

2025 年专硕机器学习

June 2025

1 问题介绍

随着智能家居和物联网技术的快速发展，家庭用电监测与管理在节能减排、降低用电成本以及实现智能化生活方面扮演着越来越重要的角色。家庭电力消耗数据不仅能够帮助居民更好地理解和调整自身的用电行为，提升能源利用效率，还能为电力公司提供科学依据，以实现电力负荷的合理调度和动态管理，促进智能电网的发展。

家庭电力消耗具有复杂的时序特性，受季节变化、节假日、家庭成员的生活习惯、用电设备类型以及气象条件等多种因素的影响。这些多因素的交互作用使得电力消耗数据表现出明显的非线性和多变量相关性，给准确的电力负荷预测带来了极大的挑战。基于此背景，利用多变量时间序列模型对家庭总有功功率进行精确预测，具有重要的现实意义。本研究以法国一户家庭的电力消耗公开数据集（UCI Machine Learning Repository 中“Individual household electric power consumption” 数据集）为基础，结合气象等外部因素，构建多变量时间序列预测模型。具体任务是基于过去 90 天的多维用电及气象数据，分别预测未来 90 天（短期预测）和 365 天（长期预测）的每日总有功功率变化趋势。准确的预测不仅有助于用户合理安排用电计划，降低峰值负荷，还能推动智能电网的动态调度和可再生能源的高效接入。

本文将分别采用 LSTM 和 Transformer 模型进行预测，并尝试对 Transformer 进行改进，旨在提升长短期预测的准确性和稳定性。通过对比不同模型的预测效果，探讨多变量时间序列建模在家庭用电负荷预测中的应用价值和潜力。

相关源码可参考 GitHub¹。

¹<https://github.com/CharyChung/Machine-Learning>

2 模型

2.1 LSTM

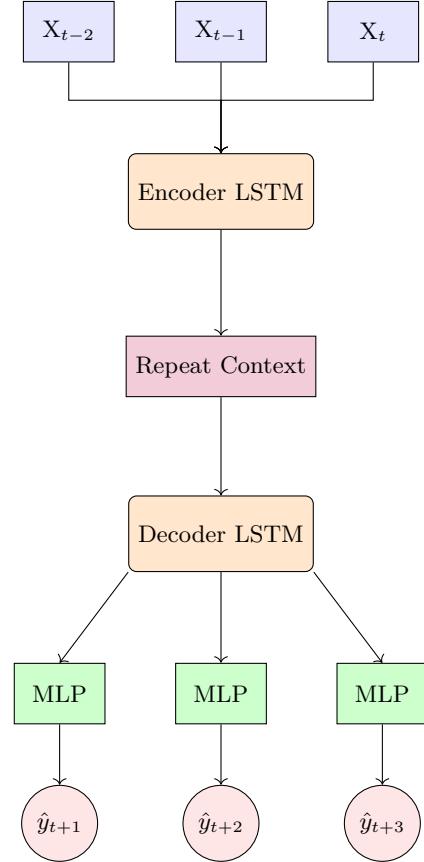
2.1.1 模型结构概述

本模型采用 Encoder-Decoder 结构的长短期记忆网络 (LSTM) 进行多变量多步时间序列预测，目标是根据过去 90 天的电力与气象数据，预测未来 90 天或 365 天的家庭总有功功率 (global_active_power)。

模型结构主要包括以下几个部分：

- **Encoder:** 使用单层 LSTM 编码输入的多变量时间序列，提取其时间依赖特征，并输出最后一个时间步的隐状态 (hidden state)。
- **重复层 (Repeat Layer):** 通过全连接层 (Linear) 将隐状态重复并扩展至输出序列长度 (n_{output})，作为 Decoder 的输入。
- **Decoder:** 使用 LSTM 解码重复后的序列，生成对应长度的隐藏序列。
- **输出层 (Output Projection):** 每个时间步的输出由一个包含非线性激活 (ReLU) 的全连接网络进行回归，预测未来每一天的目标值。

2.1.2 模型结构图示意



2.1.3 模型优点与局限

该模型在多变量时间序列建模任务中表现出良好的适应性。首先，它采用 Encoder-Decoder 架构，支持变长输入与输出，适合多步预测场景。LSTM 编码器能够捕捉输入序列的长期依赖特征，解码器在重复隐状态的基础上生成目标序列，每个时间步通过非线性 MLP 网络独立预测，从而提升了表达能力。同时，该结构设计简洁，训练过程稳定，适合中小规模数据建模。

然而，该模型也存在一定局限性。由于仅使用编码器的最终隐状态作为上下文向量，可能无法充分捕捉整个历史信息，尤其在序列较长或包含季节性成分时易丢失信息。在滚动预测过程中，前一时间步的预测结果被作为输入滑动使用，这种方式可能导致误差逐步积累。此外，面对具有强周期性、长趋势的数据时，模型的泛化能力仍有限，需结合注意力机制或位置编码等

机制进一步优化。

2.1.4 模型伪代码

Algorithm 1: Encoder-Decoder LSTM Model (Detailed)

Input: Multivariate input sequence $X \in \mathbb{R}^{n_{\text{input}} \times d}$

Output: Predicted output sequence $\hat{y} \in \mathbb{R}^{n_{\text{output}}}$

Encoding:

Initialize hidden state h_0 and cell state c_0 for encoder LSTM

Pass input sequence X through encoder LSTM:

$$h_{\text{enc}}, c_{\text{enc}} \leftarrow \text{LSTM}_{\text{enc}}(X, h_0, c_0)$$

Obtain final hidden state h_{enc} as context vector

Repeat context:

Use Linear layer to expand h_{enc} into sequence H_{rep} :

$$H_{\text{rep}} \in \mathbb{R}^{n_{\text{output}} \times \text{hidden_dim}}$$

Each time step shares the same content of h_{enc}

Decoding:

Initialize decoder LSTM with hidden state h_{enc} and cell state

$$c_{\text{enc}}$$

Pass H_{rep} through decoder LSTM:

$$H_{\text{dec}} \leftarrow \text{LSTM}_{\text{dec}}(H_{\text{rep}}, h_{\text{enc}}, c_{\text{enc}})$$

Output Layer:

For each time step t in H_{dec} :

Pass $H_{\text{dec}}[t]$ through MLP:

$$\hat{y}_t = \text{MLP}(H_{\text{dec}}[t]) \in \mathbb{R}^1$$

Stack \hat{y}_t to form output sequence $\hat{y} \in \mathbb{R}^{n_{\text{output}}}$

2.2 Transformer

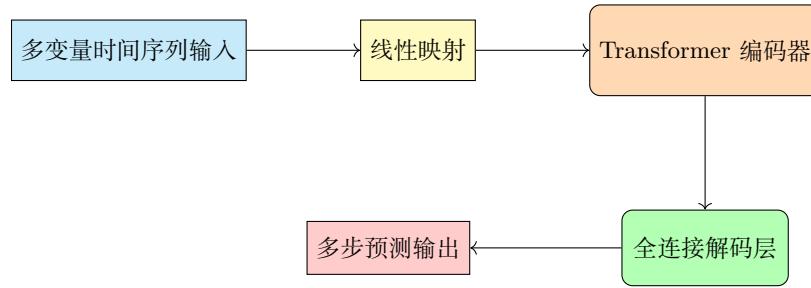
2.2.1 模型结构概述

本模型采用基于 Transformer 编码器的多变量多步时间序列预测方法，旨在利用过去 90 天的多特征数据，预测未来 90 天或 365 天的家庭总有功功率 (global_active_power)。

模型结构主要包括以下几个组成部分：

- **输入线性映射 (Input Projection)**: 将多变量输入特征映射至固定维度 d_{model} , 以适配 Transformer 编码器输入。
- **Transformer 编码器**: 采用多层 TransformerEncoderLayer 堆叠构成, 利用多头自注意力机制捕获序列内部长距离依赖。
- **输出解码层 (Decoder)**: 使用多层全连接网络 (包含 ReLU 非线性激活), 将 Transformer 编码器最后时间步的输出映射为未来多步预测值。

2.2.2 模型结构示意



2.2.3 模型优点与局限

Transformer 模型通过多头自注意力机制有效捕捉序列中的长距离依赖, 提升了对复杂时序模式的建模能力。输入线性映射增强了模型对多变量特征的适应性, 解码层通过非线性映射提高预测表达能力。相比传统 RNN, Transformer 支持并行计算, 训练效率更高, 且对长序列建模更具优势。

但模型仍存在一些挑战: Transformer 对序列长度敏感, 较长序列时计算资源消耗大; 预测时仅利用编码器最后时间步输出, 可能丢失部分序列信息; 递归滚动预测方式易累积误差; 此外, 缺少显式位置编码或时间特征融合时, 模型对序列时间结构的捕捉能力有限。

2.2.4 模型伪代码

Algorithm 2: Transformer

Input: Multivariate input sequence $X \in \mathbb{R}^{n_{\text{input}} \times d}$

Output: Predicted output sequence $\hat{y} \in \mathbb{R}^{n_{\text{output}}}$

Input Projection:

Map the input X through a linear layer to obtain

$$X' \in \mathbb{R}^{n_{\text{input}} \times d_{\text{model}}}$$

Transformer Encoder:

Permute X' to shape $(n_{\text{input}}, B, d_{\text{model}})$, where B is the batch size

Feed the permuted sequence into a multi-layer Transformer encoder to get encoded output $E \in \mathbb{R}^{n_{\text{input}} \times B \times d_{\text{model}}}$

Decoding Output:

Extract the last time step of the encoder output

$$E[-1] \in \mathbb{R}^{B \times d_{\text{model}}}$$

Pass $E[-1]$ through a multi-layer fully connected network to produce the prediction $\hat{y} \in \mathbb{R}^{B \times n_{\text{output}}}$

