SCLformer 一种新的长期时间序列预测模型

2023 年 5 月 24 日

[摘要]：时序预测是一种现实生活中常见的场景，其表现为给定一段时间段数据预测未来时间段的情况。通常，时序预测涉及较短的数据序列，而对于长期序列，现有的模型往往无法取得理想的预测效果。长期时间序列问题对模型本身提出的较高要求，应该能捕获长时间长序列中的依赖关系。近年来 Transformer 在时序预测领域逐渐崭露头角，尽管 Transformer 本身无法直接适应长期时间序列问题，但通过改进模型，例如对编码器、解码器和自注意力机制进行改进，可以使其更好地适应这类问题。本研究通过对现有改进模型 Informer 进一步改进，将 Transformer 模型的编码器和解码器进行一个上采样与下采样，使得输入进每一层的数据量可以大幅度降低，从而提高运算速度，同时在位置编码操作中，我们采用了一种自适应的编码，在四个大型数据集上的实验表明，本模型的效果可以优于 Informer 模型，为长期时间序列问题提供了一种新的模型方案。

[关键词]： Transformer 长期时间序列

[**ABSTRACT**]: Time series prediction is a common scenario in real life, which involves predicting the future based on a given time period data. Typically, time series prediction involves short data sequences, while for long-term sequences, existing models often fail to achieve ideal prediction results. Long-term time series issues impose higher demands on the model itself, requiring it to capture dependency relationships over a long time period. In recent years, the Transformer has gradually emerged in the field of time series prediction. Although the Transformer itself cannot directly adapt to long-term time series problems, by improving the model, such as improving the encoder, decoder, and self-attention mechanism, it can better adapt to this type of problem. In this study, we further improved the existing improved model Informer, by upsampling and downsampling the encoder and decoder of the Transformer model, so that the amount of data input into each layer can be greatly reduced, thereby improving the computing speed. At the same time, we used an adaptive encoding method in the position encoding operation. Experiments on four large datasets showed that the performance of the proposed model can be superior to that of the Informer model, providing a new model solution for long-term time series problems.

[**Key words**]: Transformer, Long-term time series

目录

1. 介绍 **1**
2. 相关工作 **2**
2.1 传统模型 2
2.2 Transform 模型 2
2.3 Informer 模型 3
3. 初步研究 **5**
3.1 问题定义 5
3.2 Transformer 的架构 6
4. 方法论 **6**
4.1 位置编码器 6
4.2 自注意力机制 7
4.3 编码器和解码器 9
5. 实验 **10**
5.1 数据集 10
5.2 实验细节 11
5.2.1 实验设置 11
5.2.2 基准测试 12
5.3 实验结果 12
5.4 超参测试 12
5.4.1 Encoder 层数 12
5.4.2 GRU 的隐藏层层数 15
5.4.3 seq len 长度 16
5.4.4 label len 的长度 16
5.4.5 多头注意力机制头数 17
5.4.6 采样因子数 18
5.4.7 模型宽度 19

6. 结论 **22**
参考文献 **23**

1. 介绍

时间序列预测问题是一种基于历史数据来预测未来变化的问题，它在许多领域都有着广泛的应用，例如股票的未来走向的预测，风力的未来预测等都是时序预测问题中常见的情形。随着深度学习领域的发展，基于 RNN 和 CNN 模型等技术，已经取得了比传统统计方法更好的效果[1]。

在上述场景中，我们可以利用大量的历史数据来进一步预测未来数据，使得预测更加的精确，这种预测需求被称为长时间序列预测问题 (Long sequence time-series forecasting, LSTF)。长时间预测和短时间预测是时序预测中两个不同的研究方向，其侧重点各有不同。长时间序列预测问题指的是输入数据的时间跨度很大，例如几个月或者几年的数据，如图一所示。其相较于短期数据表现为预测跨度较大，相较于短期数据常见的 48 或更少的时间节点，长期序列则会拥有几百甚至更多的预测需求。长时间序列预测问题的难点在于，长时间序列中可能包含了多种复杂的模式和趋势，例如周期性、季节性、趋势性等，这些模式和趋势可能会随着时间的变化而发生变化，或者相互影响。因此，长时间序列预测问题需要能够捕捉到这些模式和趋势的变化，并且能够有效地从长时间序列中提取出有用的特征。

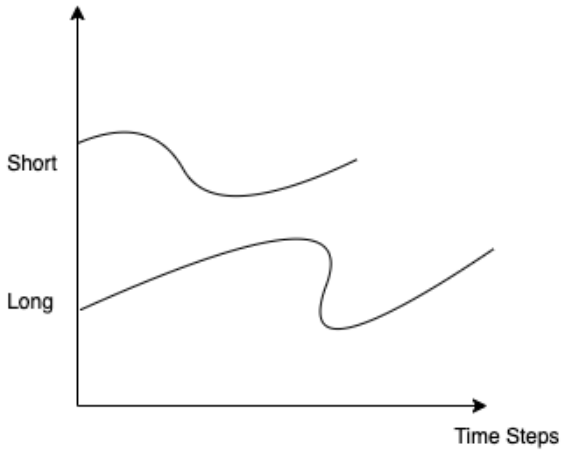


图 1 序列预测长度

时序问题主要是根据历史的统计数据而预测未来变化趋势，由于其输入数据为一段时间的历史数据，也意味着对于时序预测问题，输入的时间序列越长，时间范精度越大，则预测效果会更好。传统的时序预测模型，例如自回归模型 (AR)、移动平均模型 (MA)、自回归移动平均模型 (ARMA) 等，都是基于线性假设的，它们往往不能很好地处理非线性和非平稳的长时间序列。另一类常用的时序预测模型是基于神经网络的模型，例如循环神经网络 (RNN)、长短期记忆网络 (LSTM)、门控循环单元 (GRU)[2]等，它们能够处理非线性和非平稳的数据，但是也存在一些问题。一方面，神经网络模型需要大量的数据和计算资源来训练，这对于长时间序列预测问题来说可能不太实际。另一方面，神经网络模型也容易遇到长期依赖问题，即难以捕捉到长时间序列中较远的历史信息对当前预测的影响。

近年来，Transformer 在长期依赖捕获领域展现出优于传统模型 RNN 的效果。自注意力机制可以将序列的传播路径最大长度降低到理论上最短的 $O(1)$，并且可以避开循环结构，这使得基于自注意力机制的 Transformer 在长期时间序列问题上有无限可以挖掘的潜力。

2. 相关工作

2.1 传统模型

RNN 是一种递归神经网络，于 1980 年代被提出；与传统神经网络不同，它能够处理带有时序特征的数据，并根据上一个时间步的输出作为当前时间步的输入。虽然 RNN 在时序预测中有很好的表现，但是它存在的短暂记忆问题（Vanishing gradients）限制了它应用的广泛性。

为了解决 RNN 的短暂记忆问题，1997 年 Hochreiter 和 Schmidhuber 提出了一种新型的神经网络：长短时记忆网络（LSTM）。LSTM 的灵感来源于计算机科学中的计数器和指针等记忆单元，引入了三个门来控制信息的流入和流出。通过这些门的组合，LSTM 能够在长序列中提取、利用并保存信息，具有了处理长时序数据的能力。LSTM 单元的隐藏状态 s_t 是由一系列门和激活函数操作组成的，它们的输入是上一次的隐藏状态 s_{t-1} 和当前的输入 x_t。遗忘门和输入门分别控制有多少信息应该被遗忘或添加到当前状态中。候选状态是用当前输入和上一次的输出计算出的。输出门和 LSTM 单元的最终输出分别用于确定哪些信息将被输出到下一个时间步。由于它能够解决 RNN 的短暂记忆问题，因此在时序预测中表现更出色。

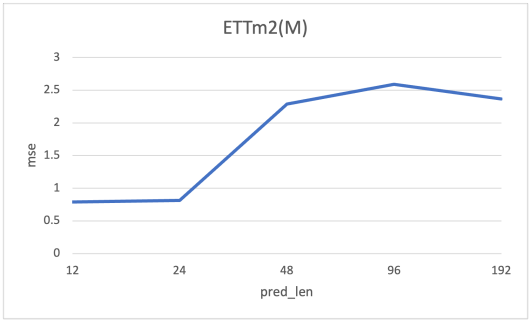
除此之外，ARMA 模型也是常用的传统时间序列预测模型，它是结合了自回归（AR）和移动平均（MA）两种模型的特点。ARMA 模型的优点是适用于线性时间序列的预测，并且较容易理解和解释。

传统模型虽然在时序预测上具有一定的优势，但是其在处理长序列上存在一定的不足，例如 RNN 模型在处理长序列时会出现梯度消失的问题，导致其无法捕捉到序列中的长期依赖关系；ARMA 模型在处理长序列时会出现参数过多的问题，导致其无法进行有效的训练。再如改进的 LSTM 模型，经过我们的实验，可以发现其在短序列上 (序列长度小于 48) 的效果较好，但是当进入长预测序列后效果下降较快。

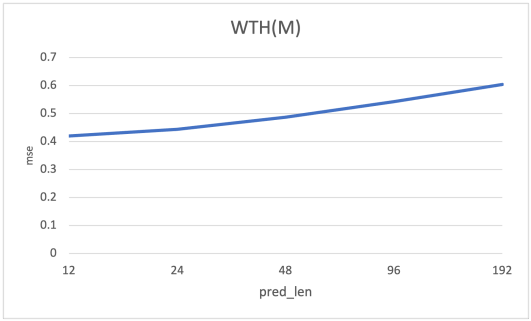
2.2 Transform 模型

Transformer 模型是谷歌于 2017 年提出在机器翻译领域使用的一种深度学习模型[3]，Transformer 主要是由编码器和解码器组成，其中编码器和解码器都采用了多头注意力机制，由多少个编码器就有多少个解码器，每一个编码器都相同，包含两个子层：自注意力层和前馈神经网络，解码器则多了一个 Masked 的多头注意力层。

