实验报告：基于LSTM、Transformer与TemporalConvNet的单车租赁数量预测

1. 问题介绍

随着共享单车的普及，准确预测单车的租赁数量对于城市交通管理和单车调度至关重要。单车租赁数量的预测是一个典型的时间序列预测问题，需要考虑时间的动态变化以及环境因素（如气温、湿度、工作日等）。本实验的目标是使用三种不同的模型（LSTM、Transformer、TemporalConvNet）来预测未来一段时间内的单车租赁数量，并通过实验评估它们在这一任务中的表现。[1]

预测任务：根据所提供的数据对未来单车租赁数量（cnt）进行预测。基于过去 I=96小时的数据曲线来预测未来（i）O=96 小时（短期预测）和（ii）O=240 小时（长期预测）两种长度的变化曲线（需要分别训练，即长期预测的模型参数不能用于短期预测）。按照方法分为三部分。

前两部分为基础题，第三部分为开放题，各占总分的三分之一：

1. 使用 LSTM 模型进行预测；
2. 使用 Transformer 模型进行预测；
3. 使用自己提出的改进模型进行预测，结构不限。此部分以原理的新颖程度为首要评价标准，性能为次要评价标准。

训练与测试：数据集主要分为 train 和 test 两部分（具体见文件“train_data.csv”和“test_data.csv”）。请使用两种评价标准进行测试，即均方误差（MSE）与平均绝对误差（MAE）。至少进行五轮实验，并对结果取平均值，同时提供标准差（std）以评估结果的稳定性。

目标设置：

- 基于过去96小时的租赁数量以及环境特征（如温度、湿度等），预测未来96小时（短期）和240小时（长期）的单车租赁数量。
- 对比LSTM、Transformer和TiDE模型的预测性能，分析它们的优缺点。

