值处理

根据 3.1.3 部分的异常值分析，采取以下步骤进行处理：

1. 根据四分位数，常规操作是将大于四分位数的上界和小于四分位数的下界的数值进行删除，但是由于本次数据是时间序列数据，故没有采取删除操作，而是对异常值标记为 NaN，再使用插值方式进行填充。
2. 根据拉依达准则（3σ 准则）标记异常值。其大致过程如下：

3σ 准则：设对被测量变量进行等精度测量，得到 x_1，x_2，……，x_n，算出其算术平均值 x 及剩余误差 $v_i=x_i-x$（i=1,2,...,n），并按贝塞尔公式算出标准误差 σ，若某个测量值 x_b 的剩余误差 v_b（1<=b<=n），满足$|v_b|=|x_b-x|>3\sigma$，则认为 x_b 是含有粗大误差值的坏值，进行标记。贝塞尔公式如下：

$$\sigma=[\frac{1}{n-1}\sum_{i=1}^{n}v_i^2]^{1/2}=\{[\sum_{i=1}^{n}x_i{}^2-(\sum_{i=1}^{n}x_i)^2/n]/(n-1)\}^{1/2}$$

3）样本确定

在进行过异常值和缺失值处理之后，就得到了可以直接使用的数据。附件 1 中，最终可以使用的样本数据有 819 条。

4.2 气象条件分类

4.2.1 气象条件影响因素分析

已知在污染物排放情况不变的条件下，某一地区的气象条件有利于污染物扩散或沉降时，该地区的 AQI 会下降，反之会上升。因此接下来会根据气象条件对污染物浓度的影响，对气象条件进行分析。以站点 A 的数据为依据，主要分析的内容包括：气象条件彼此之间的相关性，污染物浓度彼此之间的相关性以及气象条件和污染物之间的相关性。主要采取的相关性分析方法包括 pearson 相关系数、Spearman 相关系数以及 Kendall 相关系数。

其中，两个变量之间的皮尔逊相关系数定义为两个变量之间的协方差和标准差的商，其计算方式如下所示：

$$\rho_{X,Y}=\frac{\operatorname{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}$$

Kendall(肯德尔)系数的定义为：n个同类的统计对象按特定属性排序，其他属性通常是乱序的。同序对（concordant pairs）和异序对（discordant pairs）之差与总对数（$n*(n-1)/2$)的比值定义为 Kendall(肯德尔)系数。

斯皮尔曼相关系数被定义成等级变量之间的皮尔逊相关系数。对于样本容量为n的样本，n个原始数据被转换成等级数据，相关系数ρ的公式表示为：

$$\rho=\frac{\sum_i(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i(x_i-\bar{x})^2\sum_i(y_i-\bar{y})^2}}$$

pearson 相关系数通常用于分析定量数据，当数据满足正态性时可用 Pearson 相关系数查看变量间关系情况。与之对应的是，当数据为定量数据，且不服从正态性则使用 Spearman 相关系数。Kendall 相关系数通常用于评分数据一致性水平研究，比如评委打分，数据排名等。所以使用站点 A 的一次气象预报数据为基础时，使用皮尔逊相关系数分析气象条件和污染物变量之间的相关性关系，如图 4-4 所示。同时以站点 A 的气象实测数据为基础，也使用皮尔逊相关系数分析气象实测数据和污染物变量之间的相关性关系，如图 4-5 所示。

从图 4-4 中，不难得出以下结论：

1. 气象条件之间存在一定的相关性关系，例如：长波辐射和比湿的正相关程度高达 86%。同时，短波辐射、地面太阳能辐射与感通热量、潜通热量的正相关程度分别高达 96%、97%，值得注意的是，短波辐射指地面吸收的太阳能短波辐射的地表

能通量，地面太阳能指地面吸收的太阳能所有波长辐射的地表能通量，地面吸收的太阳辐射能会以感热形式及潜热形式释放至大气中，会以能通量形式表示这部分感热及潜热。所以从物理角度理解，这四者之间的强正相关关系是成立的。

2. 污染物浓度之间也存在一定的相关性关系。例如：对于 SO2 小时平均浓度来说，其于 NO2、PM10 以及 PM2.5 的小时浓度平均数据也呈现比较明显的正相关关系。此外，污染物整体都有一定的正相关性的关系，少数存在负相关或者无关的关系。
3. 气象一次预报数据和污染物浓度之间存在一定的相关性关系。例如：大气压这一气象条件与其他污染物浓度都存在比较明显的正相关关系。另外，温度和 O3 也具有一定的正相关关系，对这二者进行单独分析，如图 4-6 所示，可以看到，从 2020 年 10 月至 2021 年 5 月，地表温度和 O3 浓度变化趋势基本一致，二者有比较明显的正相关关系。

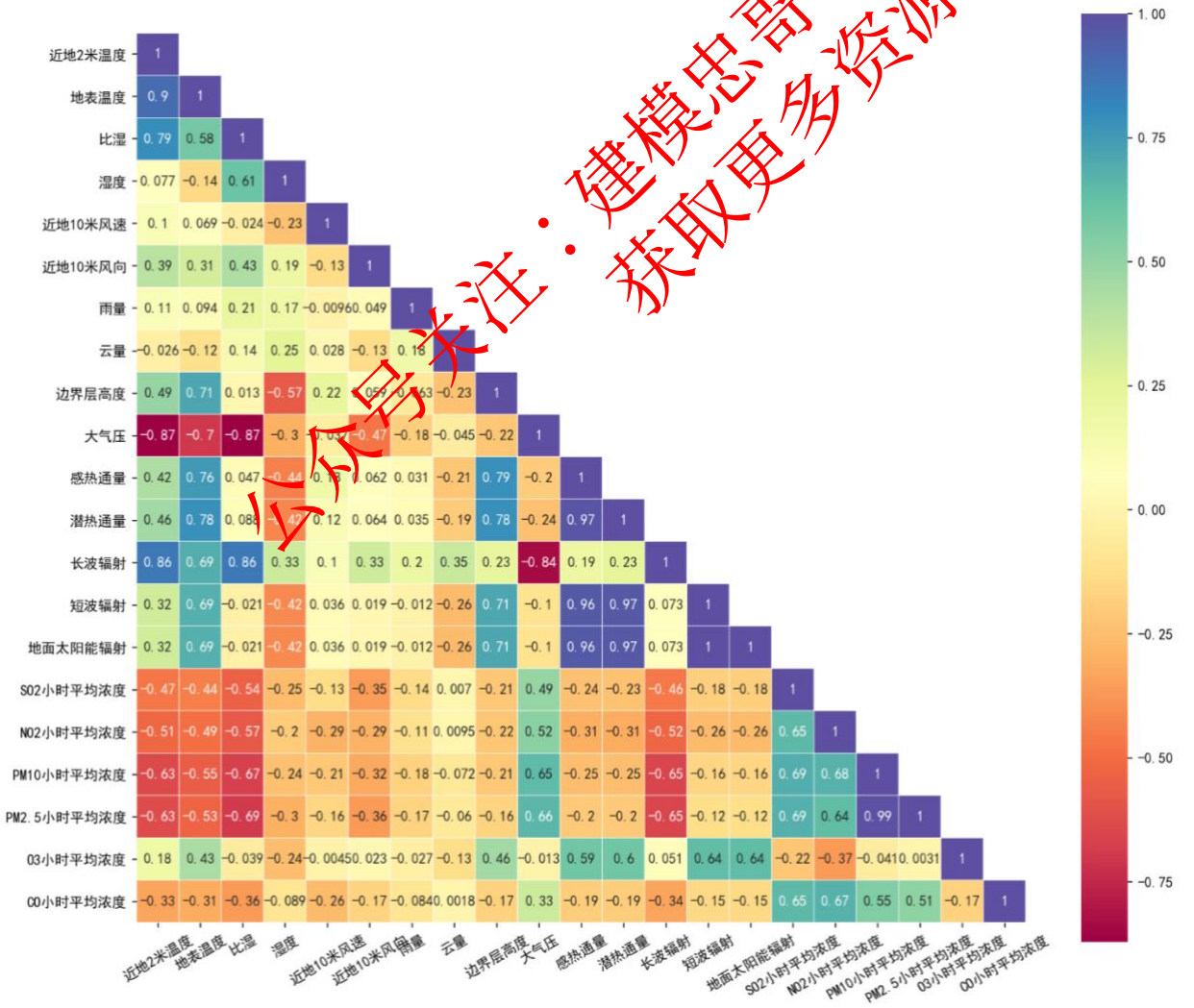

图 4-4 气象一次预报数据和污染物实测数据相关性

4.2.2 气象条件的分类和特征

通过相关性分析，可以明确污染物浓度和气象数据之间的相关性，这里进行进一步细致的分析，同时，为了获取真实的相关性，所以这里只分析实测气象数据与污染物之间的关联，主要分析单个污染物浓度与气象数据之间的关系。

1. 如图 4-5 所示，太阳辐射强弱对光化学反应有重要的影响，而温度升高对 O3 生成速率有显著影响。对于站点 A 来说，在一定范围内，温度越高，O3 污染物的浓度越高。同时，可以发现臭氧超标日和污染日主要出现在 7—10 月。臭氧浓度日变化呈单峰变化，06—08 时最低，最大值出现在午后 14—15 时。臭氧浓度变化和气象条件关系密切，低浓度臭氧大多出现在气温较低、相对湿度和风速较大的天气，臭氧浓度超标多出现在气温较高、相对湿度和风速较小的天气。
2. 如图 4-6 所示，温度、湿度、风速、风向这四种气象因素对大部分污染物浓度的影响趋势相同。例如，其对 SO2、NO2、PM10 以及 PM2.5 都是负相关，
3. 如图 4-7 所示，SO2 监测浓度整体趋势变化与湿度基本一致，但是具有一定的滞后性。
4. 如图 4-8 所示，NO2、PM10 和 PM2.5 这三种污染物的变化趋势基本一致，同时可以看到，PM2.5 的趋势和湿度的变化趋势非常相近，但是也具有一定的滞后性。

经过上述分析，总结如下：

① 将温度、湿度、气压、风向和风速这五个气象因素作为一类，作为空气质量监测中的基础监测要素。一方面因为这五种数据便于监控和获取，另一方面，其对污染物影响也非常关键。

② 除去上述五种气象因素外，其余气象因素作为一类，作为空气质量检测中的增益检测要素。一方面是因为其相对不容易获取，一般是通过模拟气象场得到的预测值；另一方面是因为这些气象因素彼此之间关联性较高，不容易进行分析。但是其依然对空气质量预测有一定的积极作用，故作为第二类进行保留。