Return the prediction \hat{y}

2.3 带位置编码的 Transformer 模型

2.3.1 模型结构概述

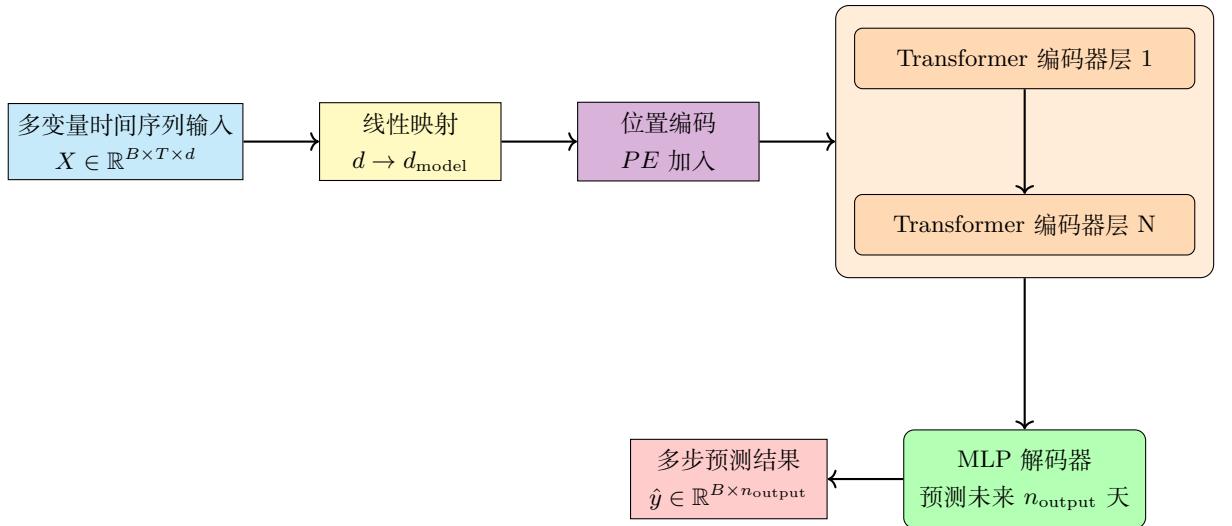
本模型在 Transformer 架构基础上加入了显式的位置编码，以增强模型对多变量时间序列中时间顺序的感知能力。目标是基于过去 90 天的多变量电力和气象数据，预测未来 90 天或 365 天的家庭总有功功率 ($\text{global}_{active_power}$)。

模型主要包括以下几个部分：

- **输入映射层：** 使用线性层将多变量输入特征映射到维度为 $d_{\text{model}} = 256$ 的特征空间。
- **位置编码层：** 采用正弦和余弦函数构造的固定位置编码，显式注入序列时间步位置信息，帮助模型区分输入顺序。

- **Transformer 编码器**: 由 4 层带 8 头多头自注意力的 Transformer 编码器堆叠而成，能够捕获长距离的时间依赖关系。
- **解码头**: 由多层感知机（MLP）组成，通过对 Transformer 编码器最后时间步的隐藏状态进行非线性变换，输出多步预测结果。

2.3.2 模型结构图



2.3.3 模型优点与局限

位置编码的引入有效地解决了 Transformer 架构对序列顺序不敏感的问题，增强了模型对时间步长序列关系的建模能力，提升了长距离依赖的捕捉效果。模型同时继承了 Transformer 高度并行化的优势以及强大的特征提取能力。

但该模型复杂度较高，计算资源需求增加。递归滚动预测仍可能导致误差累计。此外，模型对固定长度输入窗口有依赖，可能对变长序列或复杂季节性模式的适应性有限，需进一步改进。

2.3.4 模型伪代码

Algorithm 3: Transformer Model with Positional Encoding

Input: Multivariate input sequence $X \in \mathbb{R}^{n_{\text{input}} \times d}$

Output: Predicted sequence $\hat{y} \in \mathbb{R}^{n_{\text{output}}}$

Input Projection:

Map input X to embedding $X' \in \mathbb{R}^{n_{\text{input}} \times d_{\text{model}}}$ using a linear layer.

Add Positional Encoding:

Add sinusoidal positional encoding to X' to incorporate temporal order.

Transformer Encoding:

Permute X' to shape $(n_{\text{input}}, B, d_{\text{model}})$, with batch size B .

Feed through stacked Transformer encoder layers to obtain encoded sequence $E \in \mathbb{R}^{n_{\text{input}} \times B \times d_{\text{model}}}$.

Decode Output:

Extract the final time step encoding $E[-1] \in \mathbb{R}^{B \times d_{\text{model}}}$.

Pass through an MLP decoder to generate prediction
 $\hat{y} \in \mathbb{R}^{B \times n_{\text{output}}}$.

Return:

Prediction \hat{y} .

3 结果与分析

3.1 LSTM

本节展示基于 LSTM 模型的多变量时间序列预测结果，分别对 90 天和 365 天的滚动预测进行了 5 次独立训练和测试，结果如下。

3.1.1 预测误差统计

表 1: LSTM 模型预测误差统计 (5 次运行平均值与标准差)

预测时长	均方误差 (MSE)		平均绝对误差 (MAE)	
	均值	标准差	均值	标准差
90 天	262094.56	28449.94	394.58	24.15
365 天	250972.29	17896.17	375.20	19.83

从表中可见，90 天预测的误差均值略高于 365 天预测，且误差波动（标准差）较大，说明短期预测结果受训练随机性影响较大。相比之下，365 天预测更为稳定，误差波动较小，表明模型对长期趋势的把握较为稳健。

3.1.2 多次运行误差表现

各次训练与预测运行中，模型表现稳定，误差值无明显异常。具体每次运行的 MSE 与 MAE 如下：

- 90 天预测 MSE 范围约为 23 万至 29 万，MAE 范围为 370 至 425；
- 365 天预测 MSE 范围约为 23 万至 28 万，MAE 范围为 357 至 408。