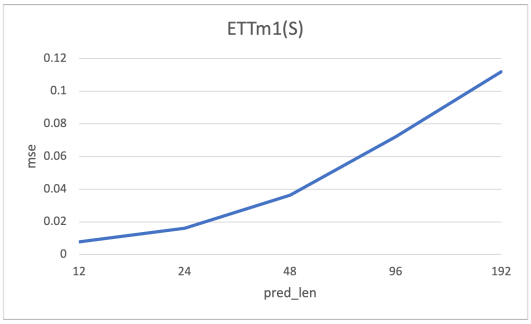
现有的 Transformer 的时间序列模型如图 3 所示。模型的输入是时间序列，首先经过一个编码层，将时间序列中的每个时间步的特征向量映射到高维向量空间，使得原始序列具有位置或日期的特定标记。这些向量然后会在 Transformer 编码器中被编码，通过注意力机制来捕获时间点之间的关系，生成一个新的向量序列，该序列再被



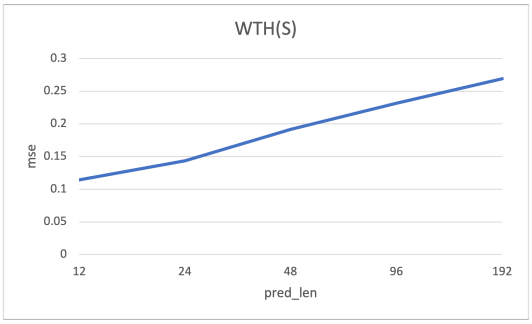
a) 多变量情况下 ETTm2 上的 LSTM 效果



b) 多变量情况下 WTH 上的 LSTM 效果



c) 单变量情况下 ETTm1 上的 LSTM 效果



d) 单变量情况下 WTH 上的 LSTM 效果

图 2 LSTM 在测试数据集上随序列长度的效果变化，MSE 越小代表效果越好

送到 Transformer 解码器进行解码，完成预测操作以得到最终的预测结果。

对于长期依赖问题对于 Transformer 则不会出现，Transformer 本身基于多头注意力 (Multi-head Attention) 机制使其可以同时兼顾每一个输入的时间片，对每一个时间片赋予不同的注意力值。由于这种原理，导致 Transformer 在处理时间序列上天然不会忽略之前的影响，这也就解决了 RNN 等传统模型遇到的长期依赖问题。尽管 Transformer 解决了传统模型中的长期依赖问题，但是其仍然在处理长序列问题上存在一定不足。

为了解决这个问题，一些研究者提出了一些改进的方法，例如使用相对位置编码[4]、分层注意力[5]、局部自注意力[6]等。这些方法都是在保留 Transformer 的优点的同时，尝试增强其对长序列的建模能力。其中，相对位置编码是在多头注意力机制中引入了位置信息，使得 Transformer 可以捕捉到序列中的顺序关系；分层注意力是将输入序列分为不同的层次，并在每个层次上应用不同的注意力函数，从而实现对不同粒度的信息的关注；局部自注意力是将输入序列划分为若干个局部区域，并在每个区域内进行自注意力计算，从而减少计算复杂度和内存消耗。

2.3 Informer 模型

Transformer 模型在处理长序列上存在三个主要问题[7]：

1. 自注意力机制的时间空间复杂度为 $O(L^2)$，相对较慢

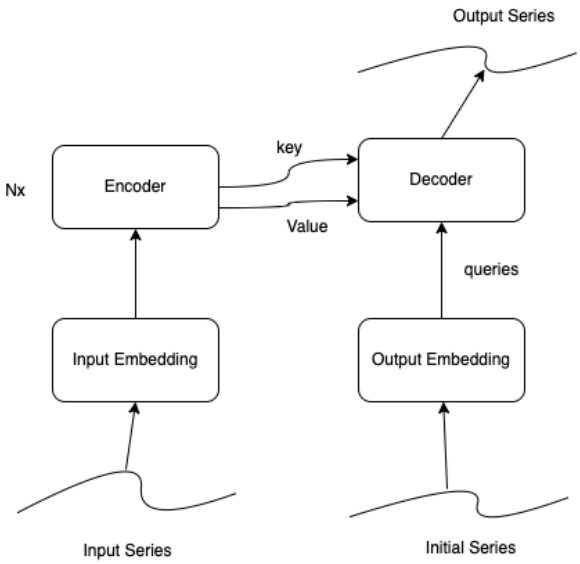


图 3 基于 Transformer 的预测模型的结构

2. Transformer 中编码器和解码器的参数量为 $O(NL^2)$，其中 N 为层数，L 为输入序列长度，在层数多或输入序列过长时会出现内存爆炸。

3. Transformer 的生成过程是一步一步生成出的，导致如果生成序列较长则会出现过慢的速度

Informer 的出现就是为了解决 Transformer 在应对过长序列的时候的问题。其主要提出了四个解决思路来解决上述的问题：

1. Informer 模型中使用了 ProbSparse Attention 机制来代替传统的 Self-Attention 机制进行编码。ProbSparse Attention 机制是一种基于稀疏注意力机制的注意力机制，它可以将时间空间复杂度降低为 $O(LlogL)$，从而使得模型在处理长序列时更加高效。

2. Informer 模型通过一种蒸馏机制，将编码器层数减半，从而使得模型的空间复杂度降低为 $O(\frac{N}{2}LlogL)$。这种蒸馏机制可以有效地减少模型的参数量，从而使得模型在处理长序列时更加高效。

3. Informer 模型采用了生成式解码器，使得解码器一次性可以生成长序列，从而显著提高了生成的序列长度。这种生成式解码器可以有效地解决 Transformer 在生成长序列时速度过慢的问题。

4. Informer 模型中还使用了相对位置编码，使得模型可以捕捉到序列中的顺序关系。相对位置编码是一种基于多头注意力机制的位置编码方式，它可以有效地提高模型在处理长序列时的性能。

Informer 的整体设计如图 4，编码器接收输入的长序列 X_en 然后使用 ProbSparse Attention 机制代替传统的 Self-Attention 机制进行编码，然后将编码后的结果送入解码器中，解码器使用生成式解码器进行解码，其中输入的 X_de 包括 token 编码和我们的目标预测元素 (填充为 0)，将其送入 ProbSparse Attention，与 Encoder 的输出合并后，最终得到预测结果。

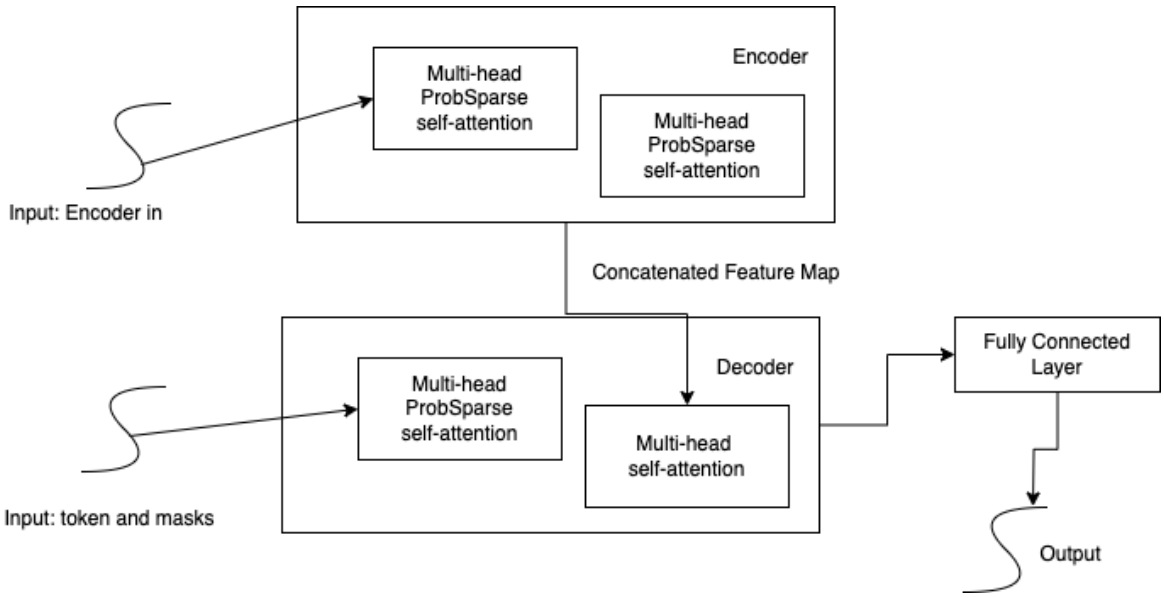


图 4 Informer 架构

通过 Informer 的设计，我们可以在长时间序列预测任务中实现更高的性能和更快的速度。Informer 模型通过改进 Transformer 模型的注意力机制、降低编码器层数和采用生成式解码器，有效地解决了长序列预测中的时间空间复杂度、参数量和生成速度等问题。这使得 Informer 模型在处理长时间序列预测问题时具有更强大的预测能力和更高的准确性。