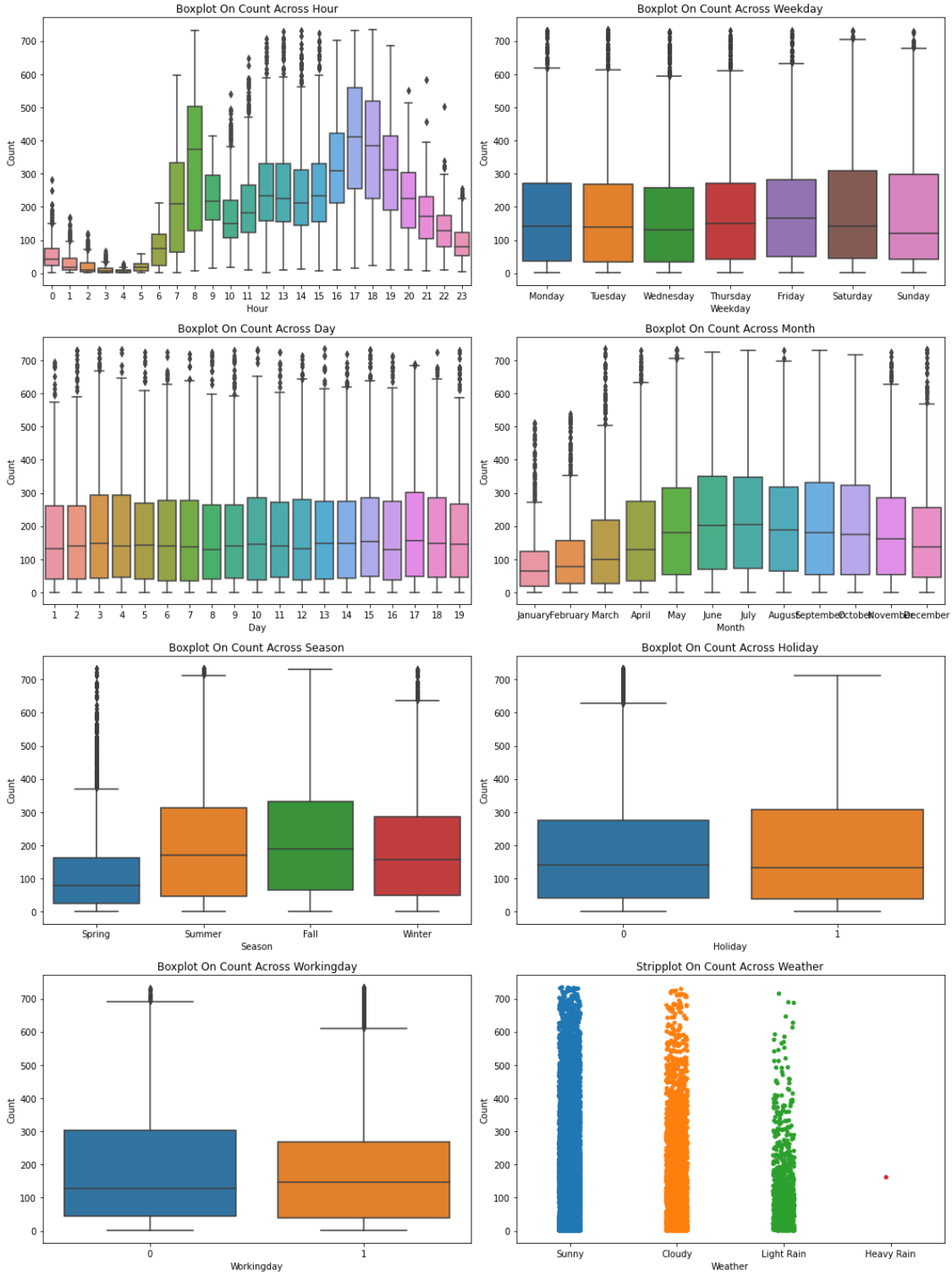
任务分工：

- 小组成员：
    - 周梓轩：本科为数据科学与大数据技术专业，研究生目前在NLP实验室研究机器翻译方向，主要是使用深度学习的方法进行自然语言之间的映射，实现源语言到目标语言的高效翻译。主要的挑战是处理复杂的语言结构、上下文依赖以及不同语言之间的语义差异，尤其是在专业领域或特定语境下。会先使用神经网络模型（如Transformer）进行语义级的翻译，再结合后处理方法（如译后编辑）优化翻译结果。近期研究深入大模型方向，结合提示词工程对中小模型进行蒸馏提高翻译性能。
    - 黄汉滨：医学图像分割方向，主要是使用深度学习的方法对病理图像中的细胞核实例进行分割。主要的挑战是核分布密集且形状各异，与背景环境的差异也很小。会先使用CV的方法分割出语义结果再经过非深度学习的方法（如与距离图的结果结合）得到实例结果
- 具体任务：
    - 数据分析和处理主要由两人商量好数据分析需求，由周梓轩具体执行。
    - 第一题LSTM主要是周梓轩负责预测，黄汉滨提供调参idea。
    - 第二题Transformer主要是黄汉滨负责预测，周梓轩分析结果及反馈训练结果。
    - 第三题为二人讨论后共同实施，具体思路见实验报告，其中TCN模型是黄汉滨找到并实验效果良好，周梓轩则是结合第一二题分析结果提出优化方向。
    - 实验过程两人讨论完成，认真分析，保质保量，分别各自撰写实验报告，敬请查阅。

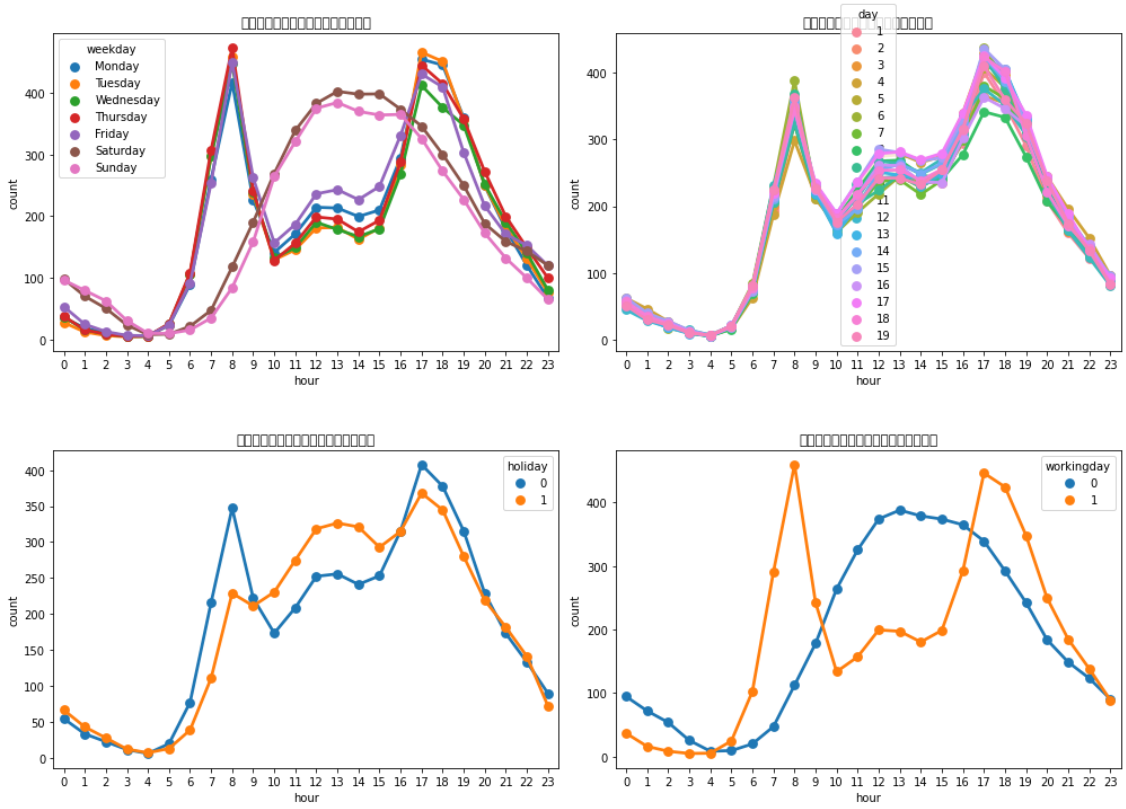
2. 数据处理（部分介绍，详情见github)