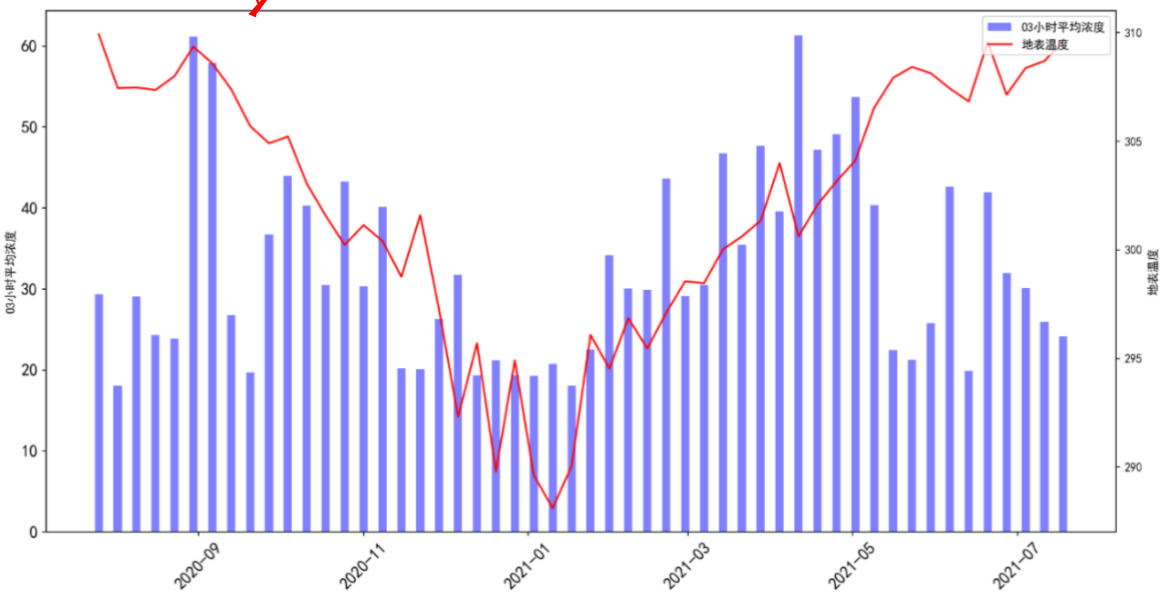

图 4-5 O3 与地表温度相关性分析图

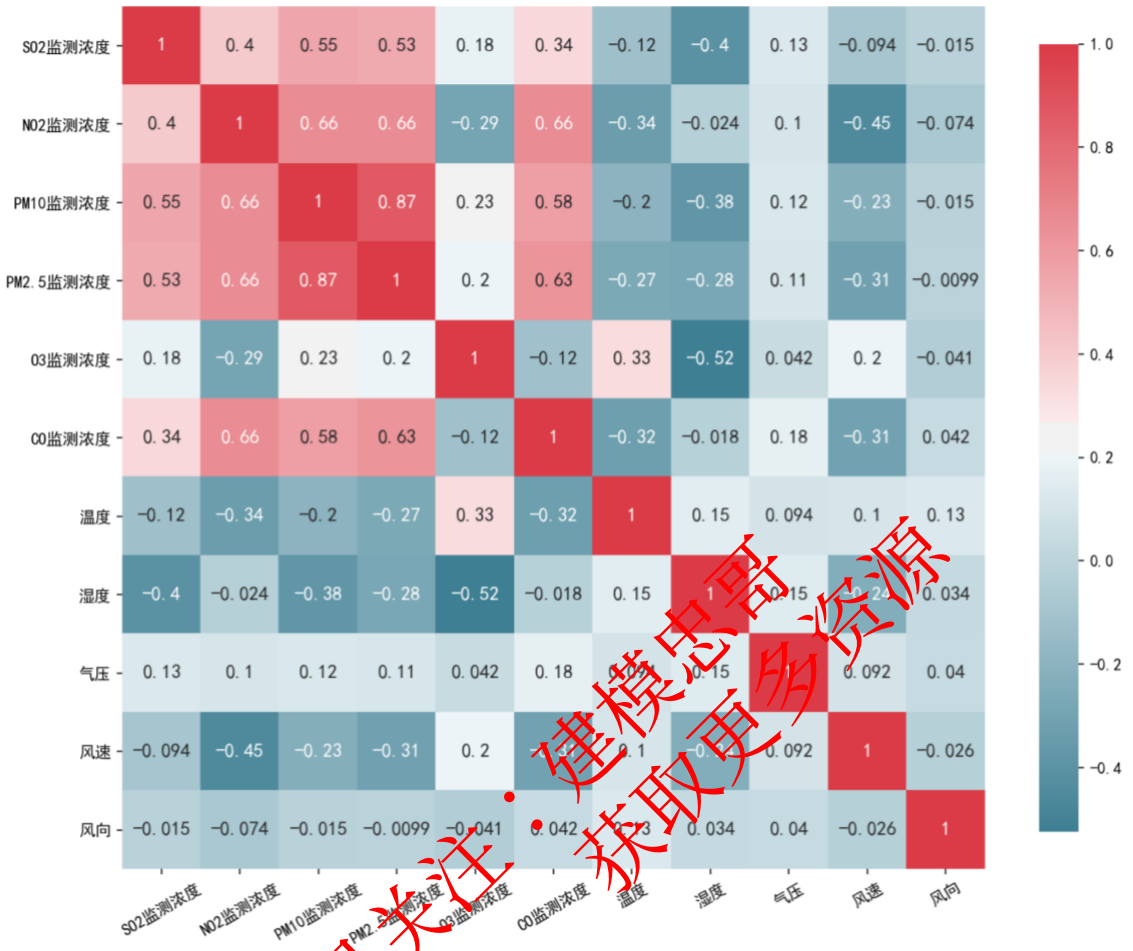

图 4-6 气象实测数据和污染物数据的相关性

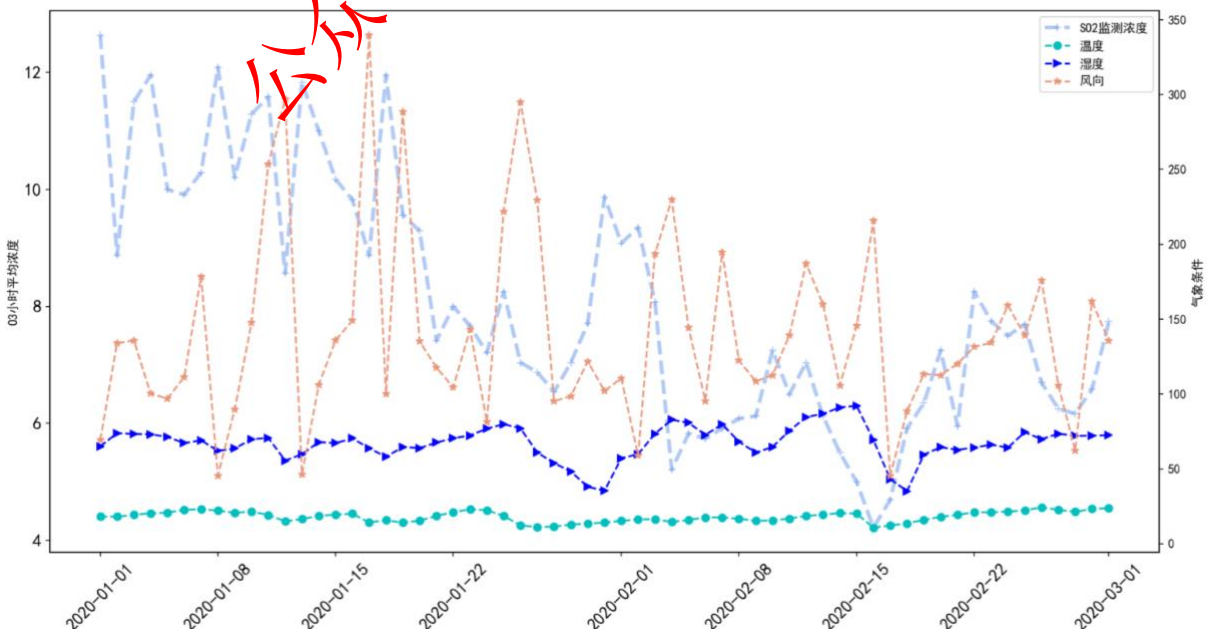

图 4-7 SO2 与温度、湿度及风向相关性分析

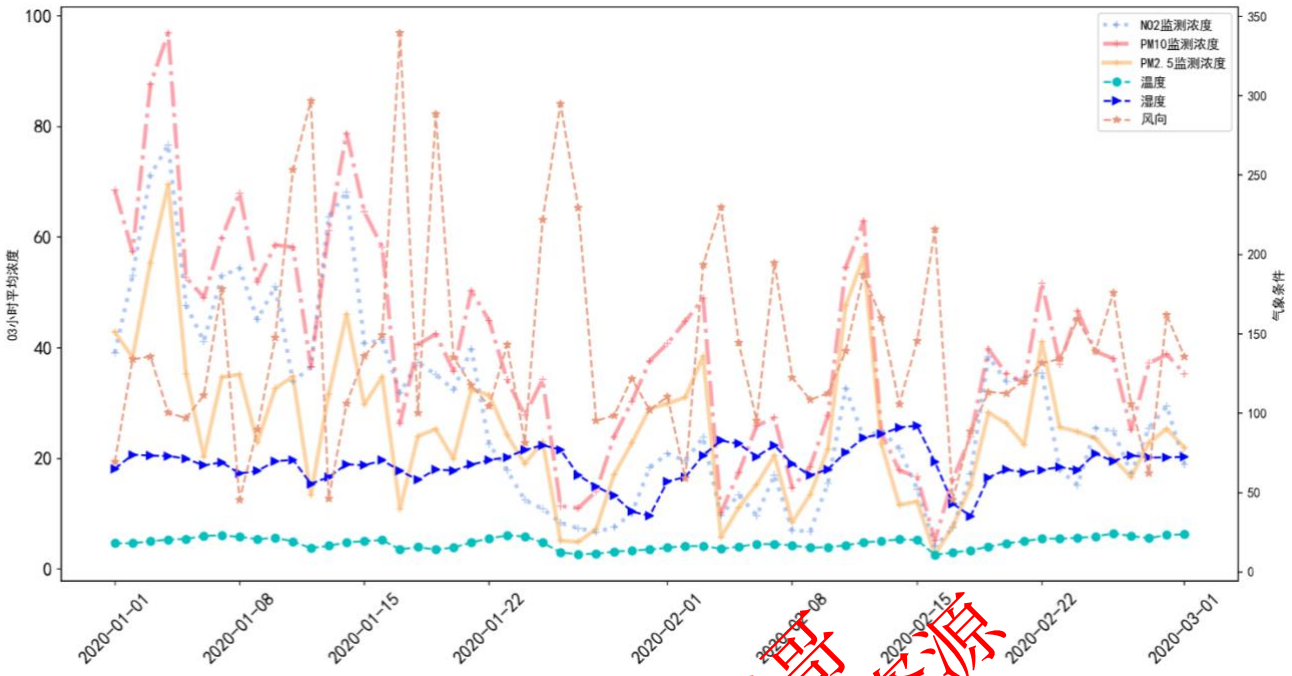

图 4-7 NO2、PM10 和 PM2.5 与温度、湿度及风向相关性分析

五．问题三：跨站点多元污染物浓度通用预报模型

5.1 问题分析

本文使用附件 1、2 中的数据，建立一个同时适用于 A、B、C 三个监测点（监测点两两间直线距离>100km，忽略相互影响）的二次预报数学模型，用来预测未来三天 6 种常规污染物单日浓度值，要求二次预报模型预测结果中 AQI 预报值的最大相对误差应尽量小，且首要污染物预测准确度尽量高。并使用该模型预测监测点 A、B、C 在 2021 年 7 月 13 日至 7 月 15 日 6 种常规污染物的单日浓度值，计算相应的 AQI 和首要污染物，将结果依照附录“污染物浓度及 AQI 预测结果表”的格式放在文中。

在依托生活常识和专业知识，本文将以上问题分解为三个层次来考虑：

首先，污染物本身的扩散而言，其浓度变化在时间尺度上是缓慢的，经常可以凭借当日或者近几日的空气污染状况对后几日的空气质量作估计，即污染物自身具备时间上一定的连续性。如何捕获这种时序关系是本章解决以上问题的基础。

其次，污染物本身以及其他气象因素是相互关联的。比如，一次污染物是指直接从污染源排到大气中的原始污染物质，如硫氧化物（SOx）、氮氧化物（NOx）等。二次污染物则是指由一次污染物与大气中已有组分或几种一次污染物之间经过一系列化学或光化学反应而生成的与一次污染物性质不同的新污染物质，如臭氧、硫酸盐、硝酸盐、有机颗粒物等。因此，如何建模多种因素间的关联性是本章解决以上问题的关键。

最后，在达成以上两种关系建模的前提下，单一站点的二次预报数学模型可以被初步建立。问题三要求构建 A/B/C 三个监测点的通用模型，一来不同站点间的信息具备潜在的共性，常规污染物的变化符合同一/或者类似的规律，即尽管站点间互相独立且互不影响，但污染物的变化切合普遍规律，或者某一区域的污染物变化呈现一定的规律特点；二来，不同站点间的信息具备可能的个性，也即局部自然气候、区域地形和人类活动及其对气象数据的影响也会对站点间的污染物预测造成干扰。综上，一个合理的二次预报数学模型既要考虑小站点间符合的“普遍规律”，也要充分考虑站点间客观存在的差异对预报的影响。

由此本章节在研究二次预报数学模型的同时，将分别围绕以上问题，对时序信号预测问题、多元时序信号预测问题和跨站点多元时序信号预测问题做深入探讨。

5.2 模型建立

5.2.1 时序信号预测问题建模

污染物的二次预报问题体现出明确的时序特性，即随时间的迁移，各污染物浓度值随时间变化，且问题三提供的数据也是按时间顺序记录。此外，一般将信号，即随时间变化的值得记录，当作是多种与时间相关成分的复合叠加：

$$x(t) = a_1 s_1(t) + a_2 s_2(t) + \cdots + a_n s_n(t)$$

其中，$a_1 a_2 \ldots a_n$为各分量的强度，$s_1(t) s_2(t) \ldots s_n(t)$为各随时间变化的信号分量。值得注意的是，往往观测到的为源信号x(t)，其随时间不断变化的机理并不明朗，也即分量的叠加往往是不可逆的，必须对时序信号进行进一步的猜测和假定，以明确分量与源信号、分量与分量间的关联性，以构建信号模型。

对于常见的时序信号分析问题一般有如下几种常见解决方案：

时序分析，这是关于“看”时间序列在一段时间内如何演变的分析方法，它包括时序的长度宽度、统计特征以及其他“可见的”特征；

频域分析，信号的很多特性难以使用时间变化来描述，但是可以用其幅度和其变化来表示，小波分析和傅里叶分解是频域分析的代表模型；

近邻分析，通过直接分析信号间的相似性和疏离程度，尽管没有对信号进行深入的分析建模，但是很直观的表现出信号间的相同和不同，通过这种基于信号相关性的分析，可以对信号间的一些共有/特有属性信息进行挖掘。

以上基本的分析方式共同构建了过去几十年间大多数的时序信号预测模型，包括但不限于线性回归模型，ARMA/ARIMA 等线性模型，时间序列分解模型，此外一些基于概率模型的技术如卡尔曼滤波、高斯混合模型也被应用到相关问题的建模中 [6]。

本章节使用深度学习技术，将二次预报问题建模为一个回归问题，即通过历史时序信号，在此为各污染物历史浓度变化及其它相关数据（一次预报数据），对时序信号变化过程进行建模，并以此为基础完成二次预报数学模型的构建。

本章使用一种特殊的循环神经网络（RNN）来建模时序信号特征 [7]。不同于深度技术中常用的全连接层对信号的所有输入都给予相同的关注，本章使用长短期记忆网络（LSTM）来构建时序信号的长依赖性特性，即某一时间点的信号变化与之前或者之后的若干点相关，而这种相关性是加权的，这也更加符合人类的一般认识 [8]。如图 3-1 中，信号点h_3与$x_0 x_1$的关联性更大，而不应该对x_3过度关心。