3. 初步研究

3.1 问题定义

对于长期时间序列问题，本质上仍然是一种时间序列预测问题。我们在数学上则可以使用如下的语言来描述。

给定一个输入序列，代表历史上的时间数据 $X_h \in \mathbb{R}^{n*c}$，其中我们有 $X_h = \{x_1^h..., x_L^h | x_i^h \in \mathbb{R}^{n*c}\}$，其中 n 为输入序列长度，c 为通道数，其目标是需要预测输出序列 $X_f \in \mathbb{R}^{m*c}$，$X_f = \{x_1^f..., x_L^f | x_i^f \in \mathbb{R}^{m*c}\}$，其中 m 为预测长度，c 为通道数，即寻找 $X_h \in \mathbb{R}^{n*c}$ 与 $X_f \in \mathbb{R}^{m*c}$ 相关的映射关系：$F : X_h \in \mathbb{R}^{n*c} -> X_f \in \mathbb{R}^{m*c}$，其中 X_h 与 X_f 均为连续的实数序列。

3.2 Transformer 的架构

基于 Transformer 的设计，一般使用编码器-解码器架构。当原始序列 X_h 的每一个时间节点 x_t^h 经过编码器编码后得到隐状态 H，然后经过解码器输出得到结果 X_f。其中 H 为序列，即 $H = \{h_1^t ..., h_L^t\}$，对于编码器，其需要根据之前的隐状态得到下一个隐状态，即依靠 h_k^t 推测 $h_{(k+1)}^t$。从而根据 $h_{(k+1)}^t$ 得到 $x_{(k+1)}^f$。

而时间序列中，Transformer 的编码器和解码器并不知道时间的先后性，这就导致我们需要对时间序列进行一定的处理，即时间编码。时间编码的方法有很多，比如使用正弦函数和余弦函数，或者使用时间差等等。通过时间编码，我们可以将时间序列的先后性引入到 Transformer 中。即将原始的 $X_h \in \mathbb{R}^{n*c}$，通过 Embedding 操作得到 $X_{in} \in \mathbb{R}^{n*d}$。

综合上述我们可以将 Transformer 的基本思路写成如下的公式表达：

$X_{in} = InputEmbedding(X_h)$

由于 Encoder 和 Decoder 采用 Self-Attention 的方式，因此我们可以将其写成如下的公式表达：

$H = Encoder(X_{in})$

$H = softmax(\frac{QK^T}{\sqrt{d}})V,\ Q = K = X_{in},\ V = R_1 * X_{in}$ 其中 $R_1 \in \mathbb{R}^{n*n}$ 是经过 token 编码的数据信息。

$X_f = Decoder(H)$

$X_f = softmax(\frac{QK^T}{\sqrt{d}})V,\ K = H,\ V = R_2 * H$ 其中 $R_2 \in \mathbb{R}^{m*n}$ 是经过 output token 编码的数据信息, $Q \in \mathbb{R}^{m*d}$ 则是 Output Embedding 得到的去生成数据的查询矩阵。

本质上，Transformer 预测主要是包括两个阶段，即初期的编码，学习到序列的时间特征，然后再通过编解码器进行编解码，学习输入输出之间的映射关系，从而实现模型的完整训练过程。

4. 方法论

我们在设计的时候主要是考虑 Transformer 四个主要部分，即：位置编码器，自注意力机制，编码器，解码器。我们也将从这四个主要方面介绍我们的改进思路。

4.1 位置编码器

在我们的改进模型中，我们在编码器部分进行了更改，我们提出一种新的位置编码器，其设计如下：

我们将该编码器称之为时间自适应编码器，其主要思想是通过一个模型来学习时间序列的特征，从而达到更好的位置编码效果。在我们目前的方案中，我们使用的 GRU 来学习原始序列中的时间特征，同理也可以使用其他深度学习模型来自行学习编码过程中的信息，对于带有信息的 X 我们会将其进入自适应编码器来进行编码，目前我们采用的 Self-adaptive Embedding 是 GRU，通过 GRU 对每一个时间点的特征加以学习来更好的表述时间数据，从而解决之前单纯采用卷积或线性层进行编码的时候，对时间数据理解不深的问题。

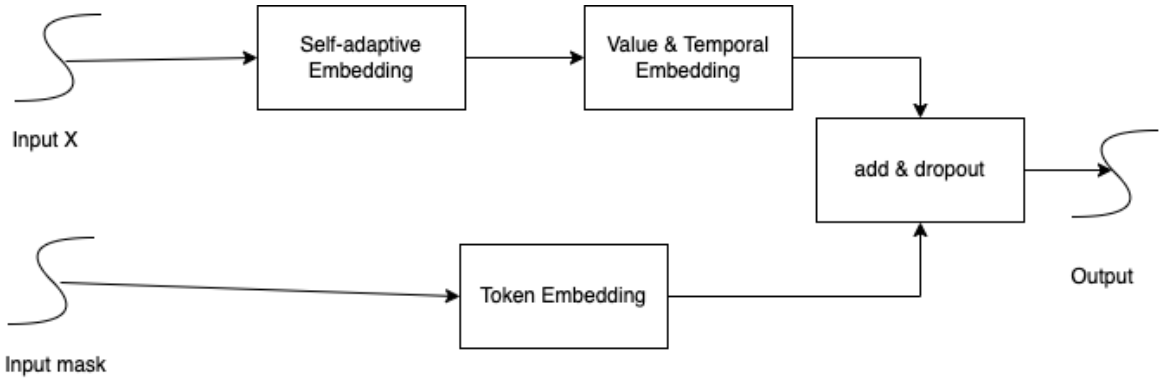


图 5 编码器设计

$$
\begin{aligned}
X_{in_{SelfEmbedding}} &= \text{Self-adaptive Embedding}(X_{in}) \\
X_{mask_{out}} &= \text{Token Embedding}(X_{mask}) \\
X_{in_{ValueEmbedding}} &= \text{Value Embedding}(X_{in_{SelfEmbedding}}) \\
X_{in_{TemporalEmbedding}} &= \text{Temporal Embedding}(X_{in_{SelfEmbedding}}) \\
X_{out} &= X_{mask_{out}} + X_{in_{ValueEmbedding}} + X_{in_{TemporalEmbedding}}
\end{aligned}
$$

其中$Self-adaptive\ Embedding = GRU$

时间自适应编码器采用了一个自适应的方式来对时间序列进行编码，不仅可以表达时间序列的时间信息，还能够学习时间序列的其他特征，如周期性、趋势等。通过采用这样的自适应编码器，相比于传统的编码方法，时间自适应编码器可以更好地表达时间序列中的时间信息和其他特征，并提高模型的表现能力。

4.2 自注意力机制

自注意力机制这里我们提出了一种线性连接层的注意力机制。传统的自注意力机制中，QKV 矩阵采用的是和原始的输入向量进行相乘，对于每一层的 Transformer 都进行这样的操作，从而达到学习时间序列的关系。在最新的研究中我们发现，对于时间序列问题，传统的线性模型也能具有一些比较好的效果[8]，除去 Transformer 本身的原始机制，在 Synthesizer[9]，MLP-mixer[10]和 MTS-Mixers[11]中也证明了这一点。时间序列本身具有低秩性，因此我们可以通过一些手段进行数据的特征提取，我们采用通过一个线性模型来进行该操作，并通过 dropout 和 normalization 来保证模型的稳定性，如图所示。对于中间可能由于线性模型的原因导致的梯度消失问题，则采用 ReLU 激活函数解决。其数学表达如下：

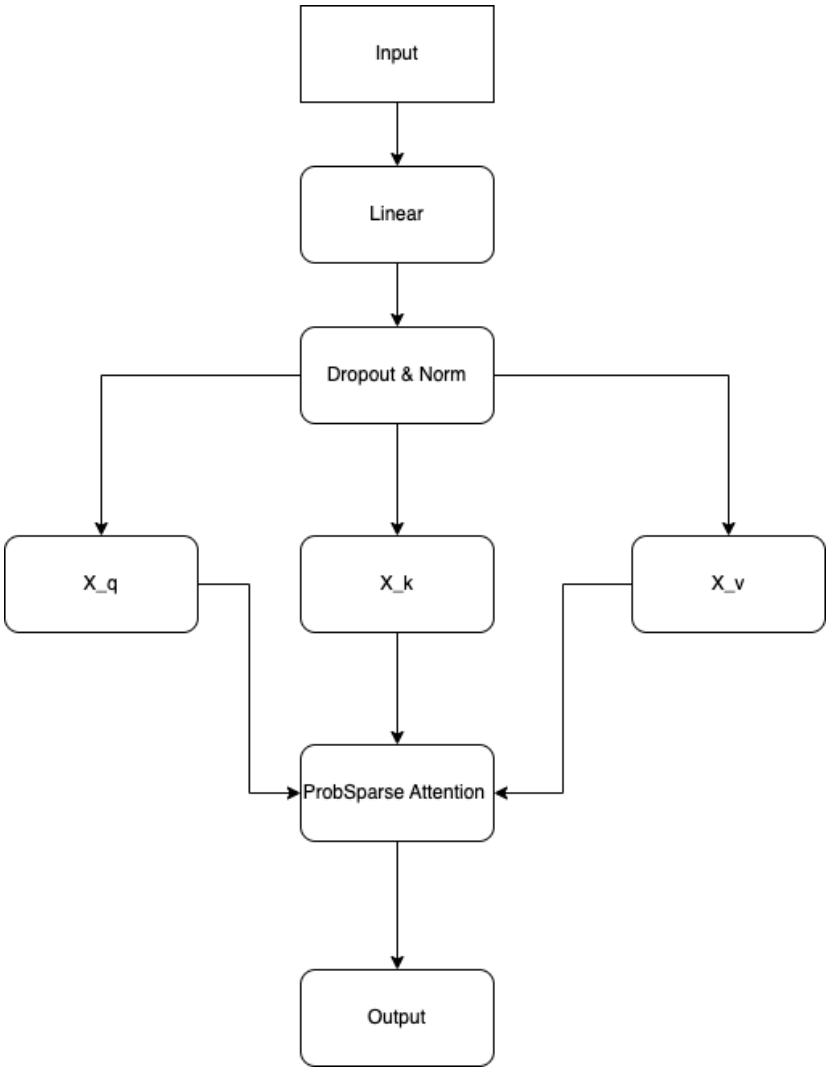


图 6 自注意力机制

$$
\begin{aligned}
X_{out} &= \text{Linear}(X_{in}) \\
X_{out} &= \text{Dropout}(\text{ReLU}(X_{out})) \\
X_{out} &= \text{Norm}(X_{out}) \\
X_{out} &= \text{Attention}(X_{out_{mid}}) \\
X_{out_{mid}} &= \text{Norm}(\text{Dropout}(\text{ReLU}(\text{Linear}(X_{in}))))
\end{aligned}
$$