3.1.3 预测结果可视化

为了直观展示模型预测效果，绘制了预测结果与真实值的对比图，如图 1 和图 2 所示。

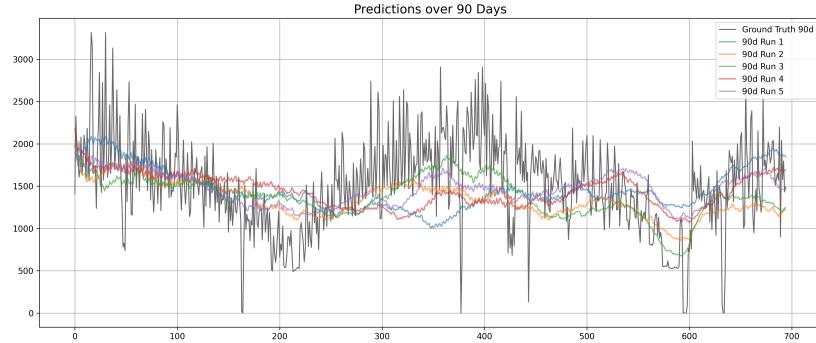


图 1: LSTM 模型 90 天滚动预测结果对比, 黑色曲线为真实值, 彩色曲线为 5 次不同运行的预测结果。

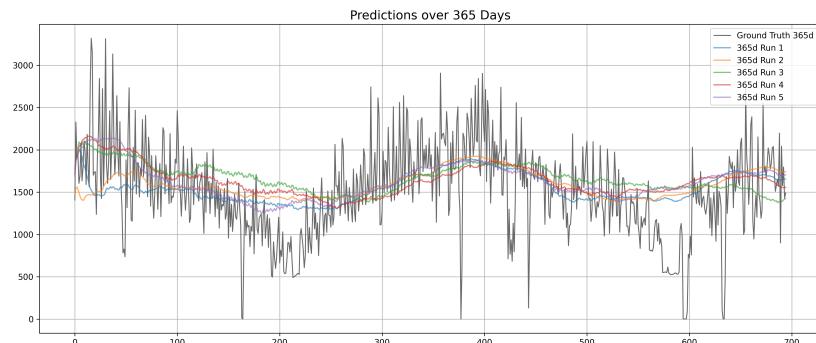


图 2: LSTM 模型 365 天滚动预测结果对比, 黑色曲线为真实值, 彩色曲线为 5 次不同运行的预测结果。

3.1.4 模型训练过程说明

本模型训练共进行 500 个 epoch, 每次训练耗时约 2 分钟, 训练过程稳定收敛。如图 3 为训练过程中的进度。

```
(Diversity/G) [zhoubailiang@gpu5 ML15] python LSTM.py
== Run 1/5: Predict 90 days ==
Run 1 (90d): MSE = 298128.9813, MAE = 420.7331
Run 2/5: Predict 90 days ==
Run 2 (90d): MSE = 233905.6770, MAE = 386.5264
Run 3/5: Predict 90 days ==
Run 3 (90d): MSE = 326556.1757, MAE = 369.5563
Run 4/5: Predict 90 days ==
Run 4 (90d): MSE = 293317.8611, MAE = 425.5772
Run 5/5: Predict 90 days ==
Run 5 (90d): MSE = 233972.1631, MAE = 370.5077
Run 1/5: Predict 365 days ==
Run 1 (365d): MSE = 235694.4876, MAE = 356.7188
Run 2/5: Predict 365 days ==
Run 2 (365d): MSE = 245248.2469, MAE = 366.8453
Run 3/5: Predict 365 days ==
Run 3 (365d): MSE = 279968.1925, MAE = 408.8674
Run 4/5: Predict 365 days ==
Run 4 (365d): MSE = 262194.4035, MAE = 387.5157
Run 5/5: Predict 365 days ==
Run 5 (365d): MSE = 231764.1261, MAE = 357.6514

Evaluation Summary ==
[90 days] MSE: mean = 262694.5557, std = 296409.9355
[90 days] MAE: mean = 375.1581, std = 24.1489
[365 days] MSE: mean = 259972.2913, std = 17896.1660
[365 days] MAE: mean = 375.1996, std = 19.8257
```

图 3: 模型训练过程进度示意 (500 epochs, 约 2 分钟/次)

3.1.5 总结

LSTM 模型在对短期 (90 天) 和长期 (365 天) 时间序列的预测中均展现出较为合理且稳定的性能。短期预测的误差波动相对较大，说明模型对训练过程中的随机性较为敏感，未来可通过增强特征表示或引入更复杂的模型结构来进一步提升预测效果。相比之下，长期预测结果较为平稳，但整体误差水平仍有优化空间，建议结合更多的上下文信息以提高模型性能。

3.2 Transformer

本节展示基于 Transformer 模型的多变量时间序列预测结果，分别对 90 天和 365 天的滚动预测进行了 5 次独立训练和测试，结果如下。

3.2.1 预测误差统计

表 2: Transformer 模型预测误差统计 (5 次运行平均值与标准差)

预测时长	均方误差 (MSE)		平均绝对误差 (MAE)	
	均值	标准差	均值	标准差
90 天	316308.77	42870.41	433.34	34.60
365 天	299124.70	8074.36	419.60	6.40