数据处理，部分参考csdn，链接如下：机器学习之共享单车骑行量预测

- **数据加载**：
  - 从CSV文件中加载训练和测试数据。
  - 对数据进行独热编码，处理假日（holiday）、工作日（workingday）和天气状况（weathersit）等分类特征。
  - 处理缺失值，确保所有特征列都存在，缺失的列补0。
- **特征选择**：
  - 选择温度（temp）、体感温度（atemp）、湿度（hum）、风速（windspeed）以及独热编码后的特征作为输入特征。
  - 选择临时用户（casual）和注册用户（registered）的租赁数量作为目标变量。
- **滑动窗口**：
  - 使用滑动窗口生成输入序列和目标序列。每个输入序列的长度为96个时间步，目标序列的长度为96个时间步（短期预测）或240个时间步（长期预测）。
- **数据分析过程**：
  - **首先对8个类别特征进行可视化**[2]
  - **不同时间段对使用量的影响(hour → count)** \ 图1：在一天中不同时间段，共享单车使用量差异明显，在8点和16-19点明显多于其他时间点，考虑到的原因是在这期间是上下班的高峰期；在0-5点明显低于其他时间点，考虑到的原因是在此期间为睡眠时间。
  - **星期几对使用量的影响(weekday → count)** \ 图2：星期几对单车总使用量没有太大的影响。
  - **一个月内的哪一天对使用量的影响(day → count)** \ 图3：一个月的哪一天对单车总使用量也没有太大的影响。
  - **月份对使用量的影响(month → count)** \ 图4：11月-4月的共享单车使用量会比其他月份少一点，可能是季节原因，冬季和春季太冷导致使用量降低。所以接下来我们观察一下季节对使用量的影响进行验证。
  - **季节对使用量的影响(season → count)** \ 图5：可以看到冬春相对夏秋使用量相对较少，与上面月份产生的结论相互印证。
  - **节假日对使用量的影响(holiday → count)** \ 图6：是否节假日对单车总使用量基本没有太大的影响。
  - **工作日对使用量的影响(workingday → count)** \ 图7：是否工作日对单车总使用量也基本没有太大的影响。
  - **天气对使用量的影响(weather → count)** \ 图8：天气对单车的影响基本符合日常生活中的实际情况，下雨天单车使用量减少，下暴雨时基本没人使用共享单车。



- **其次，两字段结合可视化分析**
  - **星期对不同时间段单车使用量的影响(hour + weekday → count)** \ 图1：与周一至周五相比，周六周天对不同时间段单车使用量的影响明显不同。
  - **日期对不同时间段单车使用量的影响(hour + day → count)** \ 图2：不同日期对不同时间段单车使用量的影响，在两个峰值之间表现出一定差异。
  - **节假日对不同时间段单车使用量的影响(hour + holiday → count)** \ 图3：相比于节假日，非节假日在上班和下班高峰期表现出更大的单车使用量。
  - **工作日对不同时间段单车使用量的影响(hour + workingday → count)** \ 图4：同样地，相比于非工作日，工作日在上班和下班高峰期表现出更大的单车使用量；而非工作日11点-17点的单车使用量更高，猜测有可能是出门吃饭或游玩。[3]



- **注册用户和非注册用户对使用量的影响(registered → count & casual → count)** \ 由下图可以得知，注册用户的使用量占据了单车总使用量的绝大多数，并且趋势与总使用量趋势一致，而非注册用户，一天中不同时段的使用量没有太大变化，这说明使用量主要注册用户决定，注册用户更有粘性。
- 由此我们产生了解决本题的思路：

选择分别预测注册用户（registered）和临时用户（casual）的租赁数量，并将它们的总和作为总租赁数量（cnt）的预测，是因为注册用户和临时用户的行为模式存在显著差异。注册用户的行为通常更稳定，受工作日、节假日等因素影响较大；而临时用户的行为则更受天气、节假日等外部因素影响，较为随机。

从结果来看这种思路的优势在于：

1. 保留了数据的完整性

- **详细信息**：注册用户和临时用户的数据提供了更详细的信息。注册用户通常是经常使用自行车租赁服务的用户，而临时用户可能是偶尔使用服务的用户。这两类用户的行为模式可能不同，通过分别预测它们的租赁数量，可以更全面地捕捉数据的特征。
- **总租赁数量**：总租赁数量（cnt）是注册用户和临时用户租赁数量的总和。通过分别预测这两类用户，可以更准确地预测总租赁数量，因为总租赁数量是这两类用户行为的综合结果。

2. 可以分析出行为模式的差异

- **注册用户**：注册用户的行为通常更稳定，因为他们是经常使用服务的用户。他们的租赁行为可能受到工作日、节假日、天气等因素的影响，但相对较为规律。
- **临时用户**：临时用户的行为可能更受天气、季节、特殊事件等因素的影响。他们的租赁行为可能更加随机和不规律。
- **综合考虑**：通过分别预测注册用户和临时用户的租赁数量，可以更好地捕捉这两类用户的不同行为模式，从而提高总租赁数量预测的准确性。