首先对输入 X_{in} 进行线性变换，然后通过 ReLU 激活函数和 Dropout 进行非线性变换，再进行 Normalization 操作，得到 Attention 输出向量 X_{out}。

我们针对自注意力机制的改进主要有以下两个特点：

1. 我们在训练模型时使用了线性模型来进行特征提取，有效避免了传统的 QKV 矩阵计算带来的一些问题，提升了模型对序列信息的建模能力。

2. 我们还添加了 dropout 和 normalization 操作，以避免模型过拟合问题，并通过采用 ReLU 激活函数来解决梯度爆炸问题，从而提高了模型的稳定性和训练效果。

4.3 编码器和解码器

编码器和解码器是 Transformer 模型中的另外两个核心部分，能够直接影响模型的预测表现。U-net 模型中提出了一种将解码器输出卷积操作的思路[12][13]。我们的设计中则是进一步进行一种上采样与下采样综合的思路，其设计如下：

1. 我们每一个 Encoder 的输出都会经过一个卷积和池化的操作，将之前的序列输出卷积为一半，同时把输出的序列保留，以便后续使用。这样可以使得 Encoder 每一层的输入都为之前的一半，从而降低 Encoder 的运算量。

2. 我们在 Decoder 阶段对 Encoder 的输出进行反卷积操作，从而恢复之前被压缩的序列长度，并将 Encoder 的输出序列拼接后作为 Decoder 的输入。这样做可以确保 Decoder 的输入序列长度与原始序列的长度相同，同时也包含了 Encoder 的所有信息。

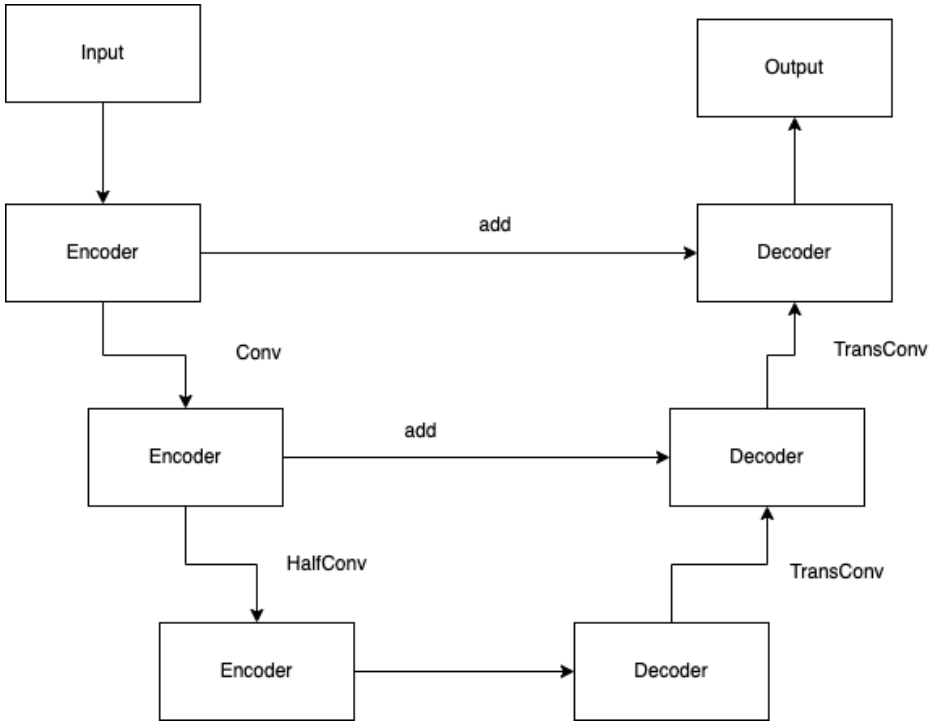


图 7 Encoder 设计

对于如上设计的 Encoder 和 Decoder，我们可以得到如下的数学表达：

假设原始经过编码后的时间序列为 X，包含 T 个时间步，每个时间步的特征维度为 d。我们采用具有 N 层的 Encoder-Decoder 结构进行时间序列预测。其中，Encoder-Decoder 结构中的 Encoder 部分可以设计如下：

$$h_0 = X$$
$$h_k = \text{Encoder}(h_{k-1}) \quad k = 1, 2, ..., N$$

其中，Encoder 表示 Transformer 中相同结构的 Encoder，h_{k-1} 表示第 k-1 层 Encoder 的输出。

在每一层 Encoder 的输出中，我们通过卷积操作对时间序列进行降采样，并保留

输出序列，如下所示：

$$
\begin{aligned}
h'_k &= \text{MaxPool1D}(\text{Conv1D}(h_k, F_k, S_k)) \\
h''_k &= \text{MaxPool1D}(\text{Conv1D}(h'_k, F'_k, S'_k)) \\
h_k &= h'_k[:T/2] + h''_k
\end{aligned}
$$

其中，Conv1D 为 1 维卷积操作，MaxPool1D 为 1 维池化操作，F_k 和 S_k 分别表示第 k 层 Encoder 卷积的滤波器大小和步长，F'_k 和 S'_k 分别表示第 k 层 Encoder 第二次卷积的滤波器大小和步长，$h'_k[:T/2]$ 表示 h'_k 序列的前一半。(这里假设卷积的滤波器大小和步长已经确定)

通过对 Encoder 每一层的输出进行卷积降采样，我们将 Encoder 的每一层的输入序列压缩了一半，从而减小了模型的运算量。同时，我们还保留了 Encoder 的所有输出序列，以便后续使用。

在 Decoder 阶段，我们对 Encoder 的输出进行反卷积操作，如下所示：

$$
\begin{aligned}
\hat{h}_N &= h_N \quad \text{(the last encoder output)} \\
\hat{h}_k &= \text{Conv1DTranspose}(\hat{h}_{k+1}, F'_k, S'_k) \\
h_{k-1} &= \hat{h}_k + h_{k-1} \quad k = N-1, N-2, ..., 1
\end{aligned}
$$

其中，Conv1DTranspose 为 1 维反卷积操作，F'_k 和 S'_k 分别表示第 k 层 Decoder 反卷积的滤波器大小和步长。通过反卷积操作，我们可以恢复之前被压缩的序列长度，并将 Encoder 的输出序列拼接后作为 Decoder 的输入。

因此，我们的设计中的 Encoder 可以通过卷积降采样和反卷积恢复操作来实现对序列的压缩和恢复，从而减小了模型的计算量，在保持模型性能的同时提高了模型的效率。

通过以上的改进方法，我们期望可以有效提升模型的预测性能，同时保证模型的稳定性，并尽可能降低其计算复杂度和参数数量。在接下来的实验部分，我们将对这些改进方法进行详细测试和验证。

5. 实验

5.1 数据集

我们采用如下三个公开的数据集，其来自于 Informer 中采用的标准数据集，同时也是目前研究长期序列问题的常用数据集。

1. ETT (Electricity Transformer Temperature)[1]：这个数据集是从中国的两个县收集的两年数据，包括两个不同的数据集，分别是 1 小时级别的 ETTh1 和 ETTh2，以及 15 分钟级别的 ETTm1 和 ETTm2。每个数据点包括目标值“油温”和 6 个功率负载特征。训练/验证/测试集分别为 12/4/4 个月。

[1] https://github.com/zhouhaoyi/ETDataset

2. ECL (Electricity Consuming Load)[2]：这个数据集收集了 321 个客户的电力消耗（千瓦时）。由于数据缺失，作者将数据集转换为 2 年内每小时的消耗，并将“MT 320”设置为目标值。训练/验证/测试集分别为 15/3/4 个月。

3. Weather[3]：这个数据集包含了近 1600 个美国地点的本地气候数据，时间跨度为 2010 年到 2013 年，每小时收集一次数据。每个数据点包括目标值“湿球温度”和 11 个气候特征。训练/验证/测试集分别为 28/10/10 个月[14]。

我们将我们的模型在这三个数据集上进行运行，并控制其他参数不变以进行比较实验。每个数据集都包括训练集、验证集和测试集，用于训练、验证和测试模型的性能。我们的代码开源在 github 上：https://github.com/happys2333/SCLformer

5.2 实验细节

5.2.1 实验设置

我们基于如下的输入参数进行实验：Encoder 的输入是一个长度为 seq len 的时间序列，Decoder 的输出由两部分组成：一部分是起始 token，它是一个长度为 label len 的时间序列片段，它可以是 Encoder 的最后一段数据；另一部分是一个全零的时间序列片段，它的长度为 pred len。我们还在这两部分的数据中添加了时间和位置信息，使其与输入数据的长度一致。这样，我们得到了一个长度为 label len+pred len 的时间序列片段，它作为 Decoder 的输入。