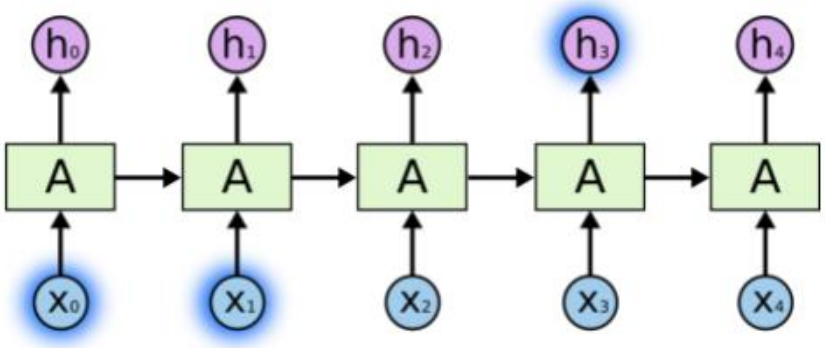

图 5-1 时序信号中的常依赖

需要注意的是，LSTM 将时序信号中信息的依赖建模为短期记忆和长期记忆的传递。如图 5-2 所示，为 LSTM 中信息传递过程。其中x_t为每一时刻的信号输入，C_{t-1}和h_{t-1}分别

为从上一时刻继承的长短记忆，模型在时刻 t 接受以上三个输入，对本时刻的隐变量状态进行估计，并继续以长短记忆的形式向下一时刻 t-1 传递记忆。为了模拟人类记忆对不同时刻信息的记忆程度，模型中提出一个遗忘门组件，来控制模型保留此刻以及上一时刻的信息分量幅值的程度。

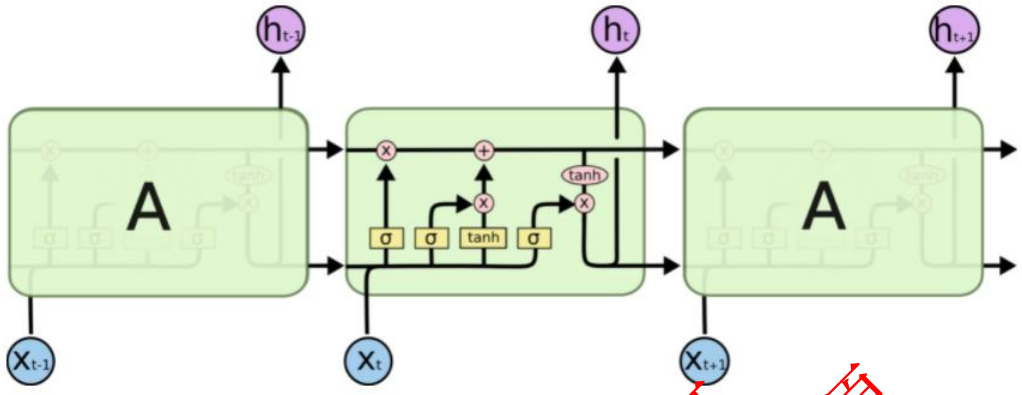

图 5-2 LSTM 内部结构

具体的，LSTM 模型接受上一时刻短记忆h_{t-1}，将其与当前时刻输入拼接进行一个简单的非线性层作为输入门：

$$\mathrm{i}_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

而输出门与输入门类似，建模某一时刻向下一时刻传递的信息：

$$\mathrm{o}_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

此外，遗忘门用来控制此时刻对此时刻信息的关注，并调节上一时刻信息的继承，以此来模拟人类对于不同信息各异的关注度。这一系列操作可被当作是对不同信息分量的加权，这一做法可被视为一种精巧注意力机制（Attention），这种做法在后续的研究中被广泛应用：

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

$$\widetilde{C_t} = \tanh(W_C * [h_{t-1}, x_t] + b_C)$$

最后，通过以上遗忘门建模的长短记忆，对输出门进行加权以确定流向下一时刻的长短记忆信息：

$$\mathrm{C}_t = f_t * C_{t-1} + i_t * C_t$$

$$\mathrm{h}_t = o_t * \tanh(C_t)$$

明显的，作为长记忆C_t，其向下一时刻的输出总包含上一时刻中蕴含的信息，而短记忆h_t则随时可被输出门所截断，并且不保证上一时刻的信息一定被保留。也因此，此种仿生的网络组件形象的建模了人类对时序信号的记忆情况，而网络的自身的非线性和参数容量也保证了该种建模方式丰富的数据表达能力。

5.2.2 多元时序信号预测问题

如众多经典的时序信号分析手段一样，为了方便描述以上的 LSTM 仅讨论单元前提下

的时序信号建模，尽管对于 LSTM 来说，从单元信号任务向多源信号扩展非常简单，即为模型赋予了同时考虑多种因素的能力，而此种能力确实众多传统信号分析方法所不具备的，如 ARMA/ARIMA 以及原始的时间序列分解模型。简明的，将原本 LSTM 模型中的向量操作替换成矩阵操作即可完成单元模型向多源模型的转换，即每一时刻隐变量都为输入信息各元素的加权和，而权重即为网络模型所要学习/调整的参数，以满足模型建立的需要。以输入门为例：

$$[w_1^i \ \ w_2^i] \cdot [h_{t-1} \ x_t] \rightarrow [w_1^i \ \ w_2^i] \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

也因此，对于一段时序信号$\{x_1 x_2 \dots x_t\}$，模型都有一段与之对应的隐变量序列$\{h_1 h_2 \dots h_t\}$与之对应，以此作为对预测任务的参考信息。为了进一步提高，LSTM 模型的表达能力，本章使用一种双向的 LSTM 模型(BiLSTM)，即将原始序列从两个方向（正向与逆向）输入 LSTM，并拼接两个方向的隐变量作为信号的最终表达$\{h_1 h_2 \dots h_t h_{t+1} \dots h_{2t}\}$。最后，为了保证时序信号模型的容量，即保证其可以充分表达预期的时间信号特点，本章采用堆叠 LSTM 的方式来扩增模型容量，即为 Stacked-BiLSTM。

一般的，对于问题关注的六种常规污染物单日浓度值，分别建立模型进行预测。由此，多元的含义被定义为，模型具有多元信息输入能力，并对下一时刻进行多元信息预测。本章使用一个简单的多层感知机（MLP）作为预测模型，在 LSTM 隐变量的基础上为各常规污染物建立预测模型，记作 Stacked-BiLSTM-MLP。为了进一步的简化模型表示，并抑制模型可能存在的过拟合问题，对于各污染物的预测，本章并未设计独立的预测网络，而是为一个 MLP 模型设计多个输出（此处数量为 6，即 MLP 网络的输出为一个末位维度为 6 的张量），在最大程度上共享了模型参数（这一设计在众多应用中被证明是一种提高模型泛化能力的有效手段）。

如图 5-3 所示为本章所述的 Stacked-BiLSTM-MLP 模型的一般化表示，输入特征（一般为 BxNxD 的张量，B 为批次数量，N 是序列长度，D 为输入特征维度，即每时刻信号特征维度）经由 K 轮迭代组成的 Stacked BiLSTM 模型处理，输出维度为 BxNx2D 维度的时序信号特征张量；接着对该向量进行展平操作（Flatten），此时维度为 Bx2ND；将展平后的特征向量送入 MLP 模块，进行最后的预测工作。同样的，该操作将迭代 K 轮，并输出维度为 Bxd 的张量，d 为问题所要预测的污染物数量。

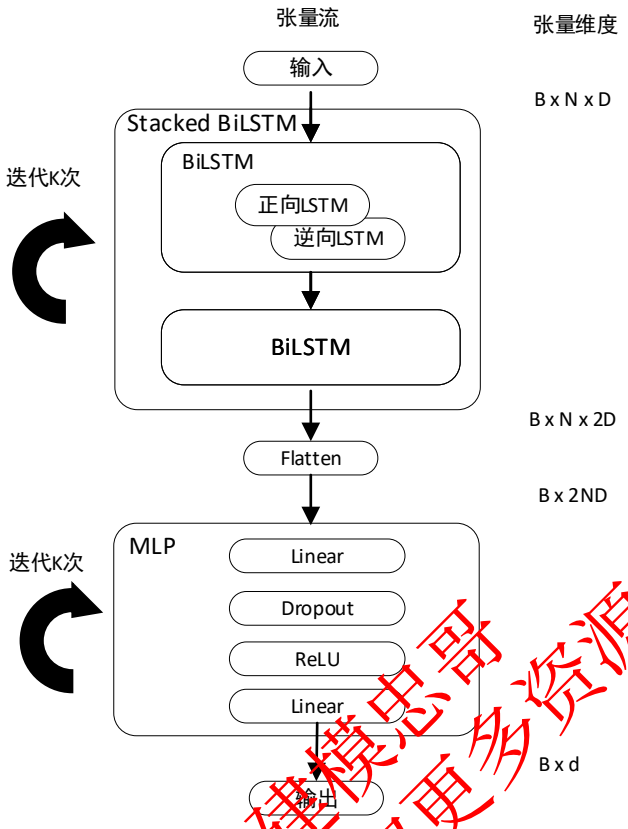

图 5-3 Stacked-BiLSTM-MLP

至此，该模型基本具备了单次的多源时序信号预测的能力，即通过输入一段多元信息序列（此处与六种污染物浓度值可能存在潜在联系的信息），为下一时刻预测一组多元信息（此处为六种污染物浓度值）。但是在问题中要求一次预测三天的污染物浓度，即 2021 年 7 月 13/14/15 号当日的浓度。本章采用一种简单的方式对模型进行相应的扩展：设输入序列的末位时刻为 t，则第一次预测 t+1 完成后，将预测值插入至原始输入后一位构成新输入，此时输入的末位为 t+1，再次输入模型，预测下一时刻 t+2，重复以上步骤，可得到 t+1,t+2,t+3 三个时刻的污染物浓度预测值，即所谓的二次预测。

如上所述，以上模型构成了本章对多元时序信号预测问题解决方案的基础。

5.2.3 跨站点多元时序信号预测问题建模

问题三要求构建 A/B/C 三个监测点的通用模型，尽管以上模型已经具备对单个或者多个监测点建模的能力，但这种能力本质上来说是对所有的信号套用同一的数学模型，并不考虑不同监测点信号的本身的特点。简单来说，站点周围的人口数量、地形、交通情况以及气候特点等因素都会是二次预报结果潜在的影响因素，而将不同站点数据混为一谈明显是不合理的。如何表现出跨站点（即有多个站点）时序信号的个性与共性，是本章具体要讨论的。

一方面，本文认识到不同站点间的信息具备潜在的共性，常规污染物的变化符合同一/或者类似的规律，即尽管站点间互相独立且互不影响，但污染物的变化切合普遍规律，或

者某一区域的污染物变化呈现一定的规律特点；另一方面，不同站点间的信息具备可能的个性，也即局部自然气候、区域地形和人类活动及其对气象数据的影响也会对站点间的污染物预测造成干扰。综上，一个合理的二次预报数学模型既要符合站点间的“普遍规律”，也要充分考虑站点间客观存在的差异对预报的影响。

本章将信号预测模型视为一个概率模型，在符合时序信号概率模型中采样下一时刻的信号。如图 5-4，上一节所建立的可被视为一个概率分布模型$P_G(x;\theta)$，其中 x 为前 T 时刻的信号输入，即视作当前时刻状态，θ为需要学习的参数。而上节 Stacked BiLSTM 的特征输出可被抽象为一种概率分布 Z，作为一个生成器 G 生成新的信号数据特征，而这一数据的分布特征要贴合数据中潜在的概率分布。

同样的，数据所表现出的概率分布被定义为$P_{data}(x;\theta)$，此时θ为需要信号中的隐变量。那么该如何用概率模型去拟合信号数据变成为了当前模型的核心问题，就附录所言的近地面臭氧污染形成机制，自由基循环和 NOx 循环相互耦合作用，使 NO 不断转化为 NO2，NO2 的光解使 O3 逐渐积累，其中的机理复杂且难以明确建模，更不必提多个监测点间自然人文状况的明显差异对模型的影响，而如何从各信号表达中剥离出这种差异性以建模通用模型则显得更加困难。

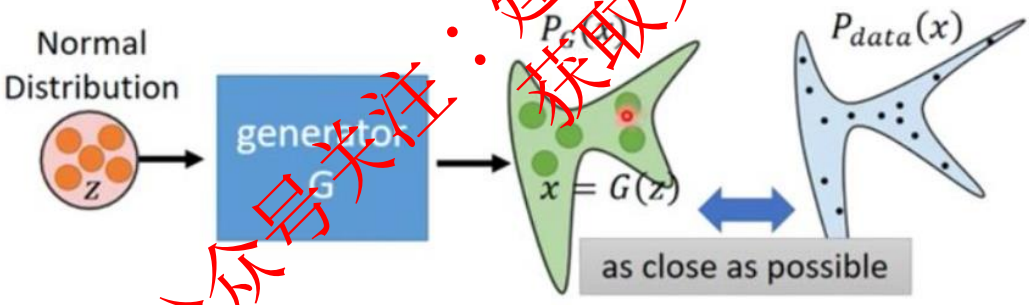

图 5-4 生成器概率分布拟合

虽然，P_G和P_{data}的具体模型并不明确（即θ的明确定义），但是可以从这些分布中直接采样，让这一些样本点分布尽可能的贴合，通过不断调整P_G则可以间接的学习到P_{data}的分布特性，即：

$$G^* = \arg\min_G Div(P_G, P_{data})$$

那么该如何计算它们之间的差异呢，这里本章遵循对抗生成网络（GAN）的思路 [9]，引入一个判别器来计算分部间的差异，即：

$$V(G,D) = E_{x\sim P_{data}}[logD(x)] + E_{x-P_G}[\log(1-D(x))]$$

$$D^* = argmax_D V(D,G)$$

其中，学习的目的是最大化 V(G, D)，要求从P_{data}中的采样 $logD(x)$应该越大越好，而P_G中采样希望他越小越好。综上，一个判别模型应该能够明确的区分不同数据分布中的采样。因此，生成器的优化目标可被改写为：

$$G^* = argmin_G \max_D V(D,G)$$

出于便利性，以上论述将判别器当做是一个二分类任务，实际上P_{data}表达的是另一站点的数据采样，这一采样也可由 G 生成，用来剥离站点的个性化信息。而在问题三的情境中，这将是个三分类任务，生成器用以产生来自不同站点信号的中间表示，而判别器则通过这些表示，来区分他们来自哪个站点。