3.具体结果体现

- 结果中反应来看，注册用户的使用更加稳定，数据拟合曲线更符合实际，但是会出现略低。而未注册用户的使用更加具有随机性，因此拟合数据往往超出实际情况很多。所以二者可以进行调整，带权重进行**合并，综合的曲线会更切合真实数据。**

3. 数据集定义

- **BikeDataset**：
  - 定义一个自定义的`Dataset`类，用于加载和处理数据。
  - `__getitem__`方法返回输入序列、临时用户目标和注册用户目标。

4. 模型

LSTM模型：

LSTM（Long Short-Term Memory）是一种特殊的RNN（循环神经网络），能够捕捉序列数据中的长期依赖性，适合用于时间序列预测。LSTM在处理长期依赖的序列数据时比传统的RNN更具优势。

本题在处理中，分开预测了注册和用户和临时用户的

操作流程如下

- 定义一个LSTM模型，包含LSTM层和线性解码器。
- 输入层：包括时间特征、气温、湿度、工作日等。
- LSTM层的输入维度为特征数量，隐藏维度为64，层数为2，dropout率为0.1。
- 线性解码器的输出维度为预测长度（96或240）。
- 模型的前向传播使用最后一个时间步的输出进行预测。
- 输出层：预测未来单车租赁数量，输出节点数量为1。

训练时使用Adam优化器，并设置学习率为0.001，训练周期为80epoch，批量大小为32。

LSTM模型伪代码：

```
class LSTMModel(nn.Module):
    def __init__(self, input_dim, pred_len, hidden_dim=64, num_layers=2,
dropout=0.1):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers, batch_first=True,
dropout=dropout)
        self.decoder = nn.Linear(hidden_dim, pred_len)

    def forward(self, x):
        lstm_out, (h_n, c_n) = self.lstm(x)
        out = self.decoder(lstm_out[:, -1, :])  # 只使用最后一个时间步的输出进行预测
        return out
```

Transformer模型：

Transformer是一种基于自注意力机制（Self-Attention）的模型，具有并行计算的优势，特别适用于处理大规模的序列数据。Transformer能够捕捉序列中任意位置的关系，因此具有较强的建模能力。

操作流程如下

- 定义一个Transformer模型，包含编码器、位置编码、Transformer层和解码器。
- 输入层：包括时间特征、气温、湿度、工作日等。
- 编码器将输入特征映射到模型维度。
- 位置编码为模型提供序列中每个元素的位置信息。
- Transformer层包含多个编码器层，通过自注意力机制捕捉序列中的长距离依赖关系。
- 解码器将Transformer层的输出映射到预测长度。
- 输出层：预测未来单车租赁数量，输出节点数量为预测长度（96或240）。

训练时使用Adam优化器，并设置学习率为0.001，训练周期为100epoch，批量大小为32。

Transformer模型伪代码：

```
class TransformerModel(nn.Module):
    def __init__(self, input_dim, pred_len, d_model=64, nhead=4, num_layers=2,
dim_feedforward=128, dropout=0.1):
        super(TransformerModel, self).__init__()
        self.encoder = nn.Linear(input_dim, d_model)
        self.positional_encoding = nn.Parameter(torch.zeros(1, SEQ_LEN,
d_model))
        self.transformer = nn.Transformer(d_model, nhead, num_layers,
num_layers, dim_feedforward, dropout)
        self.decoder = nn.Linear(d_model, pred_len)

    def forward(self, x):
        x = self.encoder(x) + self.positional_encoding
        x = x.permute(1, 0, 2)  # Transformer expects (seq_len, batch_size,
d_model)
        x = self.transformer(x, x)
        x = x.permute(1, 0, 2)  # Back to (batch_size, seq_len, d_model)
        x = self.decoder(x[:, -1, :])  # Decode the last time step
        return x
```

TemporalConvNet+LSTM+Transformer：

模型思路：TCN + LSTM + Transformer

在前面的结果中，我们发现图表结果中，LSTM在预测注册人员的情况下曲线更贴合，而transformer在预测未注册人员的情况下更贴合，因此我们打算分别用LSTM和transformer预测不同的情况，并希望加入一个模块使得二者在对应领域的预测更加准确，从而使得预测的cnt结果更加准确。

为什么选择TCN：

在前面的LSTM和Transformer分别训练并预测的结果中，我们发现，尽管LSTM能够有效捕捉长时序依赖，且Transformer在全局建模方面表现出色，但两者都在处理序列中的局部模式时存在一定局限性。LSTM虽然能够捕获长依赖关系，但在时间序列的局部特征提取上可能不如卷积神经网络（CNN）高效。而Transformer虽然能够自适应地捕捉全局依赖关系，但在处理较短的时间窗口时，可能面临计算

效率较低的问题，尤其是当输入序列长度较长时，计算复杂度会显著增加。[4]