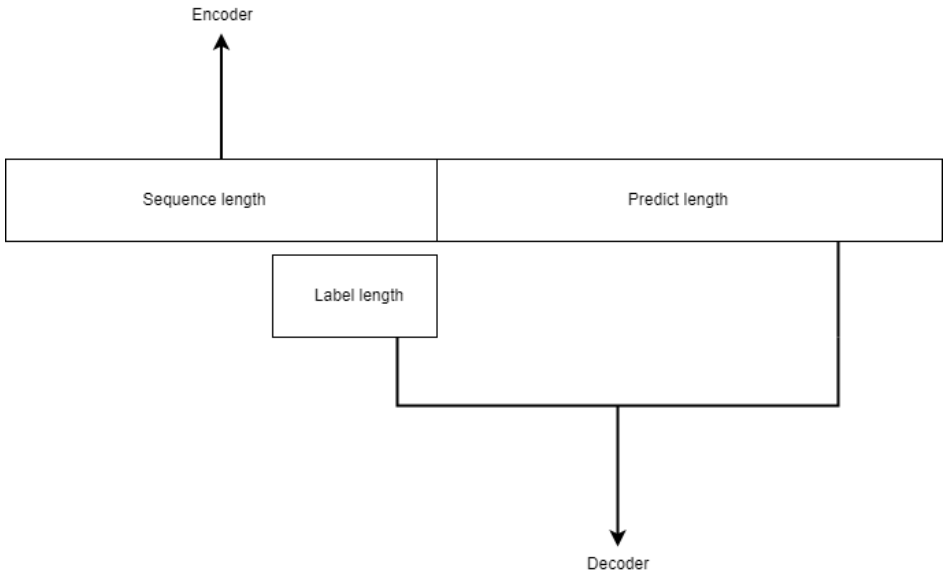


图 8 实验序列样例

[2]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[3]https://www.ncei.noaa.gov/data/local-climatological-data/

5.2.2 基准测试

我们选取了四种模型作为对照的 Baseline 模型，其中有 Transformer based 的 Informer[7]和其他传统模型，LSTM[15]，ARMA[16]，DeepAR[17]。我们将我们的模型与这些模型进行了对比，其中 Informer 是我们的主要对比模型。

为了更好的探究模型的自适应编码的效果，我们使用了两种 SCLformer 模型，一种是去掉自适应编码层的，另一种是使用自适应编码层的。我们的实验重点是测试不同预测长度下模型的效果表现，我们在不同的数据集上进行了测试，并获得了他们在测试集上的 MAE 和 MSE, 如下表 1,2 所示。我们对基准模型和我们的模型进行了测试，其中 SCLformer+ 代表了去掉自适应编码层的版本。

我们的方法是使用 L2 损失函数为训练的损失函数，使用 Adam 优化器[18]，所有数据训练的提前暂停周期为 10 个周期。。我们的检测效果使用 MSE 和 MAE 作为评价指标，其中 MSE 是均方误差，MAE 是平均绝对误差。其公式为：$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$，$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$。我们所有的代码都在 PyTorch 上编写实现，测试在一块 RTX 2080Ti 上进行。

5.3 实验结果

为了公平的比较每一个模型的性能，我们将每一个模型的参数设计都为相同的，尽可能排除因为超参不同而导致的效果问题。我们的实验结果如下表所示，我们将每一个模型的 MAE 和 MSE 都列出来。我们在不同数据集上都采用了相同的 Seq len 和 Label len，而 Pred len 采用了 $\{96, 192, 336, 720\}$ 四组数据，来测试不同模型在长期时间序列问题上的效果。由于 Transformer 本身的机制导致其在多变量问题下的效果和单变量效果有差异性，所以我们对每个数据集均采用了多变量和单变量两种情况下的测试机制，单变量情况下我们只输入需要预测的序列来预测，多变量我们会给予更多的信息，更多的数据维度来测试。

5.4 超参测试

我们测试了多种模型本身的超参数效果，得到更多的数据来测试我们模型的不同参数下的状态，我们将我们的模型的超参数设置为如下的几种情况，来测试不同的超参数对于模型的影响。

5.4.1 Encoder 层数

我们的模型核心是 Attention 和 Encoder 的改进，因此我们测试了不同的 Encoder 层数对于模型的影响，我们将 Encoder 层数设置为 $\{1, 2, 3, 4\}$ 四种情况，来测试不同的 Encoder 层数对于模型的影响。其中我们的测试数据选用了多变量数据，并分别测试了在不同的预测长度下的情况。

我们可以看出在四组测试的情况下，我们可以看出随着层数的增加，模型的效果会有所提升，且模型层数越多我们可以获取越好的效果，这证明在我们的模型中的信

表 1 多变量下的结果

Methods		SCLformer		SCLformer +		Informer		LSTM		ARMA		DeepAR	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.945	0.771	0.903	0.73	0.906	0.746	1.201	0.861	0.846	0.607	0.833	0.701
	192	0.948	0.75	1.024	0.797	0.843	0.694	1.23	0.831	0.861	0.62	0.939	0.749
	336	1.144	0.828	0.999	0.785	1.157	0.849	1.292	0.921	0.875	0.635	1.088	0.785
	720	1.064	0.797	1.065	0.835	1.231	0.887	1.126	0.839	0.884	0.653	1.078	0.846
ETTh2	96	1.918	1.119	1.767	1.097	2.723	1.404	2.193	1.179	3.154	1.353	2.283	1.196
	192	2.393	1.153	3.516	1.605	5.904	2.11	3.236	1.402	3.166	1.357	3.355	1.565
	336	3.122	1.432	3.56	1.586	3.511	1.553	2.533	1.274	3.149	1.351	2.681	1.405
	720	2.838	1.312	4.232	1.762	3.283	1.56	3.498	1.552	3.113	1.342	3.049	1.333
ETTm1	96	0.79	0.684	0.655	0.585	0.556	0.529	1.124	0.8222	0.865	0.619	0.779	0.701
	192	0.729	0.631	0.697	0.631	0.872	0.69	1.261	0.898	0.871	0.621	0.805	0.712
	336	1.015	0.77	0.865	0.715	0.933	0.739	1.14	0.843	0.882	0.631	0.83	0.724
	720	0.947	0.744	0.927	0.751	0.918	0.732	1.184	0.869	0.899	0.644	0.889	0.75
ETTm2	96	1.348	0.898	0.696	0.607	0.539	0.522	1.134	0.887	3.121	1.343	1.026	0.816
	192	1.366	0.89	1.083	0.753	1.05	0.769	2.366	1.299	3.128	1.346	1.498	0.952
	336	1.663	1.023	0.885	0.726	1.451	0.935	2.612	1.321	3.142	1.349	1.997	1.136
	720	2.405	1.241	1.835	1.042	2.441	1.172	3.127	1.508	3.152	1.352	2.368	1.212
Weather	96	0.615	0.578	0.755	0.665	0.488	0.497	0.554	0.548	0.618	0.557	0.487	0.5
	192	0.682	0.618	0.795	0.687	0.581	0.552	0.604	0.581	0.642	0.579	0.528	0.532
	336	0.7	0.632	0.676	0.613	0.604	0.571	0.617	0.589	0.647	0.588	0.546	0.547
	720	0.662	0.604	0.746	0.648	0.614	0.583	0.652	0.61	0.672	0.61	0.638	0.603
ECL	96	0.337	0.402	0.336	0.404	0.303	0.395	1.013	0.837	0.584	0.572	0.53	0.541
	192	0.306	0.38	0.311	0.39	0.291	0.378	0.932	0.793	0.582	0.574	0.492	0.522
	336	0.328	0.4	0.306	0.387	0.302	0.392	0.962	0.801	0.586	0.582	0.488	0.522
	720	0.324	0.393	0.314	0.395	0.381	0.447	0.953	0.804	0.603	0.599	0.505	0.514

表 2 单变量下的结果

Methods		SCLformer		SCLformer+		Informer		LSTM		ARMA		DeepAR	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.846	0.864	0.379	0.542	0.359	0.526	0.745	0.797	1.917	1.342	0.161	0.335
	192	0.517	0.646	0.402	0.553	0.423	0.567	0.462	0.608	1.938	1.352	0.195	0.372
	336	0.524	0.654	0.329	0.486	0.34	0.51	0.334	0.512	1.967	1.364	0.237	0.418
	720	0.783	0.831	0.925	0.916	0.303	0.487	0.938	0.921	2.02	1.389	0.338	0.519
ETTh2	96	0.604	0.658	0.22	0.38	0.212	0.365	2.873	1.584	1.549	1.109	0.164	0.321
	192	0.322	0.459	0.247	0.401	0.244	0.396	0.333	0.468	1.576	1.121	0.206	0.358
	336	0.363	0.493	0.238	0.396	0.249	0.406	0.344	0.474	1.607	1.135	0.238	0.387
	720	2.302	1.405	0.244	0.403	0.284	0.434	1.846	1.241	1.628	1.161	0.317	0.46
ETTm1	96	0.312	0.481	0.191	0.364	0.156	0.305	0.057	0.192	1.91	1.339	0.061	0.197
	192	0.352	0.5	0.315	0.485	0.258	0.41	0.112	0.288	1.912	1.34	0.136	0.309
	336	0.454	0.58	0.343	0.501	0.375	0.525	0.199	0.382	1.917	1.342	0.177	0.356
	720	0.392	0.546	0.412	0.554	0.353	0.509	0.243	0.427	1.937	1.352	0.252	0.435
ETTm2	96	0.319	0.459	0.134	0.285	0.113	0.266	0.161	0.323	1.537	1.102	0.079	0.211
	192	0.254	0.412	0.597	0.657	0.169	0.326	0.226	0.385	1.541	1.105	0.121	0.268
	336	0.291	0.44	0.239	0.398	0.211	0.371	0.261	0.416	1.549	1.109	0.173	0.329
	720	0.276	0.421	0.269	0.419	0.261	0.415	0.399	0.517	1.576	1.122	0.255	0.401
Weather	96	0.253	0.375	0.24	0.369	0.209	0.329	0.241	0.366	0.413	0.428	0.188	0.316
	192	0.35	0.436	0.503	0.568	0.245	0.364	0.269	0.389	0.424	0.444	0.233	0.355
	336	0.4	0.467	0.313	0.441	0.301	0.446	0.301	0.421	0.424	0.455	0.275	0.394
	720	0.864	0.778	0.281	0.394	0.25	0.378	0.379	0.476	0.448	0.489	0.354	0.455
ECL	96	0.999	0.734	0.309	0.418	0.294	0.401	0.971	0.796	0.593	0.623	0.376	0.443
	192	0.686	0.704	0.328	0.423	0.306	0.403	0.957	0.791	0.594	0.623	0.359	0.431
	336	0.663	0.595	0.407	0.479	0.364	0.442	0.961	0.792	0.609	0.631	0.378	0.442
	720	0.77	0.656	0.409	0.477	0.373	0.457	1.007	0.809	0.614	0.639	0.419	0.48