通俗的来讲，模型希望生成器 G 的能力强，即其产生的数据分布要尽可能符合数据分布；另一方面，模型希望判别器 D 的能力强，即其能够准确判别不同分布间的差异性。以上两种模型的优化本质上的互相矛盾的，即当生成器能力足够强时，判别器其实已经分辨不出数据在分布上的差异，这就是一种对抗性的博弈思路。而这种特点也为本章从信号中剥离出不同站点间（跨站点）的特性提供了便利。也即，当判别模型无法从信号特征中分辨出其所述监测时，则被认为检测点本身的特性已被剥离出来。

基于以上分析，在上一节提出的模型基础的上，模型中加入 GAN 组件，包括一个监测站判别器以及对抗损失（如图 3-5）：一方面模型需要准确预测下一时刻信号数据，另一方面，信号的中间隐变量表达应该对站点不具区分性。综上，完整的模型被表述为 GAN-Stacked-BiLSTM-MLP。

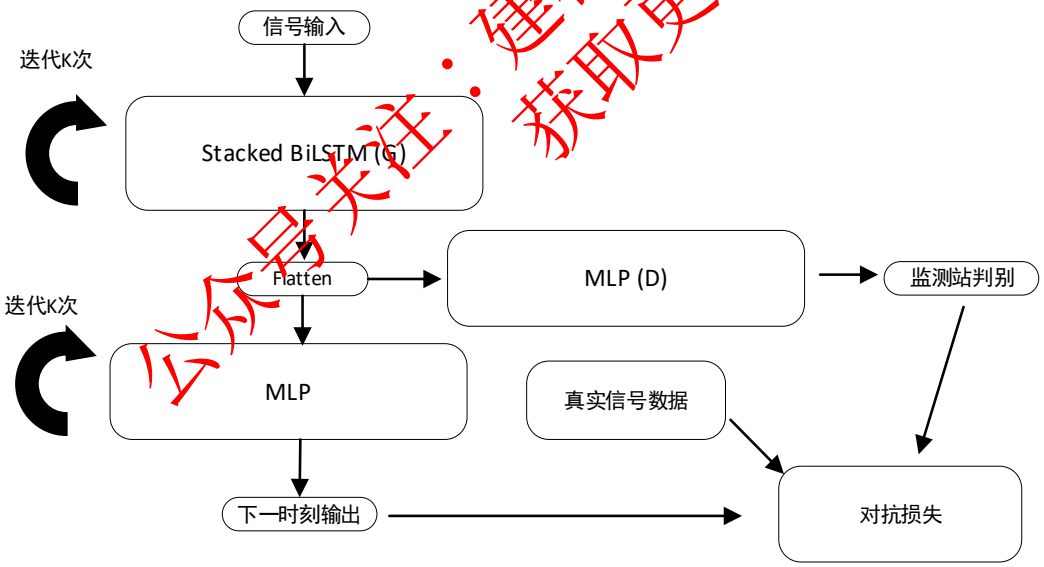

图 5-4 GAN-Stacked-BiLSTM-MLP

5.3 模型优化问题

针对于上一节提出的模型，本章使用基于梯度的优化方法对模型进行微调，优化器使用 AdamW [10]，时序信号$Stacked-BiLSTM-MLP$模型损失函数则记为：

$$L(\theta) = MSE(StackedBiLSTMMLP(X), Y)$$

其中，X 为样本，Y 为样本对应的标签，即下一时刻的污染物浓度。该模型用以约束信号预测应该尽量贴近真实数据。而站点判别则使用一个多分类交叉熵损失来约束隐变量对不同监测点的依赖关系：

$$L_2(\theta) = Cross\_Entropy(D(\hat{X}), Y)$$

其中，$D(\hat{X})$表示由判别器 MLP(D)对 Stacked-BiLSTM 产生的中间变量的判别结果。整个优化过程是轮次迭代的：首先通过L_2损失，对除 MLP(D)外的模型进行冻结，以训练判别器对不同站点信号的区分性能；再次，冻结 MLP(D)模型，通过最小化 L 损失确保模型预测准确度的同时，通过最大化L_2损失确保信号的中间向量不具区分性，即模型的更加关注站点无关的时序信号隐变量特点。

5.4 实验

5.4.1 实验数据与参数设置

本问题中，本章考虑到了气象因素对污染物扩散存在的影响。因此，在实验中本章将污染和气象等因素作为模型的特征输入。具体可描述为，第一，对于时间特征，我们将月、天两个时间因素进行拆分并通过 one-hot 映射为 Embedding 密集嵌入表示；第二，本章对污染物浓度进行编码，包括天粒度的污染物浓度值，和小时粒度的污染物浓度值，小时级别污染物浓度包括 24 小时的最大值，中位数，最小值，方差，和标准差；第三，本章对气象因素进行特征化，气象因素包括温度、湿度、气压、风速、风向的中位数，最小值，方差，和标准差。

由于本问题中未给出后三天的污染物浓度参考值，本章无法直接核验预测的正确性。因此需要对原始的空气质量预报基础数据进行重新组织。实验过程中，本章选择了 80%的是数据样本作为训练集，20%作为测试集验证模型的效果。实验过程，我们的选取的超参数值相同，学习率 0.0005，批大小为 32，输入时间步长为 6，输出步长为 3。经过多次的训练和调参，本章选取测试集上效果最好的参数作为本研究的最终模型。

5.4.2 实验结果与分析

表 5-1 给出了我们的模型和对比模型的实验结果，我们从 MAE、RMSE 和 R 三个衡量尺度来评价本研究提出的预测模型的性能。

表 5-1 模型的预测表现 （A、B、C 综合平均指标）

模型	MAE	RMSE	R
Stacked-BiLSTM-MLP	2.38	4.01	0.74
GAN-Stacked-BiLSTM-MLP	2.46	3.98	0.76

如表 5-1 中，本章分别比较了 Stacked-BiLSTM-MLP 和 GAN-Stacked-BiLSTM-MLP，本章的结果说明在处理时间序列且具有非线性特性的污染物浓度和气象数据，深度模型的能表现出较好的预测性能。对比本研究提出的 Stacked-BiLSTM-MLP 和 GAN-Stacked-

BiLSTM-MLP，差异不明显。这说明在考虑污染物浓度的时间特征时，尽管考虑了各站点的差异性，但还不足以泛化到所有站点预测中。本章的分析如下：

1）在本章设计的问题建模中，是对每个监测站点的污染物浓度分别进行建模。对于单个监测站的的预测，仅使用了对应站点的污染相关数据进行建模。尽管在建模过程中力求通用，剥离了站点间的个性化特点。但是在实际的拟合过程中，这种优势并不明显。因此，本章做出一下假设：其一，站点间的个性化特点作为影响站点污染物变化的重要因素应该被考虑在内。其二，站点间的个性化特点在长序列的时间变化中被弱化，时间序列往往表现出近似的变化特点，因此考虑站点间的个性化特点并无太多意义。尽管以上两个假设互相矛盾，但都是导致实验结果的潜在假设，内里的联系仍需进一步探索。

2）对于时序信号的特征工程问题，本章对 24 小时数据做了简单的采样，压缩了特征维度，一遍与实际的污染物浓度序列一同加入到序列预测任务中去。尽管如此，时序信息的损失仍然是不可避免的，这也限制了模型进一步提升的可能。以下提出一种可能的提升，粗暴的将所有信息送入模型，通过注意力机制让模型自行学习不同因素对结果的影响的权重，即让模型自动的进行特征工程。

综上所述，本章对比了针对本问题提出的模型的缺点和潜在的改进潜力。针对问题三，首先，本文从时序信号预测问题为切入点，选用改进的长短期记忆网络 LSTM 模型对由污染物浓度及其相关的气象数据构成的时序信号进行建模；其次，考虑到污染物浓度与气象数据间存在多元影响的复杂关系以及预测任务属性的多元性特点，对单变量的 LSTM 模型进行扩展，并采用迭代形式预测未来三日的六种污染物浓度变化情况。尽管实验结果并不理想，但也验证了本章方法的科学性和实用价值。

针对问题三中未来三天的污染物浓度结果，表 5-2 给出了 GAN-Stacked-BiLSTM-MLP 的污染物浓度预测结果。AQI 对应的值根据污染物浓度和给定公式计算得出。表 5-3 给出了空气质量分指数 IAQI。

表 5-2 污染物浓度及 AQI 预测结果

预报日期	地点	二次模型日值预测							
		SO_2 ($\mu g/m^3$)	NO_2 ($\mu g/m^3$)	PM_{10} ($\mu g/m^3$)	x ($\mu g/m^3$)	O_3 最大八小时滑动平均 ($\mu g/m^3$)	CO (mg/m^3)	AQI	首要污染物
2021/7/13	监测点 A	7	14	23	6	95	0.5	50	无首要污染物
2021/7/14	监测点 A	8	16	26	9	164	0.4	101	臭氧
2021/7/15	监测点 A	6	16	25	7	111	0.4	59	臭氧
2021/7/13	监测点 B	5	10	15	5	83	0.5	42	无首要污染物
2021/7/14	监测点 B	6	12	16	5	81	0.4	41	无首要污染物
2021/7/15	监测点 B	6	10	14	4	48	0.4	24	无首要污染物

2021/7/13	监测点 C	7	19	37	16	136	0.5	80	臭氧
2021/7/14	监测点 C	8	16	36	18	136	0.4	80	臭氧
2021/7/15	监测点 C	9	21	38	21	117	0.5	63	臭氧

表 5-3 空气质量分指数（IAQI）

预报日期	地点	SO_2	NO_2	PM_{10}	$PM_{2.5}$	O_3 最大八小时滑动平均	CO
2021/7/13	监测点 A	7	19	23	9	48	11
2021/7/14	监测点 A	6	22	27	12	104	10
2021/7/15	监测点 A	6	20	24	10	57	11
2021/7/13	监测点 B	5	12	15	7	42	12
2021/7/14	监测点 B	6	15	16	7	41	10
2021/7/15	监测点 B	6	13	14	6	24	10
2021/7/13	监测点 C	7	24	37	23	80	12
2021/7/14	监测点 C	8	20	35	25	80	10
2021/7/15	监测点 C	9	27	38	30	63	12

六. 问题四：多站点污染物浓度联合预测

6.1 预测方法简介

由于污染物浓度预测的复杂性和许多影响因素的不断变化的性质，多站点污染物浓度长期预测具有很高的挑战性。本研究问题中，我们关注时空因素，并提出了一种污染物图注意网络（PGAN）来预测网络节点上不同位置前方时间步长的污染物浓度状况。PGAN 采用编码器-解码器架构，其中编码器和解码器均由多个时空注意块组成，以模拟时空因素对污染状况的影响。编码器对输入数据特征进行编码，解码器预测输出序列。在编码器和解码器之间，应用 transformer 注意力层来转换编码的输入数据特征，以生成未来时间步长的序列表示作为解码器的输入 ［11］。Transformer 注意力机制对历史和未来时间步长之间的直接关系进行建模，这有助于缓解预测时间步长之间的错误传播问题。我们采用 20%的数据作为测试集，污染物浓度预测任务的实验结果证明了 PMAN 的优越性，特别是，在提前 3 小时的预测中。

6.2 问题分析

多站点的污染物浓度预测任务，受多方面因素影响，包括气象因素，时间因素，空间因素［12］。高效的方法，结合数据本身的特性，有助于污染物浓度预测在精准度上得到一个新的突破。然而，长时间污染物浓度预测任务仍然缺乏令人满意的结果，主要是由于以下问题：

1）动态空间相关性。如图 6-1 所示，监测站点网络中传感器之间污染物浓度的相关性随时间显着变化（例如，在不同强度风向天气时期）。如何动态选择相关传感器的数据来预测目标传感器的长期污染状况是一个具有挑战性的问题。

2）非线性时间相关性。同样在图 6-1 中，传感器处的污染物浓度可能会剧烈而突然地波动（例如，由于污染排放，交通拥挤），从而影响不同时间步长之间的相关性。随着时间的推移，如何对非线性时间相关性进行自适应建模仍然是一个挑战。

3）对错误传播的敏感性。从长远来看，每个时间步中的小错误可能会在对未来进行更远的预测时放大。这种误差传播使得对遥远未来的预测极具挑战性。

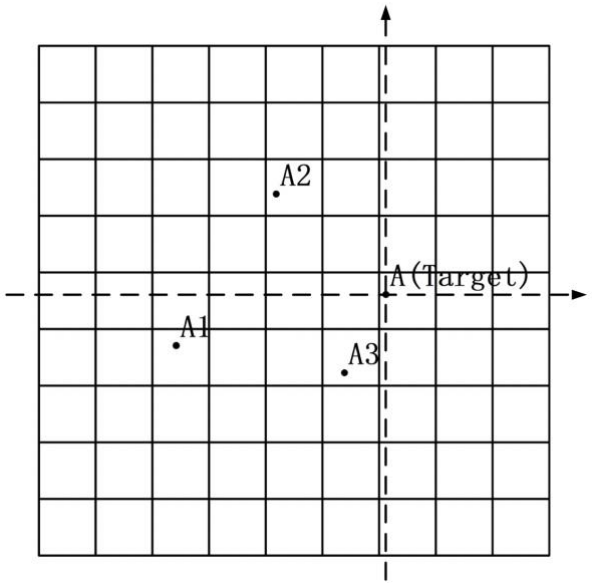

图 6-1 各监测站点相对位置示意图，正东方向为 x 轴，正北方向为 y 轴，单位：km 。A (0, 0)，A1 (-14.4846, -1.9699)，A2 (-6.6716, 7.5953)，A3 (-8.3543, -5.0438)。