因此，选择 **TCN（Temporal Convolutional Network）** 是因为它能在捕捉时间序列的局部依赖和长时间依赖方面提供平衡。TCN通过因果卷积和扩张卷积的结构，能够在较低的计算成本下捕捉不同时间尺度的模式，特别适用于时间序列数据。与传统的LSTM相比，TCN能够并行处理整个序列，避免了LSTM的递归计算瓶颈。此外，TCN在捕捉局部时序特征方面比LSTM更加高效，因此能够为后续的LSTM和Transformer模型提供更精确的输入，提升整个模型的预测性能。

操作流程如下：

- **输入特征：** 包括时间特征（如小时、日期等）、环境特征（如气温、湿度）、以及工作日、假期等信息。通过预处理将这些特征组合成一个时间序列数据集。
- **TCN层：** 首先使用TCN网络提取时间序列中的局部特征。TCN通过因果卷积（Causal Convolution）和扩张卷积（Dilated Convolution）捕捉长距离的依赖关系，适合捕捉短期波动和规律。TCN可以高效地提取序列中各个时间步之间的依赖关系，并避免了RNN等传统方法中的时间步长问题。
- **LSTM层：** LSTM能够有效地捕捉长期依赖关系，尤其是在有较大时间跨度的序列预测中，LSTM能够记住长期记忆并学习时间序列中的长期趋势。LSTM层的输出用于短期预测。
- **Transformer层：** Transformer通过自注意力机制（Self-Attention）能够捕捉到时间序列中的全局依赖关系，尤其是能够通过多头注意力捕捉到不同时间步之间的交互特征，从而进一步增强模型对全局依赖的建模能力。Transformer层的输出用于长期预测。
- **一致性损失（Consistency Loss）：** 使用一致性损失约束TCN+LSTM的短期预测结果和TCN+Transformer的长期预测结果，确保不同模型的预测具有一致性。通过L1损失计算不同模型预测值之间的差异，若误差小于阈值`delta`，使用平方损失；否则使用线性损失。一致性损失加入总损失函数，通过反向传播优化模型参数。
- **输出层：** 最后，使用两个全连接层（Decoder）分别将TCN+LSTM和TCN+Transformer的输出映射为未来一段时间的预测结果。短期预测的输出节点数量为96小时，长期预测的输出节点数量为240小时。

训练时使用Adam优化器，并设置学习率为0.001，训练周期为100epoch，批量大小为512。

伪代码：[5]

```
class TCN_LSTM_Transformer(nn.Module):
    def __init__(self, input_dim, pred_len, d_model=64, nhead=4, num_layers=2,
dim_feedforward=128, dropout=0.1):
        super(TCN_LSTM_Transformer, self).__init__()

        # 初始化TCN层，提取局部时间特征
        self.tcn = TemporalConvNet(input_dim, num_channels=[d_model, d_model*2,
d_model*4])

        # 初始化LSTM层，建模长时依赖
        self.lstm = nn.LSTM(d_model*4, d_model, batch_first=True)

        # 初始化Transformer层，建模全局依赖
        self.transformer = nn.Transformer(d_model, nhead, num_layers,
num_layers, dim_feedforward, dropout)

        # 初始化解码层，将Transformer输出映射到预测长度
        self.decoder = nn.Linear(d_model, pred_len)

    def forward(self, x):
        # TCN提取局部时序特征
        x_tcn = self.tcn(x)
```

```
        # LSTM处理TCN输出，建模长期依赖
        x_lstm, _ = self.lstm(x_tcn)

        # Transformer处理LSTM输出，建模全局依赖
        x_transformer = x_lstm.permute(1, 0, 2)  # 转换形状为 (seq_len,
batch_size, d_model)
        x_transformer = self.transformer(x_transformer, x_transformer)
        x_transformer = x_transformer.permute(1, 0, 2)  # 恢复形状为 (batch_size,
seq_len, d_model)

        # 只输出最后一个时间步的预测结果
        x_pred = self.decoder(x_transformer[:, -1, :])  # 只解码最后时间步的输出
        return x_pred
```

5. 其他函数

- **train_and_evaluate**：
  - 定义训练和评估函数，使用均方误差（MSE）作为损失函数。
  - 训练过程中，模型在每个epoch结束时计算训练损失和测试损失。
  - 评估过程中，计算预测值和真实值的均方误差（MSE）和平均绝对误差（MAE）。
  - 记录每个epoch的预测结果和真实值，用于后续可视化。
- **plot_prediction**：
  - 定义一个函数，用于绘制原始数据、预测值和真实值的对比图。
  - 保存图表到指定路径。
- **main**：
  - 加载训练和测试数据。
  - 创建数据加载器（DataLoader）。
  - 初始化LSTM模型、损失函数和优化器。
  - 进行多次实验（5次），记录每次实验的MSE和MAE。
  - 选择最佳的预测结果进行可视化。
  - 计算并输出最终的MSE和MAE的均值和标准差。