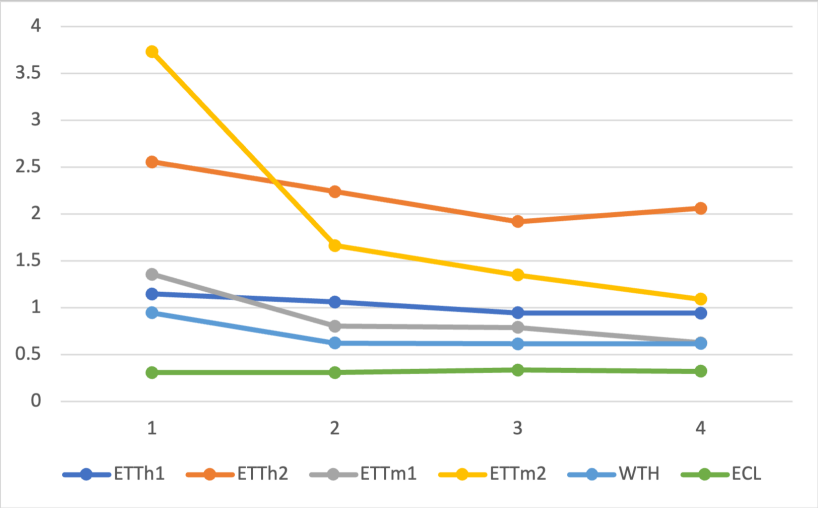


图 9 多变量下 Encoder 层数对 MSE 情况 (预测长度 96)

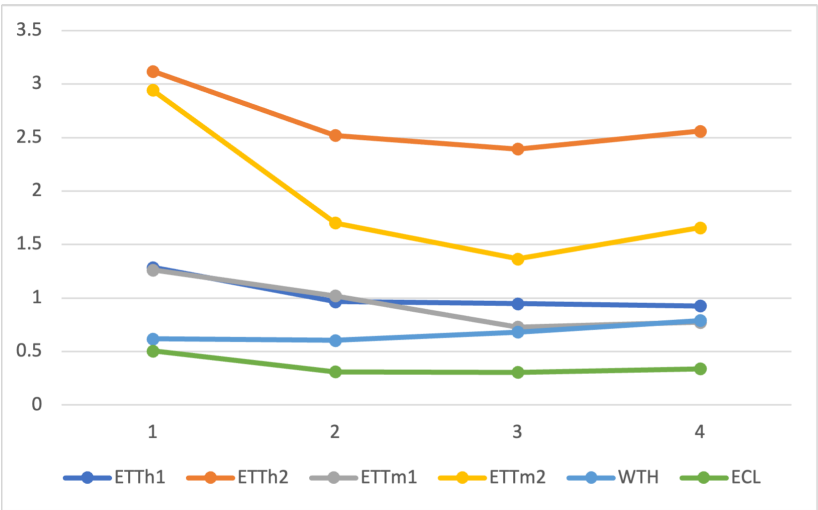


图 10 多变量下 Encoder 层数对 MSE 情况 (预测长度 192)

息保留较好并没有因为我们的卷积操作而丢失过多信息。

5.4.2 GRU 的隐藏层层数

在我们的模型中，另一个核心是自适应的时间位置编码。其中由于我们使用了一个 GRU 模型，这使得其隐藏层数与我们的效果有关，我们测试了不同的 GRU 隐藏层数对于模型的影响，我们将 GRU 隐藏层数设置为 $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ 十种情况，来测试不同的 GRU 隐藏层数对于模型的影响。其中我们的测试数据选用了多变量数据。

我们的数据基本上可以看出，隐藏层在 6-8 的时候效果较好，但是在 8 层之后效果会有所下降，这是由于 GRU 的隐藏层过多会导致其历史信息的丢失。

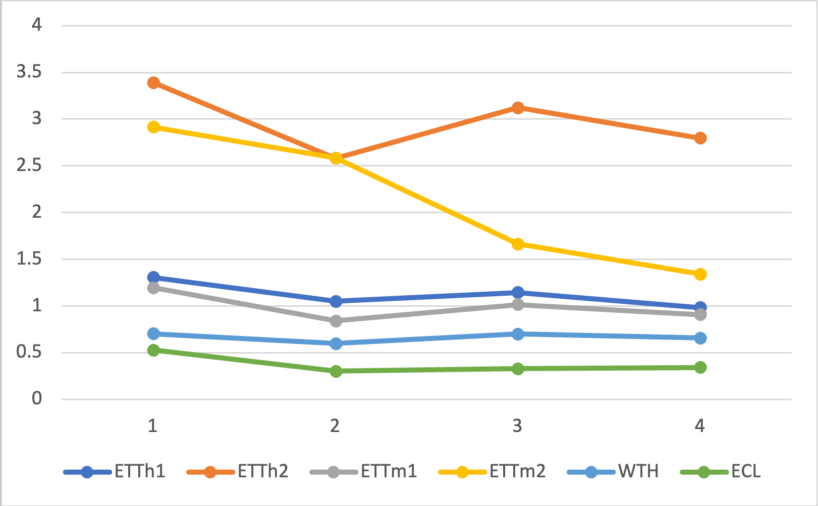


图 11 多变量下 Encoder 层数对 MSE 情况 (预测长度 336)

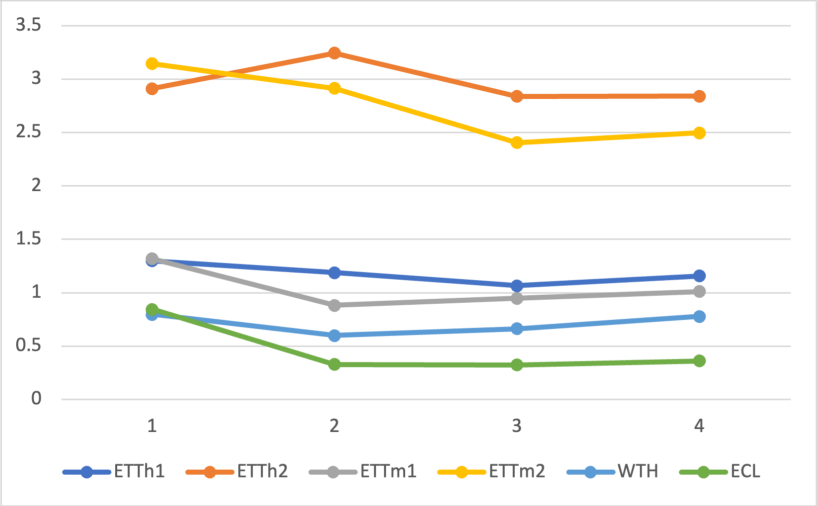


图 12 多变量下 Encoder 层数对 MSE 情况 (预测长度 720)

5.4.3 seq len 长度

实验中我们也实验了有关输入序列长度对数据结果的影响，我们将 seq len 设置为 $\{96, 192, 336, 720\}$ 四种情况，来测试不同的 seq len 对于模型的影响。其中我们的测试数据选用了多变量和单变量数据。

我们可以看见在 seq len 为 720 的时候效果较好，我们分析可能是由于这样给模型的 Encoder 的数据量较大，可以让模型更加全面化的分析数据，从而实现更好的效果。

5.4.4 label len 的长度

label len 是我们的模型中的一个超参数，代表了先验序列长度，我们将 label len 设置为 $\{24, 48, 96\}$ 三种情况，来测试不同的 label len 对于模型的影响。其中我们的