6.3 解决方案

为了解决上述挑战，我们提出了一个污染物图多注意网络（PGAN）来预测网络图上多监测站点的污染物浓度随着时间步长。PGAN 遵循编码器-解码器架构，其中编码器对输入特征（污染物浓度、气象因素和时间戳）进行编码，解码器预测输出序列。在编码器和解码器之间添加了一个 transformer 注意力层，以转换编码的历史输入特征以生成未来的表示。编码器和解码器都由一堆时空注意力块组成 ST-Attention 组成。每个 ST-Attention 块由用于建模动态空间相关性的空间注意机制、用于建模非线性时间相关性的时间注意机制以及用于自适应融合空间和时间表示的门控融合机制形成。Transformer 注意力机制模拟历史和未来时间步长之间的直接关系，以减轻错误传播的影响。

1）针对该问题提出了空间和时间注意力机制来分别模拟动态空间和非线性时间相关性。此外，我们设计了一个门控融合来自适应地融合由空间和时间注意力机制提取的信息。

2）此外，我们还提出了一种转换注意力机制来将历史输入特征转换为未来的表示。这种注意力机制模拟历史和未来时间步长之间的直接关系，以减轻错误传播的问题。

6.4 参数定义

我们将监测站点网络表示为加权有向图 $G = (V, E, A)$。这里，V 是一组 $N = |V|$表示监测站点网络上（例如，A1 传感器）的节点；E 是一组边，表示顶点之间的连通性；$A \in R^{N \times N}$是加权邻接矩阵，其中 A_{v_i,v_j}表示顶点v_i和v_j之间的接近度（由监测站点的坐标位置距离测量）。时间步长 t 的污染状况表示为图G上的图信号 $X_t \in R^{N \times C}$，其中C是感兴趣的污染相关特征的数量（污染物浓度、气象因素和时间戳）。鉴于在历史 P 时间步长的 N

个顶点上的观测值 $X = (X_{t_1}, X_{t_2}, \ldots, X_{t_P}) \in R^{P \times N \times C}$，我们的目标是预测所有顶点的下$Q$个时间步长的污染物浓度，表示为$\hat{Y} = (\hat{X}_{t_{P+1}}, \hat{X}_{t_{P+2}}, \ldots, \hat{X}_{t_{P+Q}}) \in R^{Q \times N \times H}$，$H$表示 6 个污染物属性。

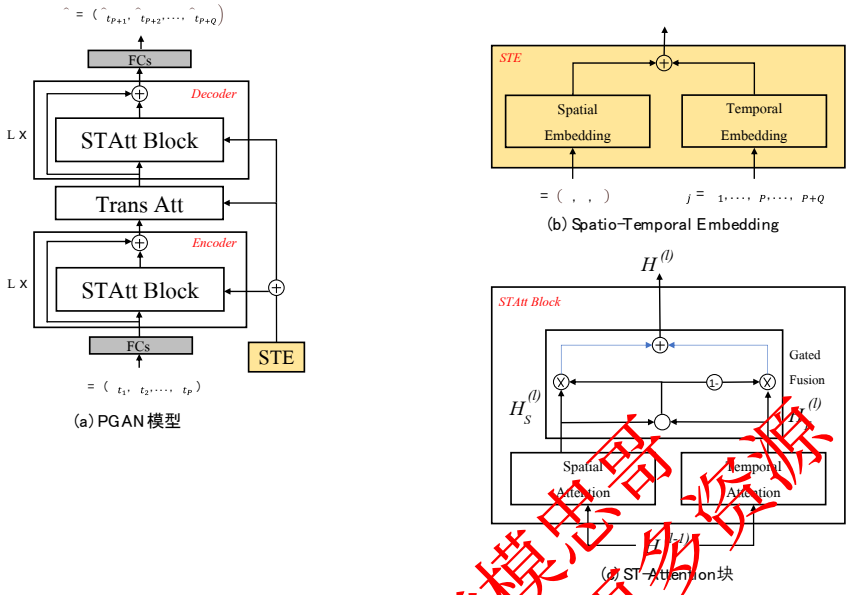

图 6-2 污染物图多注意力网络（PGAN）的框架。(a) PGAN 由一个时空嵌入 (STE)、一个编码器和一个解码器组成，两者都带有 L 个时空注意力 ST-Attention 块（STAtt Block）、一个 transformer 注意力层 (TransAtt) 和两个全连接层。(b) 时空嵌入包含空间嵌入和时间嵌入。(c) ST-Attention 块通过门控融合结合了空间和时间注意机制。

6.5 方法描述

图 6-2 说明了我们提出的污染物图多注意网络(PGAN)的框架，它具有编码器-解码器结构。编码器和解码器都包含 L 个带有残差连接的 ST-Attention 块。每个 ST-Attention 块由具有门控融合的空间和时间注意力机制组成。在编码器和解码器之间，向网络添加了一个 transformer 注意力层，以将编码的输入特征转换到解码器。我们还通过时空嵌入（STE）将图结构和时间信息合并到多注意机制中。此外，为了便于残差连接，所有层都产生 D 维的输出。

6.5.1 时空嵌入

由于污染物浓度的演变受到基础检测站点网络的限制，因此将监测站点网络信息纳入预测模型至关重要。为此，我们提出了一种空间嵌入，将顶点编码为保留图结构信息的向量。具体来说，我们利用 node2vec 方法来学习顶点表示 [13]。此外，为了与整个模型共同训练预学习的向量，这些向量被输入到一个两层的全连接神经网络中。然后，我们获得空间嵌入，表示为 $e_{v_i}^S \in R^D$，其中 $v_i \in V$。

空间嵌入仅提供静态表示，不能表示监测站点网络中监测传感器之间的动态相关性。因此，我们进一步提出了一个时间嵌入来将每个时间步长编码为一个向量。具体来说，让

一天有T个时间步长。我们使用 one-hot 编码将每个时间步的星期几和一天中的时间编码到R^7和R^T中，并将它们连接成一个向量R^{7+T} [14]。接下来，我们应用两层全连接神经网络将时间特征转换为向量R^D。在我们的模型中，我们嵌入了历史 P 和未来 Q 时间步长的时间特征，表示为 $e_{v_j}^T \in R^D$，其中 $t_j = t_1, \ldots, t_P, \ldots, t_{P+Q}$。

为了获得时变顶点表示，我们将上述空间嵌入和时间嵌入融合为时空嵌入（STE），如图 6-2（b）所示。具体来说，对于时间步长t_j的顶点 v_i，STE 定义为 $e_{v_i,t_j} = e_{v_i}^S + e_{v_j}^T$。因此，$P+Q$时间步长中$N$个顶点的 STE 表示为$E \in R^{(P+Q)\times N\times D}$。STE 包含图结构和时间信息，它将用于空间、时间和 transfomer 注意力机制中。

6.5.2 ST-Attention 块

如图 6-2（c）所示，ST-Attention 块包括空间注意力、时间注意力和门控融合。我们将第l个块的输入表示为 $H^{(l-1)}$，其中顶点v_i在时间步长t_j的隐藏状态表示为 $h_{v_i,t_j}^{(l-1)}$。第l个块中空间和时间注意力机制的输出表示为H_S^l和H_T^l，其中顶点v_i在时间步t_j的隐藏状态分别表示为$hs_{v_i,t_j}^{(l)}$和$ht_{v_i,t_j}^{(l)}$。在门控融合之后，我们得到第l^{th}个块的输出，表示为$H^{(l)}$。出于说明目的，我们将非线性变换表示为：

$$f(x) = ReLU(xW + b) \quad (1)$$

其中 W，b 是可学习参数，$ReLU$是激活函数。

6.5.3 空间注意力

一个监测站点的污染物浓度受其他监测站点的影响，影响程度不同。这种影响是高度动态的，会随着时间而变化。为了对这些属性进行建模，我们设计了一种空间注意力机制来自适应地捕获污染物监测站点网络中传感器之间的相关性。关键思想是在不同的时间步为不同的顶点（例如传感器）动态分配不同的权重。对于时间步t_j的顶点v_i，我们计算所有顶点的加权和：

$$hs_{v_i,t_j}^{(l)} = \sum_{v\in V} \alpha_{v_i,v} \cdot h_{v,t_j}^{(l-1)} \quad (2)$$

其中V表示所有顶点的集合，$\alpha_{v_i,v}$是注意力分数，表示顶点 v对v_i的重要性，注意力分数的总和等于 1：$\sum_{v\in V} \alpha_{v_i,v} = 1$。

在某个时间步长，当前的污染物浓度和监测站点网络结构都会影响传感器之间的相关性。例如，某个监测站点的严重污染可能会显着影响其相邻站点的污染物浓度。受这种直觉的启发，我们考虑污染相关因素特征和图结构来学习注意力分数。具体来说，我们将隐藏状态与时空嵌入连接起来，并采用缩放点积方法来计算顶点v_i和v之间的相关性：

$$s_{v_i,v} = \frac{\langle h_{v_i,t_j}^{(l-1)} \| e_{v_i,t_j}, h_{v_i,t_j}^{(l-1)} \| e_{v,t_j} \rangle}{\sqrt{2D}} \quad (3)$$

其中，$\|$表示连接操作，$\langle\ \rangle$表示内积，2D 表示$h_{v_i,t_j}^{(l-1)} \| e_{v_i,t_j}$的维度。然后，$s_{v_i,v}$通过 softmax 函数进行归一化：

$$\alpha_{v_i,v} = \frac{exp(s_{v_i,v})}{\sum_{v_r \in V} exp(s_{v_i,v_r})} \quad (4)$$

得到注意力分数$\alpha_{v_i,v}$后，可以通过等式（2）更新隐藏状态。

为了稳定学习过程，我们将空间注意力机制扩展为多头注意力机制。具体来说，我们将 K 个并行注意力机制与不同的可学习投影连接起来：

$$s_{v_i,v}^{(k)} = \frac{\langle f_{s,1}^{(k)}(h_{v_i,t_j}^{(l-1)} \| e_{v_i,t_j}), f_{s,2}^{(k)}(h_{v_i,t_j}^{(l-1)} \| e_{v,t_j}) \rangle}{\sqrt{d}} \quad (5)$$

$$\alpha_{v_i,v}^{(k)} = \frac{exp(s_{v_i,v}^{(k)})}{\sum_{v_r \in V} exp(s_{v_i,v_r}^{(k)})} \quad (6)$$

$$hs_{v_i,t_j}^{(l)} = \|_{k=1}^{K} \left\{ \sum_{v \in V} \alpha_{v_i,v}^{(k)} . f_{s,3}^{(k)}(h_{v_i,t_j}^{(l-1)}) \right\} \quad (7)$$

其中第k^{th}头注意中$f_{s,1}^{(k)}$，$f_{s,1}^{(k)}$，和$f_{s,1}^{(k)}$表示不同的非线性函数，如公式（1）所示，并除以$d = D/K$维度。

6.5.4 空间注意力

一个位置的污染物浓度与其先前的观察相关，并且相关性随时间步长非线性变化。为了对这些属性进行建模，我们设计了一种时间注意机制来自适应地对不同时间步长之间的非线性相关性进行建模。请注意，时间相关性受污染物状况和污染物浓度上下文语义的影响。例如，发生在无风气象时段的高污染监测站点可能会影响未来同站点污染物浓度几个小时。因此，我们同时考虑污染特征和时间来衡量不同时间步长之间的相关性。具体来说，我们将隐藏状态与时空嵌入连接起来，并采用多头方法来计算注意力分数。形式上，考虑顶点v_i，时间步长t_j和 t 之间的相关性定义为：

$$\mu_{t_j,t}^{(k)} = \frac{\langle f_{t,1}^{(k)}(h_{v_i,t_j}^{(l-1)} \| e_{v_i,t_j}), f_{t,2}^{(k)}(h_{v_i,t}^{(l-1)} \| e_{v,t}) \rangle}{\sqrt{d}} \quad (8)$$

$$\beta_{t_j,v}^{(k)} = \frac{exp(\mu_{t_j,t}^{(k)})}{\sum_{t_r \in N_{t_j}} exp(\mu_{t_j,t_r}^{(k)})} \quad (9)$$

其中 $\mu_{t_j,t}^{(k)}$ 表示第k 个头中t_j和t的注意力分数，即表示时间步长 t对t_j的重要性。一旦获得了注意力分数，则在时间步t_j处顶点v_i的隐藏状态更新如下：

$$ht_{v_i,t_j}^{(l)} = \|_{k=1}^{K} \left\{ \sum_{t \in N_{t_j}} \beta_{t_j,v}^{(k)} . f_{t,3}^{(k)}(h_{v_i,t}^{(l-1)}) \right\} \quad (10)$$

公式（8）、（9）和（10）中的可学习参数通过并行计算在所有顶点和时间步长之间共享。

6.5.5 门控融合机制

某个监测站点在某一时间步的污染物浓度状况与其之前的值和其他站点的污染物浓度状况相关。如图 6-2（c）所示，我们设计了一个门控融合来自适应地融合空间和时间表示。在第l个块中，空间和时间注意力机制的输出表示为H_S^l和H_T^l，它们在编码器中的形状为$R^{P \times N \times D}$ 或在解码器中具有$R^{Q \times N \times D}$的形状。H_S^l和H_T^l融合为：

$$H^l = z \odot H_S^l + (1-z) \odot H_T^l \quad (11)$$

$$z = \sigma(H_S^l W_{z,1} + H_T^l W_{z,2} + b_z) \quad (12)$$

其中W和b表示的是可学习的参数，$\odot$表示的矩阵相乘，σ表示 sigmoid 函数，z表示门。门控融合机制自适应地控制每个顶点和时间步长的空间和时间依赖性的流动。