6. 实验结果与分析

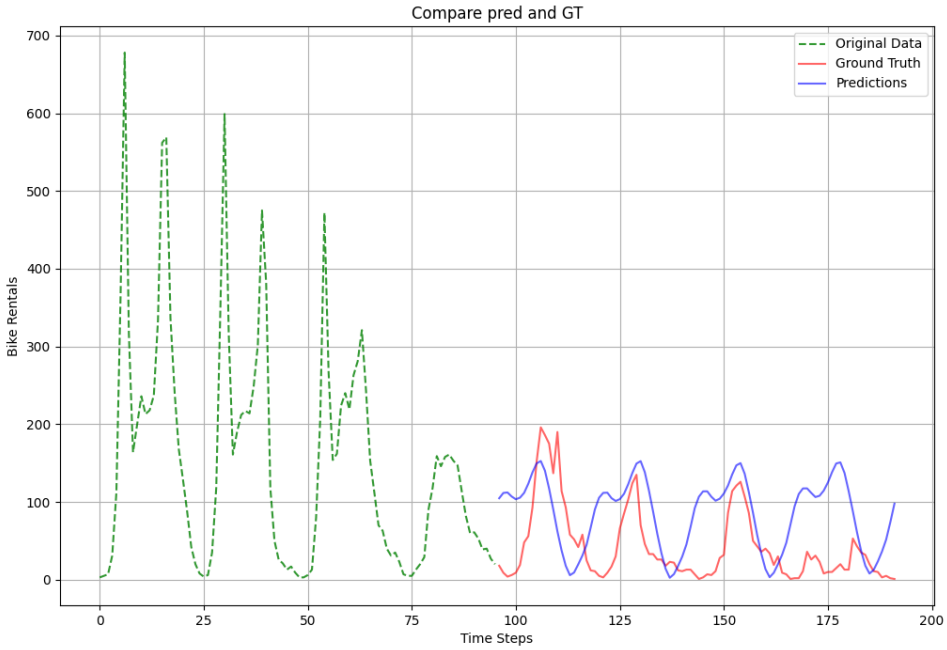
- **实验设置**
  - **评价标准**：使用均方误差（MSE）与平均绝对误差（MAE）作为主要评价指标。
  - **实验轮次**：每个模型分别进行5轮实验，记录每轮的MSE和MAE，最后计算平均值和标准差。
- **实验结果：**

在训练和测试过程中，LSTM、Transformer和TiDE模型在短期（96小时）预测上表现较为相似，但在长期（240小时）预测上，Transformer模型和TiDE模型表现优于LSTM模型，尤其是在捕捉长期趋势方面。