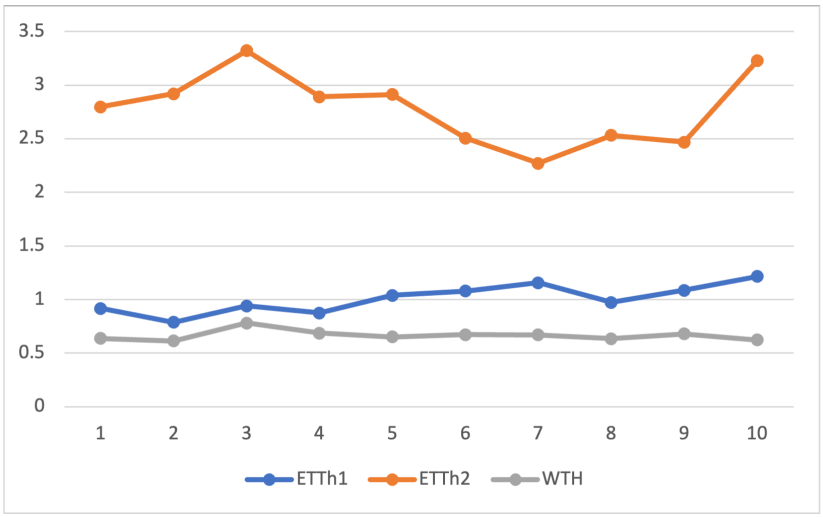


图 13 多变量下隐藏层数的 MSE 情况

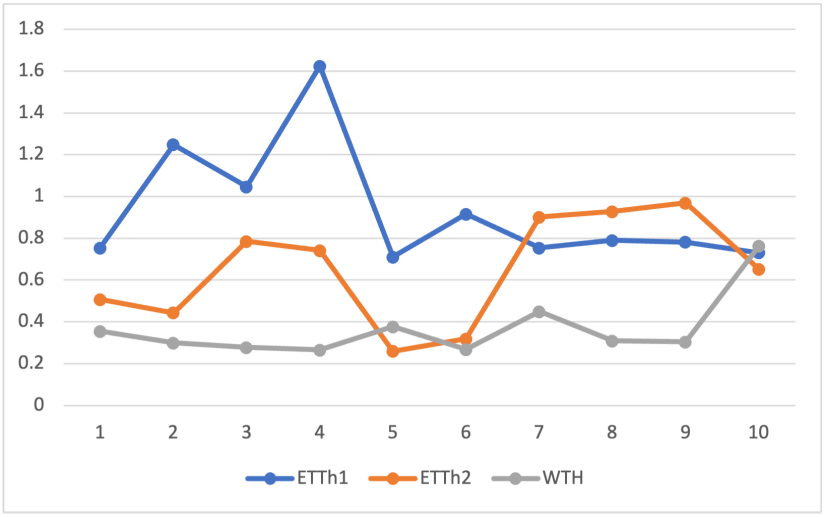


图 14 单变量下隐藏层数的 MSE 情况

测试数据选用了多变量数据。

我们可以看见在 label len 为 48 的时候效果较好，我们分析可能是由于单次取样此时能被传统的时间模型很好的兼容，相对的长度较短，方便后续的生成，同时也能兼顾更多的信息。

5.4.5 多头注意力机制头数

由于使用的 Transformer 中的 Multi head Self-Attention 机制，所以我们测试了不同头数在多变量情况下的效果，我们将多头注意力机制头数设置为 $\{3, 6, 7, 8, 12, 14, 16, 32\}$ 八种情况，来测试不同的多头注意力机制头数对于模型的影响。其中我们的测试数据选用了多变量和单变量数据。

在单变量和多变量情况下头数的影响不一样，在多变量下 8 为较优解，而在单变

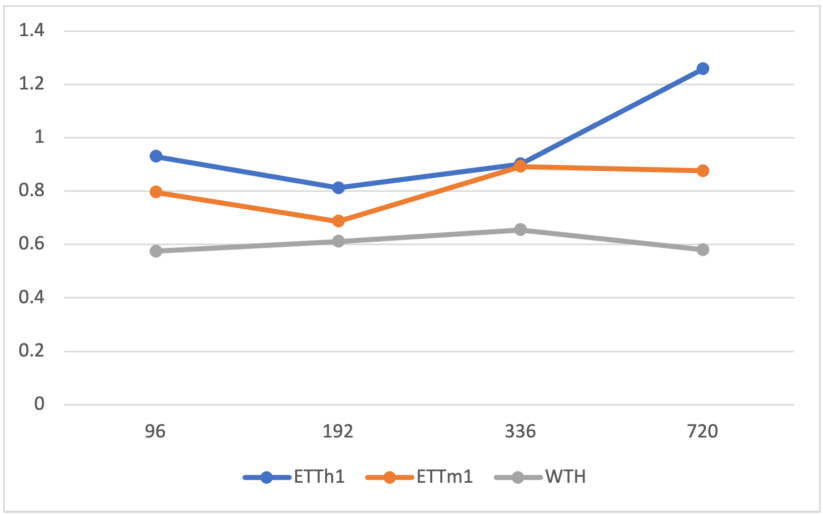


图 15 多变量下 seq len 的 MSE 情况

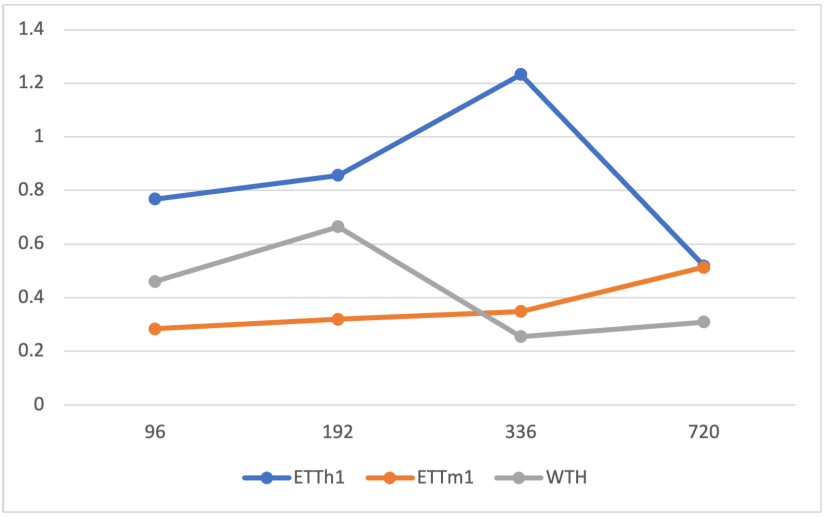


图 16 单变量下 seq len 的 MSE 情况

量下 16 为较优解，这可能是由于多变量下的数据更加复杂，更少的头数就可以捕捉到足够多的信息，而单变量下的数据相对简单，所以需要更多的头数来捕获数据中的关系。

5.4.6 采样因子数

由于我们的模型里面也有 Informer 中的 ProbSparse Attention 机制，所以我们测试了不同采样因子数在多变量情况下的效果，我们将采样因子数设置为 $\{3, 5, 8, 10\}$ 四种情况，来测试不同的采样因子数对于模型的影响。其中我们的测试数据选用了多变量数据。

我们发现在采样因子数为 3 的时候能更好捕获序列中的关系，这可能是由于采样因子数过大会导致模型过于稀疏。

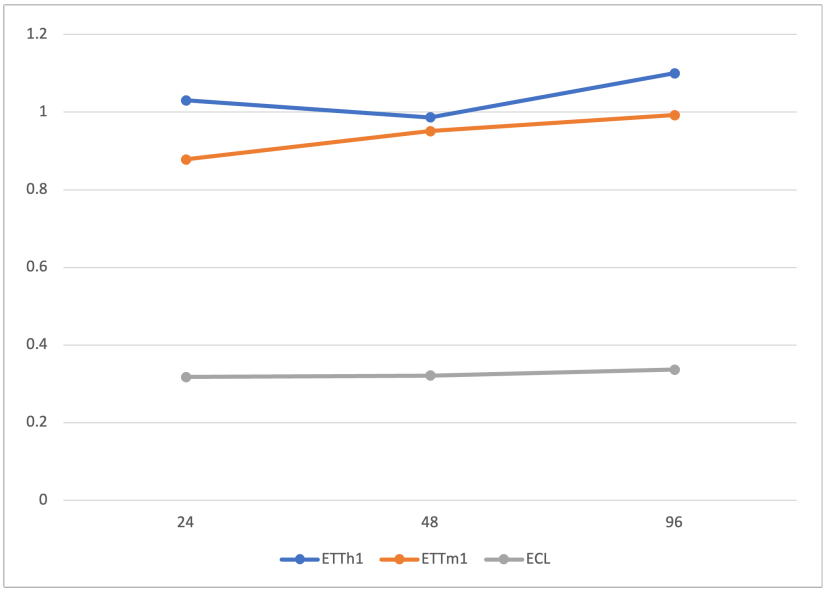


图 17 多变量下 label len 的 MSE 情况

5.4.7 模型宽度

Transformer 的模型宽度决定其对时间的理解能力，我们测试了不同模型宽度在多变量情况下的效果，我们将模型宽度设置为 $\{64, 128, 256, 512, 1024\}$ 五种情况，来测试不同的模型宽度对于模型的影响。其中我们的测试数据选用了多变量数据。