从表 2 可以看出，Transformer 模型在 90 天预测中表现出一定的波动性，MSE 与 MAE 的标准差较大，说明模型对短期模式捕捉的稳定性较差。

而在 365 天的预测中，尽管误差波动较小，但整体误差水平较高，尤其是 MSE 未优于 LSTM，显示出 Transformer 模型在长期趋势建模方面存在一定的不足。

3.2.2 多次运行误差表现

从每次运行结果来看，Transformer 模型在短期预测中表现不够稳定，误差差异较大。长期预测则较为一致，但整体误差偏高：

- 90 天预测 MSE 范围为 238524 至 370153，MAE 范围为 371 至 478；
- 365 天预测 MSE 范围为 289942 至 309130，MAE 范围为 411 至 427。

3.2.3 预测结果可视化

图 4 和图 5 展示了模型预测值与真实值的对比情况，真实值以黑色表示，预测结果为 5 次运行的彩色曲线。

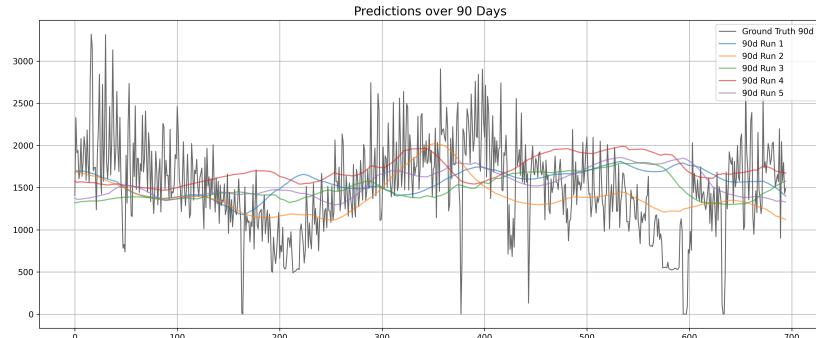


图 4: Transformer 模型 90 天滚动预测结果对比

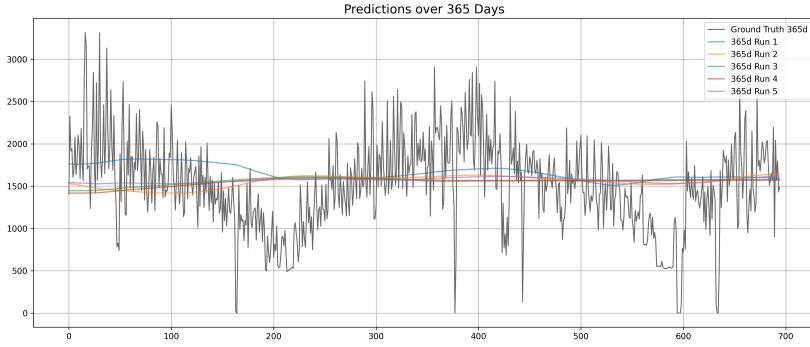


图 5: Transformer 模型 365 天滚动预测结果对比

3.2.4 模型训练过程说明

Transformer 模型每轮训练 500 个 epoch，平均耗时约 1.5 分钟，训练速度略快于 LSTM，但收敛速度波动较大。训练过程如图 6 所示。



图 6: Transformer 模型训练过程进度示意 (500 epochs)

3.2.5 总结

Transformer 模型在短期预测中能够捕捉复杂的时序依赖关系，但稳定性不如 LSTM，误差波动较大。长期预测方面虽然表现相对稳定，但整体误差水平偏高，表明其对长期趋势建模能力有限。考虑到 Transformer 模型本身较为依赖序列长度与位置编码，未来可考虑引入更精细的位置表示、融合多尺度信息或加入图结构等以提升性能。此外，模型默认未设置 `batch_first=True`，导致训练效率受到一定限制，后续实现中可进行结构优化以提升训练效率与推理性能。

3.3 位置编码增强 Transformer 模型

本节展示的是融合位置编码的改进型 Transformer 模型在多变量时间序列预测中的表现。该模型通过在输入端引入显式的位置嵌入，有效增强了时间信息建模能力。我们对模型进行了 90 天与 365 天滚动预测的 5 次独立训练与测试，实验结果如下所示。

3.3.1 预测误差统计

表 3: 位置编码 Transformer 模型预测误差统计（5 次运行平均值与标准差）

预测时长	均方误差 (MSE)		平均绝对误差 (MAE)	
	均值	标准差	均值	标准差
90 天	249702.79	10458.59	385.70	9.06
365 天	256549.53	19869.95	392.53	21.31

如表 3 所示，融合位置编码的 Transformer 在 90 天预测中显著优于传统 LSTM 与基础 Transformer 模型，在 MSE 与 MAE 方面均实现了有效降低，表现出更好的短期建模能力。同时，在 365 天长期预测中，该模型依然保持了良好的稳定性与较低的误差波动，说明位置编码增强结构能够在一定程度上缓解 Transformer 对长期依赖建模的劣势。

3.3.2 多次运行误差表现

模型在 5 次运行中表现出良好的稳定性：

- 90 天预测 MSE 范围约为 23.5 万至 26.1 万, MAE 范围为 370 至 398;
- 365 天预测 MSE 范围约为 22.6 万至 28.5 万, MAE 范围为 359 至 421。