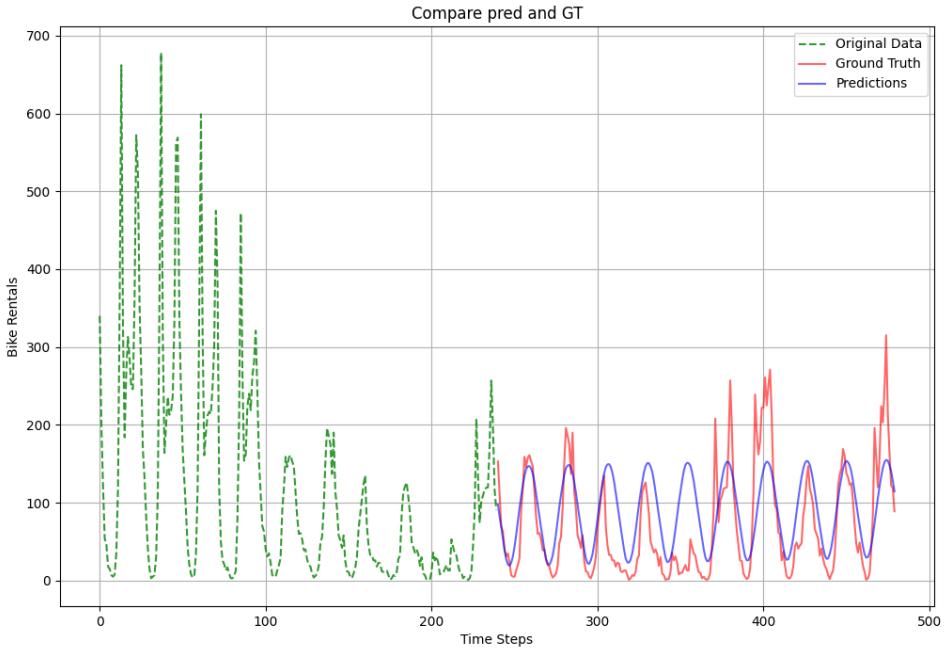
- **预测与真实值的对比图：**

以下是LSTM、Transformer和TiDE模型对未来96小时和240小时单车租赁数量的预测结果与真实值的对比图。

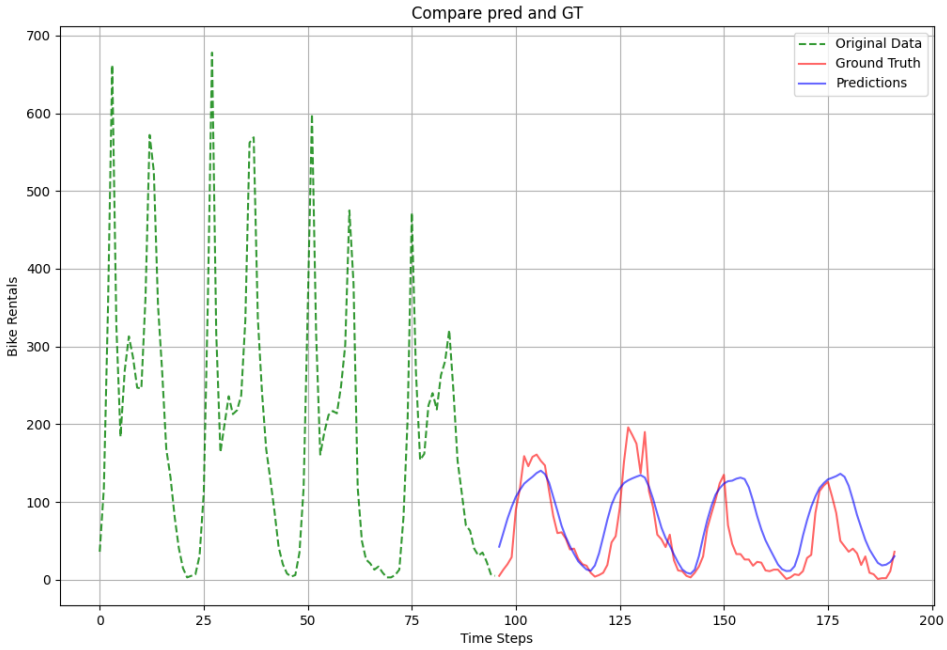
- **LSTM短期预测（96小时）：**



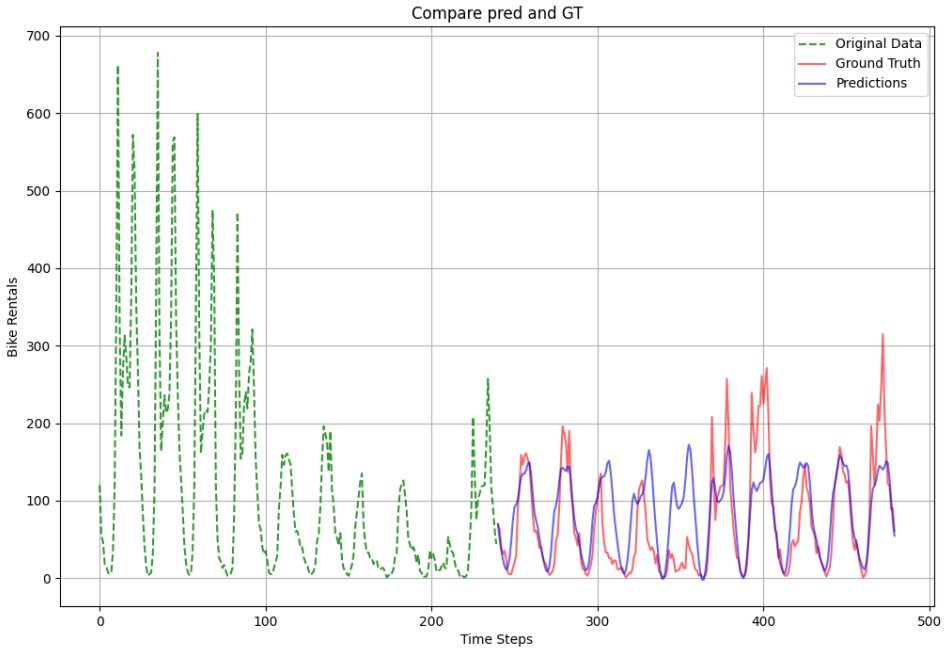
- **LSTM长期预测（240小时）：**



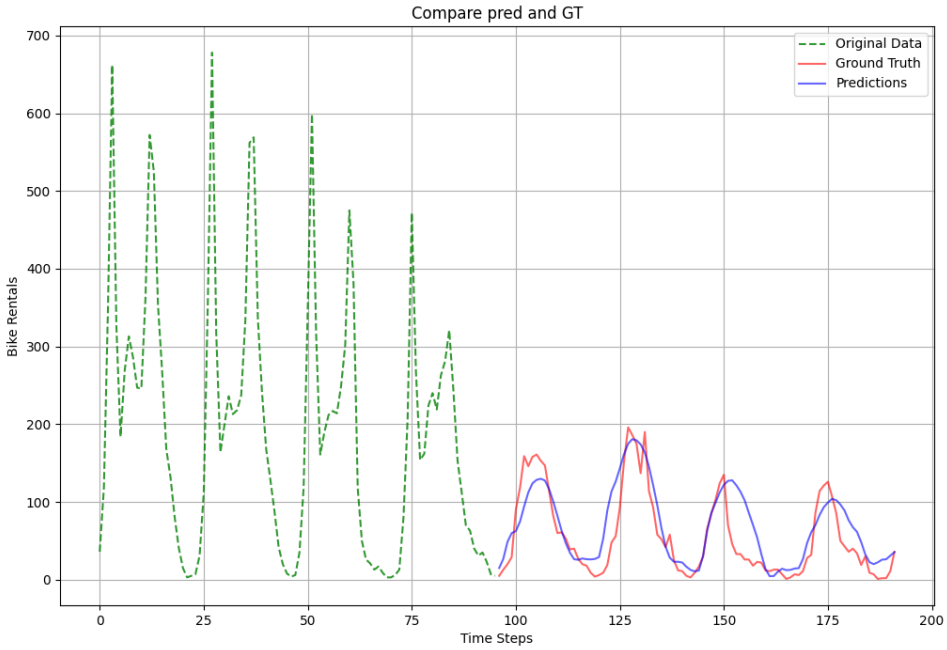
- **Transformer短期预测（96小时）**：



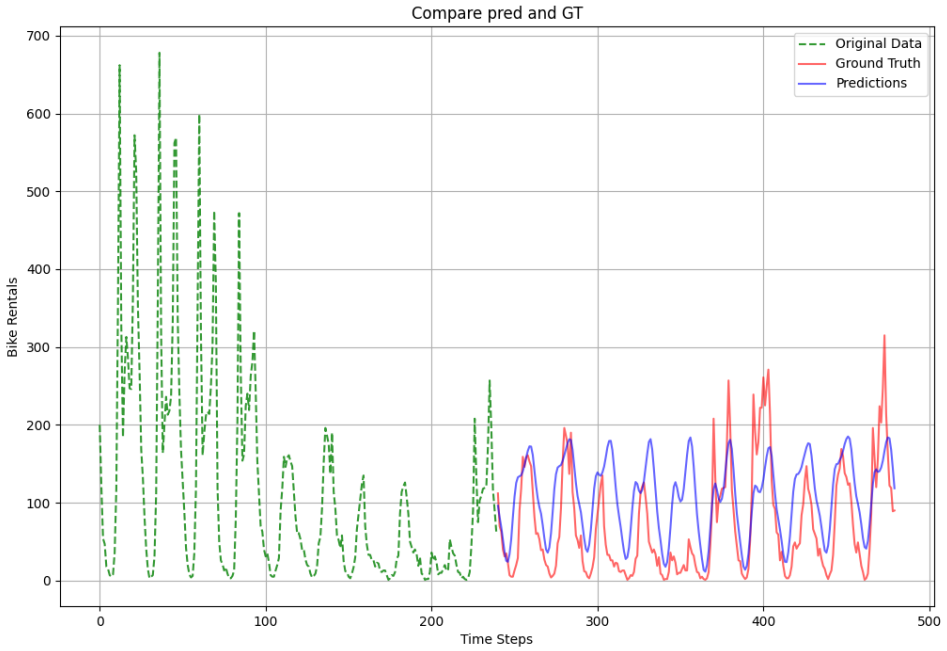
- **Transformer长期预测（240小时）**：



- **TemporalConvNet+LSTM短期预测（96小时）：**



- **TemporalConvNet+Transformer长期预测（240小时）：**



模型	*MSE*（短期预测）	*MAE*（短期预测）	*MSE*（长期预测）	*MAE*（长期预测）
LSTM	30747.6621 (±1238.2142)	117.1020 (±3.1278)	37331.0938 (±667.6597)	129.5617 (±2.2192)
Transformer	27410.0977 (±3353.7104)	111.8838 (±6.2045)	33437.7891 (±1478.0574)	125.4707 (±2.7250)
TemporalConvNet+LSTM+Transformer	26734.6816 (±3725.2563)	111.4200 (±7.8727)	33166.8164 (±2774.1555)	126.6419 (±5.6809)

从图中可以看出：