6.5.6 Transformer 注意力机制

为了缓解长时间范围内不同预测时间步长之间的错误传播效应，我们在编码器和解码器之间添加了一个 transformer 注意层。它对每个未来时间步长和每个历史时间步长之间的直接关系进行建模，以转换编码的输入数据特征以生成未来表示作为解码器的输入。对于顶点 v_i，预测时间步长t_j（$t_j = t_{P+1}, \ldots, t_{P+Q}$）与历史时间步长 t（$t = t_1, \ldots, t_P$）之间的相关性通过时空嵌入测量：

$$\lambda_{t_i,t}^{(k)} = \frac{\langle f_{tr,1}^{(k)}(e_{v_i,t_j}), f_{tr,2}^{(k)}(e_{v_i,t}) \rangle}{\sqrt{d}} \quad (13)$$

$$\gamma_{t_j,t_r}^{(k)} = \frac{exp(\lambda_{t_j,t}^{(k)})}{\sum_{t_r=t_1}^{t_P} exp(\lambda_{t_j,t}^{(k)})} \quad (14)$$

使用注意力分数$\gamma_{t_j,t_r}^{(k)}$，通过在所有历史 P 时间步长中自适应地选择相关特征，将编码的输入特征转换到解码器中：

$$ht_{v_i,t_j}^{(l)} = \Big\|_{k=1}^{K} \Big\{ \sum_{t=t_1}^{t_P} \gamma_{t_j,t}^{(k)} \cdot f_{t,3}^{(k)}(h_{v_i,t}^{(l-1)}) \Big\} \quad (15)$$

公式（13）、（14）和（15）中的可学习参数通过并行计算在所有顶点和时间步长之间共享。

6.5.7 编码-解码

如图 6-2（a） 所示，PGAN 是一种编码器-解码器架构。在进入编码器之前，使用全连接层将历史观察 $X \in R^{P \times N \times C}$ 转换为 $H^{(0)} \in R^{P \times N \times D}$。 然后，$H^{(0)}$被送入带有 L 个 ST-Attention 块的编码器，并产生一个输出$H^{(L)} \in R^{P \times N \times D}$。 在编码器之后，添加一个变换注意层来转换编码特征$H^{(L)}$以生成未来序列表示$H^{(L+1)} \in R^{P \times N \times D}$。 接下来，解码器在$H^{(L+1)}$上堆叠 L 个 ST-Attention 块，并产生输出为$H^{(2L+1)} \in R^{P \times N \times D}$。 最后，全连接层产生 Q 时间步长预测 $\hat{Y} \in R^{Q \times N \times C}$。

以上我们介绍了 PGAN 的整个特征提取的算法详细流程。PGAN 可以通过反向传播通过最小化预测值和真实值之间的平均绝对误差 (MAE) 进行端到端训练：

$$L(\theta) = \frac{1}{Q} \sum_{t=t_{P+1}}^{t_{P+Q}} |Y_t - \hat{Y}_t| \quad (16)$$

6.6 实验

6.6.1 实验数据与参数设置

本问题中，我们考虑到了气象因素对污染物扩散存在的影响。因此，在实验中，我们将污染和气象等因素作为模型的特征输入。具体可描述为，第一，对于时间特征，我们将月、天两个时间因素进行拆分并通过 one-hot 映射为 Embedding 密集嵌入表示；第二，我们对污染物浓度进行编码，包括天粒度的污染物浓度值，和小时粒度的污染物浓度值，小时级别污染物浓度包括 24 小时的最大值，中位数，最小值，方差，和标准差；第三，我们对气象因素进行特征化，气象因素包括温度、湿度、气压、风速、风向的中位数，最小值，方差，和标准差。

实验过程中，我们选择了 80%的是数据样本作为训练集，20%作为测试集验证模型的效果。实验过程，我们的选取的超参数值相同，学习率 0.0005，批大小为 32，输入时间步长为 6，输出步长为 3。结果多次的训练和调参，我们选取测试集上效果最好的参数作为本研究的最终模型。

6.6.2 实验结果与分析

表 6-1 给出了我们的模型和对比模型的实验结果，我们从 MAE、RMSE 和 R 三个衡量尺度来评价本研究提出的预测模型的性能［15］。

表 6-1 模型的预测表现（A、A1、A2、A3综合平均指标）

模型	MAE	RMSE	R
RNN	4.76	5.32	0.69
LSTM	4.34	5.01	0.70
Stacked-BiLSTM-MLP	2.41	3.80	0.74
GAN-Stacked-BiLSTM-MLP	2.32	3.75	0.77
PGAN	1.85	3.21	0.89

从表 6-1 中我们可以看到，我们发现深度学习模型 RNN，LSTM 相对于传统的机器学习模型，优势尽显。这说明在处理时间序列且具有非线性特性的污染物浓度和气象数据，深度模型的优势更大。对比本研究提出的 PGAN 预测模型与 RNN 和 LSTM，我们的模型 PGAN 在测试集上的表现效果最好，MAE 值为 1.85，RMSE 值为 3.21，R 值为 0.89。这说明在考虑污染物浓度的时空特征时，模型的拟合程度要更好。并且，对比第三问题中 A 值的实验结果，我们可以看出模型的精准度得到了较大的改善。我们的分析如下：

1）问题三，我们是对每个监测站点的污染物浓度分别进行建模。针对 A 站点，我们仅用 A 站点的数据进行建模，我们忽视周边站点在污染扩散情况下可能对 A 站点产生的影响。因此，问题三中，从空间维度上，我们缺乏科学性，和实际数据的支撑。污染物扩散在气象因素的条件下，运动时极其复杂的。在问题四中，我们加入了周边站点和目标站点之间的空间相关性研究，在一定程度上改善了预测的表现。

2）在加入空间因素的情况下，问题四我们考虑了时空的动态特征。空间维度的变化特性和时间维度分不开，污染物的扩散随着时间的流逝是动态变化的过程。这是问题三中的模型无法解决的，这也是问题三模型的另一个不足点。本问题中，我们将时间维度和空间维度进行了结合，这是我们相比问题三中的另一个创新的地方。

3）时间维度和空间维度的变化，对我们的预测影响力是不同的。为此，我们加入了时空注意力机制。时空注意力机制，在空间维度中动态的感知不同位置的监测站点的污染物浓度对目标站点的影响。在时间维度上，我们动态的计算历史输入数据对当前预测产生的影响。时空注意力模型较问题三中模型的理论和实际应用价值有了质的改变，具体细节我们已经在文中做了详细的介绍。因此，这也是我们模型效果提升的另一个不可缺少的因素。

4）最后就是长时间序列预测问题，我们使用 transformer 模型解决了编码器和解码器之间时空特征馈入问题。我们在文中介绍了我们减少长时间序列预测误差传递的机制，这也是我们和问题三中预测过程的重大却别。我们在本问题中保证了模型参数误差渗透问题，从而保证了最终的预测精准度。

综上所述，我们对比了本问题提出的模型和问题三中使用模型的却别和优势。针对问题四我们提出了空间和时间注意力机制来分别模拟动态空间和非线性时间相关性。此外，我们设计了一个门控融合来自适应地融合由空间和时间注意力机制提取的信息。我们还提出了一种转换注意力机制来将历史输入特征转换为未来的表示。这种注意力机制模拟历史和未来时间步长之间的直接关系，以减轻错误传播的问题。实验结果也验证了我们方法的科学性和实用价值。

未来预测问题四中未来三天的污染物浓度结果，表 6-2 给出了 PGAN 的污染物浓度预测结果。AQI 对应的值根据污染物浓度和给定公式计算得出。表 6-3 给出了空气质量分指数 IAQI。

表 6-2 污染物浓度及 AQI 预测结果

预报日期	地点	二次模型日值预测							
		SO_2 ($\mu g/m^3$)	NO_2 ($\mu g/m^3$)	PM_{10} ($\mu g/m^3$)	x ($\mu g/m^3$)	O_3 最大八小时滑动平均 ($\mu g/m^3$)	CO (mg/m^3)	AQI	首要污染物
2021/7/13	监测点 A	7	15	23	6	94	0.4	47	无首要污染物
2021/7/14	监测点 A	7	16	26	9	163	0.4	103	臭氧
2021/7/15	监测点 A	6	15	22	6	110	0.4	58	臭氧
2021/7/13	监测点 A1	7	13	23	6	83	0.4	42	无首要污染物
2021/7/14	监测点 A1	6	14	22	8	105	0.4	54	臭氧
2021/7/15	监测点 A1	6	13	21	7	98	0.4	49	无首要污染

									物
2021/7/13	监测点 A2	4	14	23	9	118	0.4	65	臭氧
2021/7/14	监测点 A2	4	11	24	10	127	0.4	72	臭氧
2021/7/15	监测点 A2	5	16	26	11	132	0.4	77	臭氧
2021/7/13	监测点 A3	5	13	13	6	90	0.4	45	无首要污染物
2021/7/14	监测点 A3	5	13	19	8	142	0.4	92	臭氧
2021/7/15	监测点 A3	4	13	13	8	111	0.4	61	臭氧

表 6-3 空气质量分指数（IAQI）

预报日期	地点	SO_2	NO_2	PM_{10}	$PM_{2.5}$	O_3 最大八小时滑动平均	CO
2021/7/13	监测点 A	7	19	23	9	47	10
2021/7/14	监测点 A	7	20	26	13	103	10
2021/7/15	监测点 A	6	19	22	9	58	10
2021/7/13	监测点 A1	7	16	23	9	42	10
2021/7/14	监测点 A1	6	18	22	11	54	10
2021/7/15	监测点 A1	6	16	21	10	49	10
2021/7/13	监测点 A2	4	18	23	13	65	10
2021/7/14	监测点 A2	4	14	24	14	72	10
2021/7/15	监测点 A2	5	20	26	16	77	10
2021/7/13	监测点 A3	5	16	13	9	45	10
2021/7/14	监测点 A3	5	16	19	11	92	10
2021/7/15	监测点 A3	4	16	13	11	61	10

参考文献

[1]. 中华人民共和国生态环境部. 2019 年中国生态环境统计年报.[DB/OL]. https://www.mee.gov.cn/hjzl/sthjzk/sthjtjnb/202108/W020210827611248993188.pdf

[2]. 2019 年全球空气状况中文报告. .[DB/OL]. http://www.szguanjia.cn/article/2360

[3]. 刘亚梦. 我国大气污染物时空分布及其与气象因素的关系. 兰州大学.

[4]. 中华人民共和国生态环境部. 2020 中国生态环境状况公报.[DB/OL]. https://www.mee.gov.cn/hjzl/sthjzk/zghjzkgb/202105/P020210526572756184785.pdf

[5]. 国家大气污染防治攻关联合中心.【专家解读】臭氧污染从何而来？如何产生？.[DB/DL]. https://mp.weixin.qq.com/s/Q0Je_KMJb7Ia7l5U_w2xWA

[6]. Ziegel E R， Box G， Jenkins G M，et al. Time Series Analysis, Forecasting, and Control[J]. Journal of Time, 1976, 31(2):238-242.

[7]. DE Rumelhart, Hinton G E， Williams R J . Learning Representations by Back Propagating Errors[J]. Nature, 1986, 323(6088):533-536.

[8]. Hochreiter, Sepp, Schmidhuber, et al. Long short-term memory[J]. Neural Computation, 1997.

[9]. Goodfellow I J， Pouget-Abadie J， Mirza M，et al. Generative Adversarial Networks[J]. Advances in Neural Information Processing Systems, 2014, 3:2672-2680.

[10]. Loshchilov I， Hutter F . Decoupled Weight Decay Regularization[J]. Arxiv. 2017.

[11]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

[12]. Ly, H. B., Le, L. M., Phi, L. V., Phan, V. H., Tran, V. Q., Pham, B. T., ... & Derrible, S. (2019). Development of an AI model to measure traffic air pollution from multisensor and weather data. *Sensors*, *19*(22), 4941.

[13]. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., & Tang, J. (2018, February). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 459-467).

[14]. Xiong, Y., Chen, S., Qin, H., Cao, H., Shen, Y., Wang, X., ... & Tang, B. (2020). Distributed representation and one-hot representation fusion with gated network for clinical semantic textual similarity. *BMC medical informatics and decision making*, *20*(1), 1-7.

[15]. Chang, Y. S., Chiao, H. T., Abimannan, S., Huang, Y. P., Tsai, Y. T., & Lin, K. M. (2020). An LSTM-based aggregated model for air pollution forecasting. *Atmospheric Pollution Research*, *11*(8), 1451-1463.