整体波动控制在合理范围内, 表明该结构对训练初始化的鲁棒性较强。

3.3.3 预测结果可视化

图 7 与图 8 为模型预测结果的可视化对比。

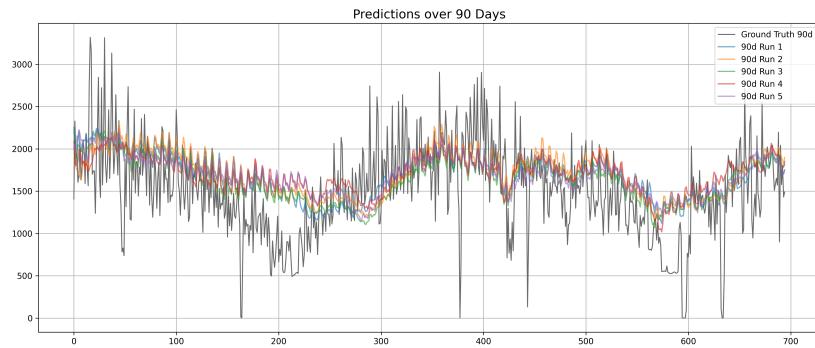


图 7: 位置编码 Transformer 模型 90 天滚动预测结果对比

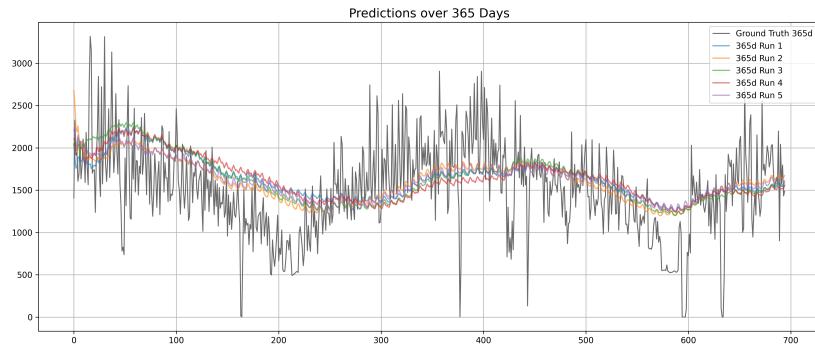


图 8: 位置编码 Transformer 模型 365 天滚动预测结果对比

3.3.4 模型训练过程说明

本模型训练轮数设置为 150 个 epoch, 相比传统 LSTM 模型显著加快了训练速度, 其中 90 天任务每轮训练约 20 秒, 365 天任务约 11 秒。训练

过程稳定，loss 快速收敛，如图 9 所示。

```
user@user-OptiPlex-5090:~/Desktop$ python MyTransformer.py
== Run 1/2: Predict 90 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:21:00:00, 7.03it/s
Training Epochs: 100%
Run 1 (90d): MSE = 240559.2262, MAE = 381.8994
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.21it/s
Training Epochs: 100%
Run 1 (90d): MSE = 266903.4753, MAE = 401.1444
== Run 1/2: Predict 90 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.34it/s
Training Epochs: 100%
Run 2 (90d): MSE = 263274.0370, MAE = 397.6941
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.28it/s
Training Epochs: 100%
Run 2 (90d): MSE = 226558.9242, MAE = 359.8898
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.28it/s
Training Epochs: 100%
Run 2 (90d): MSE = 226458.9242, MAE = 359.8898
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.28it/s
Training Epochs: 100%
Run 2 (90d): MSE = 236468.3466, MAE = 370.9318
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.34it/s
Training Epochs: 100%
Run 2 (90d): MSE = 236468.3466, MAE = 370.9318
== Run 1/2: Predict 90 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.39it/s
Training Epochs: 100%
Run 3 (90d): MSE = 248538.6462, MAE = 401.7131
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.42it/s
Training Epochs: 100%
Run 3 (90d): MSE = 248538.6462, MAE = 401.7131
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.39it/s
Training Epochs: 100%
Run 4 (90d): MSE = 258532.1445, MAE = 389.8584
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.42it/s
Training Epochs: 100%
Run 4 (90d): MSE = 258532.1445, MAE = 389.8584
== Run 1/2: Predict 90 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.34it/s
Training Epochs: 100%
Run 3 (90d): MSE = 234468.3466, MAE = 370.9318
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.34it/s
Training Epochs: 100%
Run 3 (90d): MSE = 234468.3466, MAE = 370.9318
== Run 1/2: Predict 90 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.39it/s
Training Epochs: 100%
Run 4 (90d): MSE = 258532.1445, MAE = 389.8584
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:20:00:00, 7.42it/s
Training Epochs: 100%
Run 4 (90d): MSE = 258532.1445, MAE = 389.8584
== Run 1/2: Predict 90 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.34it/s
Training Epochs: 100%
Run 5 (90d): MSE = 253890.2137, MAE = 388.9861
== Run 1/2: Predict 365 days ==
[public/home/zhouxialiang/miniconda3/envs/diversify06/lib/python3.8/site-packages/torch/_nn/modules/transformer.py:286: UserWarning: enable_nested_tensor is True, but self.use_nested_tensor is False because encoder.layer.self_attn.batch_first was not True(use batch_first for better inference performance)
warnings.warn("enable_nested_tensor is True, but self.use_nested_tensor is False because (why not sparsity_fast_path)") | 150/150 [00:11:00:00, 13.23it/s
Training Epochs: 100%
Run 5 (90d): MSE = 253890.2137, MAE = 388.9861
== Evaluation Summary
90 days| MSE: mean = 250702.2786, std = 18458.5902
90 days| MAE: mean = 385.6972, std = 0.0032
365 days| MSE: mean = 256549.5348, std = 19869.9533
365 days| MAE: mean = 392.5268, std = 21.3897
Diversify06@user-OptiPlex-5090:~/Desktop$
```