- **短期预测**：三种模型在短期预测中表现较好，预测值与真实值接近，误差较小。MSE和MAE是Transformer和TCN+LSTM模型指标较好，但是LSTM和Transformer预测过程中，预测曲线变化较小，不能自适应真实数据变化，TCN+LSTM的自适应能力更强，可以考虑提高TCN权重。
- **长期预测**：在长期预测中，Transformer和TCN+Transformer模型的预测结果更接近真实值，而LSTM的预测误差较大，同时TCN+Transformer模型的峰值变化更贴近真实值的变化趋势，TCN+Transformer的自适应能力更强，可以考虑提高TCN权重。

实验结果表明第三题提出的方法有改进效果，考虑的下一步就是提高TCN的计算权重以及模型训练参数的细节调整。

在结合了过高的未注册人员预测结果和略低的注册人员结果后发现结果明显接近真实值。证明本实验中我们提出的人员区分和模型改进方法思路均有效果。

7.代码提交

本次实验的代码可以通过以下 GitHub 链接进行访问：

- https://github.com/zhouzixuan2333/gxdc.git

8.参考文献

1. Das A, Kong W, Leach A, et al. Long-term forecasting with tide: Time-series dense encoder[J]. arXiv preprint arXiv:2304.08424, 2023.
2. Bike Sharing Demand共享单车需求预测
3. L1_Homework_用线性回归预测共享单车的需求 (kaggle.com)
4. locuslab/TCN: Sequence modeling benchmarks and temporal convolutional networks (github.com)
5. Ma C Y, Chen M H, Kira Z, et al. TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition[J]. Signal Processing: Image Communication, 2019, 71: 76-87.