附录

问题 1 代码

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")

plt.rcParams['font.sans-serif'] = ['SimHei'] #步骤一（替换 sans-serif 字体）
plt.rcParams['axes.unicode_minus'] = False #步骤二（解决坐标轴负数的负号显示问题）

O3_df=pd.read_excel("../附件 1 监测点 A 空气质量预报基础数据.xlsx",sheet_name=1,index_col=0,usecols=[0,6])
O3_df_range=O3_df.loc["2020-8-25":"2020-8-28"]

O3_rs_dict={}
for i in range(4):
    print(f"第{25+i}天的数据情况如下：")
    a=O3_eight_hours.iloc[i*24:(i+1)*24,:]
    print(a)
    max_value=a.max()
    print(max_value)
    day=25+i
    key="2020-8-"+str(day)
    O3_rs_dict[key]=max_value


day_ground=pd.read_excel("../附件 1 监测点 A 空气质量预报基础数据.xlsx",sheet_name=2,index_col=0)
day_ground.drop(["地点"],axis=1,inplace=True)
day_ground.columns=[i.split("(")[0] for i in list(day_ground.columns)]

q1_data=day_ground["2020-8-25":"2020-8-28"]
q1_data.iloc[:,4]=[112.25,92.125,169.125,200.75]
IAQI_table={
    "IAQI":[0,50,100,150,200,300,400,500],
```

```
    "CO":[0,2,4,14,24,36,48,60],
    "SO2":[0,50,150,475,800,1600,2100,2620],
    "NO2":[0,40,80,180,280,565,750,940],
    "O3":[0,100,160,215,265,800,np.nan,np.nan],
    "PM10":[0,50,150,250,350,420,500,600],
    "PM2.5":[0,35,75,115,150,250,350,500]
           }
IAQI_DF=pd.DataFrame(data=IAQI_table,index=None)
IAQI=IAQI_DF.iloc[:,0].tolist()

def calculate_AQI(C_p):
    """
    每次计算一种指标的四天的值，返回一个字典，最后放一起比一下。每次传入 q1_data 的列名，算一个的
    """
    C_p_value=q1_data.loc[:,C_p].tolist()
    C_p_range=IAQI_DF.loc[:,C_p].tolist()

    C_p_dict={}
    C_p_list=[]

    for num in C_p_value:
        for i in range(7):
            if (num > C_p_range[i]) and (num < C_p_range[i+1]):
                Lo=i  # 低位值和高位值序号只差1，只记录一个就可以
                break
        Lo_value=C_p_range[Lo]
        Hi_value=C_p_range[(Lo+1)]
        IAQI_Lo_value=IAQI[Lo]
        IAQI_Hi_value=IAQI[(Lo+1)]

        IAQI_p=math.ceil((IAQI_Hi_value-IAQI_Lo_value)/(Hi_value-Lo_value)*(num-Lo_value) + IAQI_Lo_value)

        C_p_list.append({"IAQI":IAQI_p,"AQI_lo":IAQI_Lo_value})
    return C_p_list

rs_dict={}
for i in ["SO2","NO2","PM10","PM2.5","O3","CO"]:
    a=calculate_AQI(i)
    print(f"{i}: {a}")
    a_value=list(i['IAQI'] for i in a)
    rs_dict[i]=a_value
    print(f"\n 对于{i}来说：单纯 IAQI 值：{a_value}\n")
```

```
78.
79.
80.rs_df=pd.DataFrame(data=rs_dict)
81.rs_df.max(axis=1)
```

问题 2 代码

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. import numpy as np
4. import seaborn as sn
5. %matplotlib inline
6.
7. import warnings
8. warnings.filterwarnings("ignore")
9.
10.plt.rcParams['font.sans-serif'] = ['SimHei'] #步骤一（替换 sans-serif 字体）
11.plt.rcParams['axes.unicode_minus'] = False #步骤二（解决坐标轴负数的负号显
   示问题）
12.
13.day_ground=pd.read_excel("../附件 1 监测点 A 空气质量预报基础数
   据.xlsx",sheet_name=2,index_col=0)
14.day_ground.drop(["地点"],axis=1,inplace=True)
15.day_ground.columns=[i.split("(")[0] for i in list(day_ground.columns)]
16.
17.plt.figure(figsize=(15,15))
18.ax1=plt.subplot(3,1,1)
19.ax1.set_title("原数据")
20.plt.plot(day_ground['PM10'][:"2020-04-30"], color="c")
21.
22.ax2=plt.subplot(3,1,2)
23.ax2.set_title("填充插值")
24.plt.plot(day_ground["PM10"][:"2020-04-
   30"].interpolate(method='pad', limit=2),color="r")
25.
26.ax3=plt.subplot(3,1,3)
27.ax3.set_title("线性插值")
28.plt.plot(day_ground["PM10"][:"2020-04-
   30"].interpolate(method='linear', limit=2),color="b")
29.
30.plt.savefig("插值 1.png", dpi=150)
31.
32.plt.figure(figsize=(20,10))
33.plt.xticks(fontsize=20)
```

```
plt.yticks(fontsize=20)
plt.plot(day_ground['PM10']["2019-05-31":"2019-12-15"], color="c",linewidth=4)
plt.plot(day_ground["PM10"]["2019-05-31":"2019-12-15"].interpolate(method='linear', limit=2),'o',color='lightcoral',linewidth=3)
plt.plot(day_ground["PM10"]["2019-05-31":"2019-12-15"].interpolate(method='pad', limit=2),'--',color='steelblue')
plt.legend(['row data', 'linear', 'pad'], loc='best')

plt.savefig("插值 3.png", dpi=150)

# 未插值的图像
plt.figure(figsize=(15,15))
ax1=plt.subplot(3,1,1)
ax1.set_title("原数据")
plt.plot(day_ground['PM10'][:"2020-04-30"],color="b")

ax2=plt.subplot(3,1,2)
ax2.set_title("时间插值")
plt.plot(day_ground["PM10"][:"2020-04-30"].interpolate(method='time', limit=2),color="r")

ax3=plt.subplot(3,1,3)
ax3.set_title("最近邻插值")
plt.plot(day_ground["PM10"][:"2020-04-30"].interpolate(method='nearest', limit=2),color="b")

plt.savefig("插值 2.png", dpi=150)

corr_day = day_ground.corr()
corr_day

mask = np.array(corr_day)
mask[np.tril_indices_from(mask)] = False
# 绘制图像

fig,ax = plt.subplots()
fig.set_size_inches(20,10)
# Spectral  bwr twilight Pastel1
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

sn.heatmap(corr_day,mask=mask,vmax=0.9,square=True,annot=True)
```

```
72.
73.plt.yticks(rotation=360)
74.plt.savefig("相关 1.png", dpi=150)
75.
76.weather_air=pd.read_excel("../附件 1 监测点 A 空气质量预报基础数
   据.xlsx",sheet_name=1,index_col=0,na_values=['NAN', 'NA', '—'])
77.
78.weather_air.drop("地点",axis=1,inplace=True)
79.weather_air.columns=[i.split("
   （")[0].split("(")[0] for i in list(weather_air.columns)]
80.weather_air.fillna(value=0,inplace=True)
81.
82._, ax = plt.subplots(figsize=(12, 10))
83.corr = weather_air.corr(method='pearson')
84.# corr = df.corr(method='kendall')
85.# corr = df.corr(method='spearman')
86.cmap = sns.diverging_palette(220, 10, as_cmap=True)  # 在两种 HUSL 颜色之间制
   作不同的调色板。图的正负色彩范围为 220、10，结束为真则返回 matplotlib 的 colormap
   对象
87._ = sns.heatmap(
88.    corr,  # 使用 Pandas DataFrame 数据，索引/列信息用于标记列和行
89.    cmap=cmap,  # 数据值到颜色空间的映射
90.    square=True,  # 每个单元格都是正方形
91.    cbar_kws={'shrink': .9},  # `fig.colorbar`的关键字参数
92.    ax=ax,  # 绘制图的轴
93.    annot=True,  # 在单元格中标注数据值
94.    annot_kws={'fontsize': 12})  # 热图，将矩形数据绘制为颜色编码矩阵
95.
96.plt.xticks(rotation=30)
97.plt.savefig("相关 2.png", dpi=150)
98.plt.show()
99.
100.once_predict=pd.read_excel("../附件 1 监测点 A 空气质量预报基础数
   据.xlsx",sheet_name=0,index_col=0,na_values=['NAN', 'NA', '—'])
101.once_predict_air=once_predict.iloc[:,2:]
102.once_predict_air.columns=[i.split("
   （")[0].split("(")[0] for i in list(once_predict_air.columns)]
103.once_predict_air.head()
104.once_predict_air.fillna(value=0,inplace=True)
105.
106.plt.figure(figsize=(14,12))
107.corr=once_predict_air.corr()
108.mask = np.array(corr)
109.mask[np.tril_indices_from(mask)] = False
```

```
110.
111.sns.heatmap(corr,mask=mask,linewidths=0.1,vmax=1.0,square=True,linecolor=
   'white',annot=True,cmap="Spectral")
112.plt.xticks(rotation=30)
113.
114.plt.savefig("相关 3.png", dpi=150)
115.
116.once_predict_air[["O3 小时平均浓度","地表温度"]]["2020-09-01":"2020-12-31"]
117.
118.Week_data=once_predict_air.resample("W").mean()
119.
120.fig, ax1 = plt.subplots(figsize=(16,8))
121.plt.xticks(rotation=45,fontsize=14)
122.plt.yticks(fontsize=14)
123.
124.ax1.bar(Week_data.index, Week_data["O3 小时平均浓度
   "], color="blue",label="O3 小时平均浓度",alpha=0.5, width=3)
125.ax1.set_xlabel("月份")
126.ax1.set_ylabel("O3 小时平均浓度")
127.
128.ax2 = ax1.twinx()
129.ax2.plot(Week_data.index, Week_data["地表温度"], color="red", label="地表温
   度")
130.ax2.set_ylabel("地表温度")
131.
132.fig.legend(loc="upper right", bbox_to_anchor=(1, 1), bbox_transform=ax1.t
   ransAxes)
133.plt.savefig("温度和 O3.png", dpi=150)
134.plt.show()
135.
136.Week_data_once=weather_air.resample("d").mean()["2020-1-1":"2020-3-1"]
137.
138.fig, ax1 = plt.subplots(figsize=(16,8))
139.plt.xticks(rotation=45,fontsize=14)
140.plt.yticks(fontsize=14)
141.
142.
143.ax1.plot(Week_data_once.index.date,Week_data_once["SO2 监测浓度"],"+--
   ",label="SO2 监测浓度",color="cornflowerblue",alpha=0.5,linewidth=3)
144.
145.ax1.set_xlabel("月份")
146.ax1.set_ylabel("O3 小时平均浓度")
147.
148.
```

```
ax2 = ax1.twinx()
ax2.plot(Week_data_once.index.date, Week_data_once["温度"], "o--",label="温度",color="c")
ax2.plot(Week_data_once.index.date, Week_data_once["湿度"], ">--",label="湿度",color="b")
ax2.plot(Week_data_once.index.date, Week_data_once["风向"], "*--",color="darksalmon",label="风向")


ax2.set_ylabel("气象条件")

fig.legend(loc="upper right", bbox_to_anchor=(1, 1), bbox_transform=ax1.transAxes)
plt.savefig("SO2 和其他.png", dpi=150)
plt.show()

Week_data_once=weather_air.resample("d").mean()["2020-1-1":"2020-3-1"]

fig, ax1 = plt.subplots(figsize=(16,8))
plt.xticks(rotation=45,fontsize=14)
plt.yticks(fontsize=14)

ax1.plot(Week_data_once.index.date,Week_data_once["NO2 监测浓度"],"+:",label="NO2 监测浓度",color="cornflowerblue",alpha=0.5,linewidth=3)
ax1.plot(Week_data_once.index.date,Week_data_once["PM10 监测浓度"],"+-.",label="PM10 监测浓度",color="#fd4659",alpha=0.5,linewidth=3)
ax1.plot(Week_data_once.index.date,Week_data_once["PM2.5 监测浓度"],"+-",label="PM2.5 监测浓度",color="#fdaa48",alpha=0.5,linewidth=3)
ax1.set_xlabel("月份")
ax1.set_ylabel("O3 小时平均浓度")


ax2 = ax1.twinx()
ax2.plot(Week_data_once.index.date, Week_data_once["温度"], "o--",label="温度",color="c")
ax2.plot(Week_data_once.index.date, Week_data_once["湿度"], ">--",label="湿度",color="b")
ax2.plot(Week_data_once.index.date, Week_data_once["风向"], "*--",color="darksalmon",label="风向")

ax2.set_ylabel("气象条件")

fig.legend(loc="upper right", bbox_to_anchor=(1, 1), bbox_transform=ax1.transAxes)
```

```
plt.savefig("NO2PM10 和其他.png", dpi=150)
plt.show()
```