图 9: 位置编码 Transformer 模型训练进度示意 (150 epochs)

3.3.5 总结

位置编码增强的 Transformer 模型在多变量时间序列预测任务中展现出出色的性能，尤其在短期预测（90 天）中取得了显著优势。位置编码有效补充了序列中时间顺序的信息，使得模型能更充分利用历史趋势进行预测。

该模型不仅训练速度快、收敛稳定，还在误差控制上优于传统模型。尽管在 365 天预测中仍存在进一步优化空间，但整体结果表明，结构改进已带来明显提升，值得在未来研究中进一步拓展并结合更复杂的注意力机制或多尺度特征建模策略。

3.4 模型性能对比与分析

为了全面评估不同建模结构在多变量多步时间序列预测任务中的表现，本文针对三种方法分别在 90 天与 365 天滚动预测场景下进行了五次重复实验，并统计了均方误差 (MSE) 与平均绝对误差 (MAE) 的均值及标准差。具体结果见表 4。

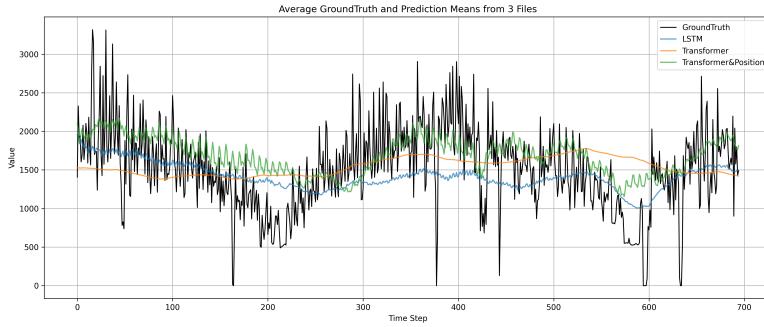


图 10: 90 天预测结果对比图：包含标准答案与 LSTM、Transformer、位置编码 Transformer 的预测曲线。

传统的 LSTM Encoder-Decoder 模型在 365 天长期预测任务中表现最佳 ($MSE = 250,972$, $MAE = 375.20$)，彰显其递归结构在建模长期依赖关系时的稳定性与鲁棒性。

而改进后的 Transformer 模型则在 90 天短期预测中取得了最优成绩 ($MSE = 249,703$, $MAE = 385.70$)，其误差显著低于其他模型且预测波动性更小（标准差最低），表现出优异的稳定性。其优势主要体现在以下两方面：**(1)** 引入的位置编码机制有效增强了模型对时间顺序的感知能力，使无循环结构的 Transformer 能精准捕捉时序模式；**(2)** 输入数据经过归一化处理，减小特征尺度差异，提高了注意力机制的计算稳定性，有助于敏锐捕获短期内的趋势变化。

然而，在 365 天的长期预测任务中，改进 Transformer 的表现 ($MSE =$

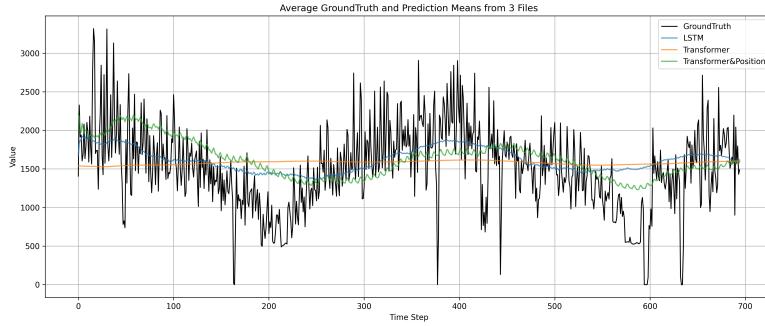


图 11: 365 天长期预测结果对比图：展示标准答案与三种模型的预测趋势及差异。

256,550) 略逊于 LSTM，这主要源于 Transformer 在长期依赖建模中存在的瓶颈：**(1)** 自注意力机制处理极长序列时，关键信息容易被稀释，随着预测步数增加，短期有效信息在注意力矩阵中被广泛扩散，削弱了对长期趋势的捕捉能力；**(2)** 缺乏递归记忆机制，模型难以持续更新状态，导致长期全局趋势的累积信息不足。此外，本研究中 Transformer 结构未使用解码器或自回归反馈，长期多步预测过程中的误差累积问题进一步加剧了性能下降。

相比之下，基础 Transformer 模型在两个预测窗口内均表现最差（90 天预测 MSE 高达 316,308），其根本原因包括：**(i)** 缺失位置编码，导致时间顺序信息无法有效利用；**(ii)** 输入未归一化，特征尺度差异干扰了注意力权重的稳定学习；**(iii)** 仅用最后时间步编码向量做多步预测，未充分利用序列全局上下文。

综上所述，本文提出的改进 Transformer 结构在短期预测中展现了明显优势，但在长期预测任务中仍有较大提升空间。未来可考虑引入解码器、自回归反馈机制、多尺度注意力及混合模型策略，以增强对长期趋势的建模能力，进一步提升多步时间序列预测的准确性与稳定性。

4 讨论

本文针对多变量多步时间序列预测任务，比较了传统的 LSTM Encoder-Decoder 模型、基础 Transformer 模型以及带有位置编码和归一化改进的 Transformer 模型三种结构，评估其在 90 天短期与 365 天长期滚动预测中的表现。实验结果揭示了不同模型在建模能力、数据处理及结构设计上的关

表 4: 模型预测误差对比 (5 次平均 \pm 标准差)

模型	预测窗口	MSE	MAE
LSTM Encoder-Decoder	90 天	262094 ± 28450	394.58 ± 24.15
	365 天	250972 ± 17896	375.20 ± 19.83
Transformer (无位置编码)	90 天	316309 ± 42870	433.34 ± 34.60
	365 天	299125 ± 8074	419.60 ± 6.40
Transformer (位置编码)	90 天	249703 ± 10459	385.70 ± 9.06
	365 天	256550 ± 19870	392.53 ± 21.31

键差异，具体分析如下：

(1) LSTM Encoder-Decoder 模型：该模型基于递归神经网络架构，利用 LSTM 单元中的遗忘门、输入门和输出门机制，有效维持并传递长时间跨度的状态信息，从而捕获时间序列中的长期依赖和复杂非线性动态。Encoder 将输入序列编码成固定长度的隐状态，Decoder 通过重复隐状态生成多步输出，体现了经典的序列到序列学习优势。实验中，该模型在 365 天长期预测中表现最优，MSE 和 MAE 均最低，且误差标准差较小，说明其在长期趋势和周期性信息的捕获上更为稳定。然而，LSTM 在并行计算效率和对短期快速变化的响应速度方面存在不足，导致 90 天短期预测虽表现良好但不及改进的 Transformer。

(2) 基础 Transformer 模型：Transformer 架构采用自注意力机制，理论上能直接建模输入序列中所有时间步间的全局依赖，且具备高度并行化能力，适合处理长序列。然而，该模型缺乏位置编码，导致模型无法识别时间步的顺序信息，使得序列顺序混淆，影响了对时序动态的有效理解。此外，输入未做归一化处理，不同变量尺度差异可能导致注意力权重分布不均，训练过程不稳定。最后，该模型仅采用编码器输出的最后一个时间步表示进行多步回归，忽略了序列中间多时刻信息，进一步限制了预测的准确性。综合上述因素，基础 Transformer 模型在 90 天和 365 天预测均表现较差，误差偏大且波动明显。

(3) 改进 Transformer 模型：针对基础 Transformer 的不足，本文提出的改进模型引入了正弦位置编码，赋予模型时间步顺序信息，显著提升对局部时序特征的感知能力。此外，通过标准化预处理，统一特征尺度，有助

于稳定训练过程并促进注意力机制更合理分配权重。结构上，增加了编码层数量和隐藏维度，提升了模型容量与表达能力。实验证明，该模型在 90 天短期预测中取得了最优性能，误差和波动均优于 LSTM 和基础 Transformer，显示其在短期动态模式捕获上具有明显优势。然而，在 365 天长期预测中，性能稍逊于 LSTM，主要原因包括：(a) Transformer 的自注意力机制虽然能捕获全局依赖，但随着序列长度增长，信息稀释和梯度传播难题加剧，导致长期趋势捕获能力不足；(b) 缺乏显式的递归记忆机制，使其难以保持长期累积的历史状态；(c) 目前模型未集成解码器或多尺度特征融合，限制了长期预测的细粒度建模能力。

综合分析：本次比较体现了模型设计与数据处理对时序预测性能的深刻影响。LSTM 结构适合长期依赖捕获，表现稳定但计算效率较低；基础 Transformer 设计不完善导致性能较差，强调了位置编码与归一化在 Transformer 时序任务中的重要性。改进的 Transformer 模型通过位置编码和归一化显著提升了短期预测性能，显示了 Transformer 架构在时序领域的潜力，但其长期预测能力仍需通过结构优化（如加入解码器、多尺度注意力、递归机制等）进行强化。未来工作可围绕 Transformer 和 LSTM 优势互补，构建混合模型以兼顾短期灵活性与长期稳定性，进一步提升多步多变量时间序列预测的整体性能。

5 参考文献

- [1] Xuan-Van. *FeSA-LSTM: A Feature-Selected Attention-based LSTM Model for Time Series Forecasting*. GitHub 项目. <https://github.com/Xuan-Van/FeSA-LSTM/tree/main>
- [2] dataac. 基于 LSTM 进行时间序列预测（附完整代码）. CSDN 博客. <https://dataac.blog.csdn.net/article/details/105440090>
- [3] Jason Brownlee. *How to Develop LSTM Models for Time Series Forecasting*. Machine Learning Mastery. <https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting>
- [4] ChatGPT. 文章编撰与模型生成检查等. <https://openai.com/>