问题 3 代码

```
# -*- coding: utf-8 -*-
import torch
import numpy as np
import pandas as pd

from torch.utils.data import Dataset
from sktime.transformations.series.impute import Imputer
from sklearn.model_selection import train_test_split

# 数据预处理
path = './resources/附件 1 监测点 A 空气质量预报基础数据.xlsx'
na_values = ['NAN', 'NA', '—']
df_sheet = pd.read_excel(path, sheet_name='监测点 A 逐日污染物浓度实测数据', na_values=na_values)


def preprocess_data(dataframe, interp='drift', interval=8):
    y = dataframe.copy()

    date = [y['监测日期'].dt.month,
            y['监测日期'].dt.isocalendar().week,
            y['监测日期'].dt.day,
            y['监测日期'].dt.quarter]

    del y['监测日期']
    del y['地点']

    if interp:
        transformer = Imputer(method=interp)
        y = transformer.fit_transform(y, y)

    # 切分时间
    y['月'] = date[0]
    y['周'] = date[1]
    y['日'] = date[2]
    y['季节'] = date[3]

    mean, std = y.mean(axis=0), y.std(axis=0)
```

```
38.     y = y.apply(lambda x: (x - x.mean()) / x.std(), axis=0)
39.
40.     y, y_ = y[:-3], y[-3:] # predict last three day, skip it
41.
42.     return [y[i:i + interval] for i in range(0, len(y) - interval + 1)],
    y_, mean.to_numpy(), std.to_numpy()
43.
44.
45.dataset, time_encodding, mean, std = preprocess_data(df_sheet)
46.train_data, test_data = train_test_split(dataset, test_size=0.2,
    random_state=42)
47.
48.
49.# 组件
50.class MovingAverageLogger:
51.
52.     def __init__(self, beta=0.98):
53.         self.b = beta
54.
55.         self.v = None
56.
57.     def update(self, v):
58.
59.         if self.v is None:
60.             self.v = v
61.         else:
62.             self.v = self.v * self.b + v * (1. - self.b)
63.
64.     def value(self):
65.         return self.v
66.
67.
68.class TimeSeriesDataset(Dataset):
69.
70.     def __init__(self, data):
71.         super(TimeSeriesDataset, self).__init__()
72.         self.data = data
73.
74.     def __len__(self):
75.         return len(self.data)
76.
77.     def __getitem__(self, x):
78.         X = self.data[x].to_numpy().astype(np.float32)[:-1]
79.         Y = self.data[x].to_numpy().astype(np.float32)[-1]
```

```
80.
81.        return torch.from_numpy(X), torch.Tensor(Y[:6])
82.
83.
84.class Baseline(torch.nn.Module):
85.
86.    def __init__(self, in_dim, out_dim, head_dim=64, dropout=0.2):
87.        super(Baseline, self).__init__()
88.
89.        self.bilstm = torch.nn.LSTM(in_dim,
90.                                    head_dim,
91.                                    2,
92.                                    batch_first=True,
93.                                    dropout=dropout,
94.                                    bidirectional=True)
95.
96.        self.mlp = torch.nn.Sequential(
97.            torch.nn.Dropout(dropout),
98.            torch.nn.Linear(head_dim * 2 * 7, head_dim),
99.            torch.nn.ReLU(),
100.            torch.nn.Dropout(dropout),
101.            torch.nn.Linear(head_dim, out_dim)
102.        )
103.
104.    def forward(self, x):
105.        return self.mlp(self.bilstm(x)[0].reshape(len(x), -1))
106.
107.    def forecasting(self, x, z, mean, std):
108.        x = x.to_numpy().astype(np.float32)
109.        z = z.to_numpy().astype(np.float32)[:, 6:]
110.
111.        ys = []
112.        for i in range(len(z)):
113.            X = torch.Tensor([x])
114.            Y = self(X)
115.            ys.append(Y)
116.
117.            y = Y.detach().numpy()[0]
118.            x = np.concatenate([x[1:], np.concatenate([y,
    z[i]])[np.newaxis, :]], axis=0)
119.
120.        return torch.cat(ys, dim=0).detach().numpy() * std[:6] + mean[:6]
121.
122.
```

```
from torch.utils.data import DataLoader

train_dataset = DataLoader(TimeSeriesDataset(train_data),
                           batch_size=16,
                           shuffle=True, drop_last=True)

test_dataset = DataLoader(TimeSeriesDataset(test_data),
                          batch_size=32,
                          shuffle=False)

lr = 3e-4
model = Baseline(10, 6)
optimizer = torch.optim.AdamW(model.parameters(), lr=lr)
criterion = torch.nn.MSELoss()

if __name__ == '__main__':

    from tqdm import tqdm

    for i in range(200):

        model.train()
        logger = MovingAverageLogger()
        with tqdm(total=len(train_dataset), desc=f'Epoch {i}: Training ...') as t:
            for X, y in train_dataset:
                optimizer.zero_grad()
                Y = model(X)
                loss = criterion(Y, y)
                loss.backward()
                optimizer.step()

                logger.update(loss.item())
                t.update()
                t.set_postfix(loss=logger.value())

        model.eval()
        logger = MovingAverageLogger()
        mse = 0
        count = 0
        with tqdm(total=len(test_dataset), desc=f'Epoch {i}: Testing ...') as t:
            for X, y in test_dataset:
                Y = model(X)
```

```
                loss = criterion(Y, y)

                count += len(Y)
                mse += loss.item() * count

                logger.update(loss.item())
                t.update()
                t.set_postfix(loss=logger.value())
        print(mse / count)

        print(model.forecasting(dataset[-1][1:], time_encodding, mean,
    std))
```

问题 4 代码

```
import tensorflow as tf
def placeholder(P, Q, N):
    X = tf.compat.v1.placeholder(shape=(None, P, N), dtype=tf.float32)
    TE = tf.compat.v1.placeholder(shape=(None, P + Q, 2), dtype=tf.int32)
    label = tf.compat.v1.placeholder(shape=(None, Q, N), dtype=tf.float32)
    is_training = tf.compat.v1.placeholder(shape=(), dtype=tf.bool)
    return X, TE, label, is_training

def FC(x, units, activations, bn, bn_decay, is_training, use_bias=True):
    if isinstance(units, int):
        units = [units]
        activations = [activations]
    elif isinstance(units, tuple):
        units = list(units)
        activations = list(activations)
    assert type(units) == list
    for num_unit, activation in zip(units, activations):
        x = tf_utils.conv2d(
            x, output_dims=num_unit, kernel_size=[1, 1], stride=[1, 1],
            padding='VALID', use_bias=use_bias, activation=activation,
            bn=bn, bn_decay=bn_decay, is_training=is_training)
    return x

def STEmbedding(SE, TE, T, D, bn, bn_decay, is_training):
    '''
    spatio-temporal embedding
    SE:     [N, D]
    TE:     [batch_size, P + Q, 2] (dayofweek, timeofday)
    T:      num of time steps in one day
```

```
    D:      output dims
    retrun: [batch_size, P + Q, N, D]
    '''
    # spatial embedding
    SE = tf.expand_dims(tf.expand_dims(SE, axis=0), axis=0)
    SE = FC(
        SE, units=[D, D], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    # temporal embedding
    dayofweek = tf.one_hot(TE[..., 0], depth=7)
    timeofday = tf.one_hot(TE[..., 1], depth=T)
    TE = tf.concat((dayofweek, timeofday), axis=-1)
    TE = tf.expand_dims(TE, axis=2)
    TE = FC(
        TE, units=[D, D], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    return tf.add(SE, TE)

def spatialAttention(X, STE, K, d, bn, bn_decay, is_training):
    '''
    spatial attention mechanism
    X:      [batch_size, num_step, N, D]
    STE:    [batch_size, num_step, N, D]
    K:      number of attention heads
    d:      dimension of each attention outputs
    return: [batch_size, num_step, N, D]
    '''
    D = K * d
    X = tf.concat((X, STE), axis=-1)
    # [batch_size, num_step, N, K * d]
    query = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    key = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    value = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    # [K * batch_size, num_step, N, d]
    query = tf.concat(tf.split(query, K, axis=-1), axis=0)
    key = tf.concat(tf.split(key, K, axis=-1), axis=0)
    value = tf.concat(tf.split(value, K, axis=-1), axis=0)
    # [K * batch_size, num_step, N, N]
```

```
    attention = tf.matmul(query, key, transpose_b=True)
    attention /= (d ** 0.5)
    attention = tf.nn.softmax(attention, axis=-1)
    # [batch_size, num_step, N, D]
    X = tf.matmul(attention, value)
    X = tf.concat(tf.split(X, K, axis=0), axis=-1)
    X = FC(
        X, units=[D, D], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    return X

def temporalAttention(X, STE, K, d, bn, bn_decay, is_training, mask=True):
    '''
    temporal attention mechanism
    X:      [batch_size, num_step, N, D]
    STE:    [batch_size, num_step, N, D]
    K:      number of attention heads
    d:      dimension of each attention outputs
    return: [batch_size, num_step, N, D]
    '''
    D = K * d
    X = tf.concat((X, STE), axis=-1)
    # [batch_size, num_step, N, K * d]
    query = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    key = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    value = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    # [K * batch_size, num_step, N, d]
    query = tf.concat(tf.split(query, K, axis=-1), axis=0)
    key = tf.concat(tf.split(key, K, axis=-1), axis=0)
    value = tf.concat(tf.split(value, K, axis=-1), axis=0)
    # query: [K * batch_size, N, num_step, d]
    # key:   [K * batch_size, N, d, num_step]
    # value: [K * batch_size, N, num_step, d]
    query = tf.transpose(query, perm=(0, 2, 1, 3))
    key = tf.transpose(key, perm=(0, 2, 3, 1))
    value = tf.transpose(value, perm=(0, 2, 1, 3))
    # [K * batch_size, N, num_step, num_step]
    attention = tf.matmul(query, key)
```

```
    attention /= (d ** 0.5)
    # mask attention score
    if mask:
        batch_size = tf.shape(X)[0]
        num_step = X.get_shape()[1].value
        N = X.get_shape()[2].value
        mask = tf.ones(shape=(num_step, num_step))
        mask = tf.linalg.LinearOperatorLowerTriangular(mask).to_dense()
        mask = tf.expand_dims(tf.expand_dims(mask, axis=0), axis=0)
        mask = tf.tile(mask, multiples=(K * batch_size, N, 1, 1))
        mask = tf.cast(mask, dtype=tf.bool)
        attention = tf.compat.v2.where(
            condition=mask, x=attention, y=-2 ** 15 + 1)
    # softmax
    attention = tf.nn.softmax(attention, axis=-1)
    # [batch_size, num_step, N, D]
    X = tf.matmul(attention, value)
    X = tf.transpose(X, perm=(0, 2, 1, 3))
    X = tf.concat(tf.split(X, K, axis=0), axis=-1)
    X = FC(
        X, units=[D, D], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    return X

def gatedFusion(HS, HT, D, bn, bn_decay, is_training):
    '''
    gated fusion
    HS:     [batch_size, num_step, N, D]
    HT:     [batch_size, num_step, N, D]
    D:      output dims
    return: [batch_size, num_step, N, D]
    '''
    XS = FC(
        HS, units=D, activations=None,
        bn=bn, bn_decay=bn_decay,
        is_training=is_training, use_bias=False)
    XT = FC(
        HT, units=D, activations=None,
        bn=bn, bn_decay=bn_decay,
        is_training=is_training, use_bias=True)
    z = tf.nn.sigmoid(tf.add(XS, XT))
    H = tf.add(tf.multiply(z, HS), tf.multiply(1 - z, HT))
    H = FC(
        H, units=[D, D], activations=[tf.nn.relu, None],
```

```
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    return H

def STAttBlock(X, STE, K, d, bn, bn_decay, is_training, mask=False):
    HS = spatialAttention(X, STE, K, d, bn, bn_decay, is_training)
    HT = temporalAttention(X, STE, K, d, bn, bn_decay, is_training,
   mask=mask)
    H = gatedFusion(HS, HT, K * d, bn, bn_decay, is_training)
    return tf.add(X, H)

def transformAttention(X, STE_P, STE_Q, K, d, bn, bn_decay, is_training):
    '''
    transform attention mechanism
    X:      [batch_size, P, N, D]
    STE_P:  [batch_size, P, N, D]
    STE_Q:  [batch_size, Q, N, D]
    K:      number of attention heads
    d:      dimension of each attention outputs
    return: [batch_size, Q, N, D]
    '''
    D = K * d
    # query: [batch_size, Q, N, K * d]
    # key:   [batch_size, P, N, K * d]
    # value: [batch_size, P, N, K * d]
    query = FC(
        STE_Q, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    key = FC(
        STE_P, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    value = FC(
        X, units=D, activations=tf.nn.relu,
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    # query: [K * batch_size, Q, N, d]
    # key:   [K * batch_size, P, N, d]
    # value: [K * batch_size, P, N, d]
    query = tf.concat(tf.split(query, K, axis=-1), axis=0)
    key = tf.concat(tf.split(key, K, axis=-1), axis=0)
    value = tf.concat(tf.split(value, K, axis=-1), axis=0)
    # query: [K * batch_size, N, Q, d]
    # key:   [K * batch_size, N, d, P]
    # value: [K * batch_size, N, P, d]
    query = tf.transpose(query, perm=(0, 2, 1, 3))
    key = tf.transpose(key, perm=(0, 2, 3, 1))
```

```
    value = tf.transpose(value, perm=(0, 2, 1, 3))
    # [K * batch_size, N, Q, P]
    attention = tf.matmul(query, key)
    attention /= (d ** 0.5)
    attention = tf.nn.softmax(attention, axis=-1)
    # [batch_size, Q, N, D]
    X = tf.matmul(attention, value)
    X = tf.transpose(X, perm=(0, 2, 1, 3))
    X = tf.concat(tf.split(X, K, axis=0), axis=-1)
    X = FC(
        X, units=[D, D], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    return X

def GMAN(X, TE, SE, P, Q, T, L, K, d, bn, bn_decay, is_training):
    '''
    GMAN
    X:        [batch_size, P, N]
    TE:       [batch_size, P + Q, 2] (time-of-day, day-of-week)
    SE:       [N, K * d]
    P:        number of history steps
    Q:        number of prediction steps
    T:        one day is divided into T steps
    L:        number of STAtt blocks in the encoder/decoder
    K:        number of attention heads
    d:        dimension of each attention head outputs
    return:   [batch_size, Q, N]
    '''
    D = K * d
    # input
    X = tf.expand_dims(X, axis=-1)
    X = FC(
        X, units=[D, D], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    # STE
    STE = STEmbedding(SE, TE, T, D, bn, bn_decay, is_training)
    STE_P = STE[:, : P]
    STE_Q = STE[:, P:]
    # encoder
    for _ in range(L):
        X = STAttBlock(X, STE_P, K, d, bn, bn_decay, is_training)
    # transAtt
    X = transformAttention(
        X, STE_P, STE_Q, K, d, bn, bn_decay, is_training)
```

```
    # decoder
    for _ in range(L):
        X = STAttBlock(X, STE_Q, K, d, bn, bn_decay, is_training)
    # output
    X = FC(
        X, units=[D, 1], activations=[tf.nn.relu, None],
        bn=bn, bn_decay=bn_decay, is_training=is_training)
    return tf.squeeze(X, axis=3)

def mae_loss(pred, label):
    mask = tf.not_equal(label, 0)
    mask = tf.cast(mask, tf.float32)
    mask /= tf.reduce_mean(mask)
    mask = tf.compat.v2.where(
        condition=tf.math.is_nan(mask), x=0., y=mask)
    loss = tf.abs(tf.subtract(pred, label))
    loss *= mask
    loss = tf.compat.v2.where(
        condition=tf.math.is_nan(loss), x=0., y=loss)
    loss = tf.reduce_mean(loss)
    return loss
```