我们发现在取 512 宽度的时候效果较好，这可能是由于模型宽度过小会导致模型的信息量不足，而模型宽度过大会导致模型的信息量过大，从而导致可能出现很容易过拟合。

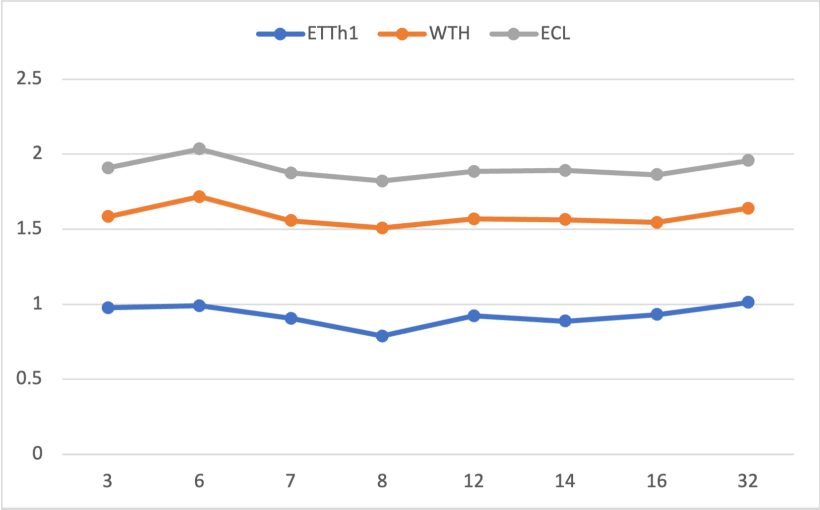


图 18 多变量下多头注意力的头数的 MSE 情况

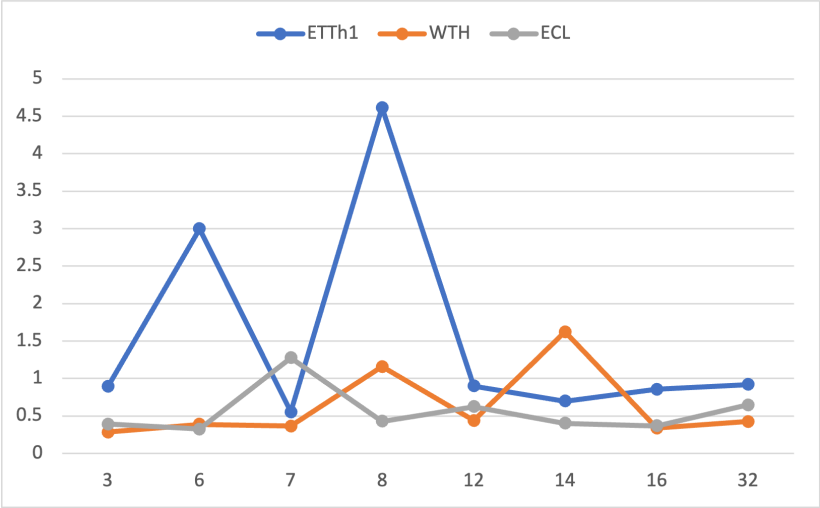


图 19 单变量下多头注意力的头数的 MSE 情况

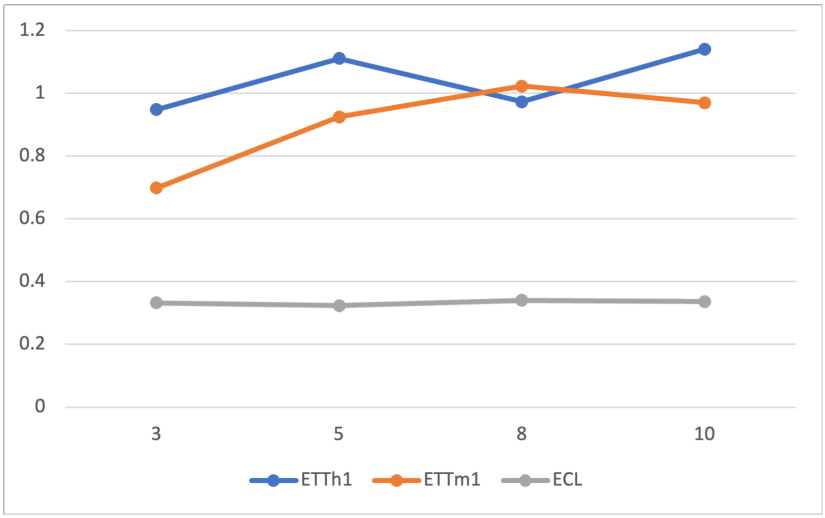


图 20 多变量下采样因子数的 MSE 情况

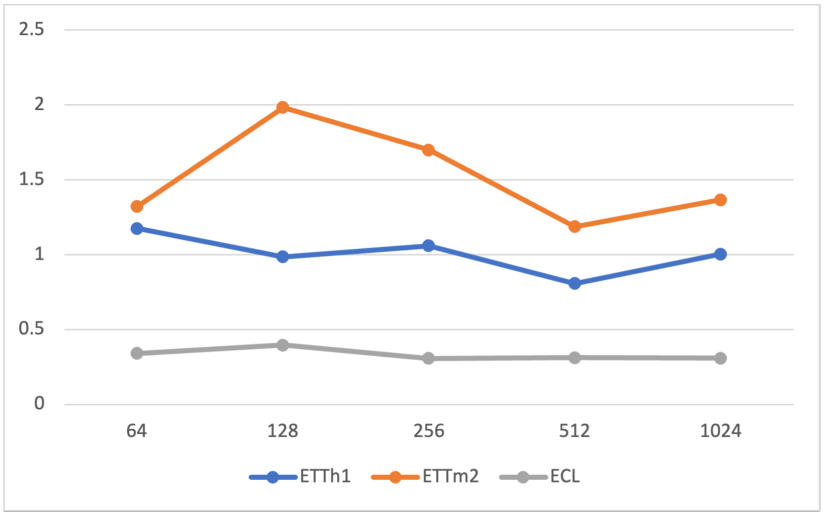


图 21 多变量下模型宽度数的 MSE 情况

6. 结论

本文提出了一种新的长期时间序列预测模型，它基于 Transformer 的结构进行进一步的改进，能够有效地捕捉长期依赖关系和复杂的非线性模式。我们在多个真实世界的数据集上对我们的模型进行了实验，与相关研究领域先进的 Informer 模型进行了对比。实验结果表明，我们的模型在长期时间预测上有着一定的优势，尤其是在多变量预测的场景下。我们认为，本模型能够更好地适应长期时间序列的特点，可以提高预测的准确性。在今后的研究中，我们希望将模型应用在更多的长期序列上，如气候变化、股票市场、交通流量等问题，以帮助模型进一步提升鲁棒性，并探索更多的应用场景和价值。同时也希望进一步的帮助改进模型的设计结构，提高模型的效果，更加能适应长期序列预测的条件。

参考文献

[1] ZHANG G P. Time series forecasting using a hybrid ARIMA and neural network model[J]. Neurocomputing, 2003, 50: 159-175.

[2] CHUNG J, GULCEHRE C, CHO K, et al. Gated feedback recurrent neural networks[C]//International conference on machine learning. 2015: 2067-2075.

[3] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[4] YANG B, WANG L, WONG D, et al. Convolutional self-attention networks[J]. ArXiv preprint arXiv:1904.03107, 2019.

[5] TU Z, LIU Y, SHI S, et al. Modeling Localness for Self-Attention Networks[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 4449-4458.

[6] DAI Z, YANG Z, YANG Y, et al. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. 2019.

[7] ZHOU H, ZHANG S, PENG J, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting[C]//Proceedings of the AAAI conference on artificial intelligence: vol. 35: 12. 2021: 11106-11115.

[8] ZENG A, CHEN M, ZHANG L, et al. Are transformers effective for time series forecasting?[J]. ArXiv preprint arXiv:2205.13504, 2022.

[9] TAY Y, BAHRI D, METZLER D, et al. Synthesizer: Rethinking self-attention in transformer models. arxiv 2020[J]. ArXiv preprint arXiv:2005.00743, 2020, 2.

[10] TOLSTIKHIN I O, HOULSBY N, KOLESNIKOV A, et al. Mlp-mixer: An all-mlp architecture for vision[J]. Advances in neural information processing systems, 2021, 34: 24261-24272.

[11] LI Z, RAO Z, PAN L, et al. MTS-Mixers: Multivariate Time Series Forecasting via Factorized Temporal and Channel Mixing[J]. ArXiv preprint arXiv:2302.04501, 2023.

[12] MADHUSUDHANAN K, BURCHERT J, DUONG-TRUNG N, et al. Yformer: U-net inspired transformer architecture for far horizon time series forecasting[J]. ArXiv preprint arXiv:2110.08255, 2021.

[13] HUANG H, LIN L, TONG R, et al. Unet 3+: A full-scale connected unet for medical image segmentation[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2020: 1055-1059.

[14] WU H, XU J, WANG J, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting[J]. Advances in Neural Information Processing Systems, 2021, 34: 22419-22430.

[15] SHI X, CHEN Z, WANG H, et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting[J]. Advances in neural information processing systems, 2015, 28.

[16] BOX G E, JENKINS G M, REINSEL G C, et al. Time series analysis: forecasting and control[M]. John Wiley & Sons, 2015.

[17] SALINAS D, FLUNKERT V, GASTHAUS J, et al. DeepAR: Probabilistic forecasting with autoregressive recurrent networks[J]. International Journal of Forecasting, 2020, 36(3): 1181-1191.

[18] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. ArXiv preprint arXiv:1412.6980, 2014.

[19] STOLLER D, TIAN M, EWERT S, et al. Seq-u-net: A one-dimensional causal u-net for efficient sequence modelling[J]. ArXiv preprint arXiv:1911.06393, 2019.

[20] WANG S, LI B Z, KHABSA M, et al. Linformer: Self-attention with linear complexity[J]. ArXiv preprint arXiv:2006.04768, 2020.

[21] LI S, JIN X, XUAN Y, et al. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting[J]. Advances in neural information processing systems, 2019, 32.

[22] LIM B, ARIK S Ö, LOEFF N, et al. Temporal fusion transformers for interpretable multi-horizon time series forecasting[J]. International Journal of Forecasting, 2021, 37(4): 1748-1764.

[23] BAI S, KOLTER J Z, KOLTUN V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[J]. ArXiv preprint arXiv:1803.01271, 2018.

[24] LIU S, YU H, LIAO C, et al. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting[C]//International conference on learning representations. 2021.

[25] DAS A, KONG W, LEACH A, et al. Long-term Forecasting with TiDE: Time-series Dense Encoder[J]. ArXiv preprint arXiv:2304.08424, 2023.

[26] ZHOU T, MA Z, WEN Q, et al. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting[C]//International Conference on Machine Learning. 2022: 27268-27286.

[27] SHUKLA S N, MARLIN B M. Heteroscedastic Temporal Variational Autoencoder For Irregularly Sampled Time Series[J]. ArXiv preprint arXiv:2107.11350, 2021.

[28] WOO G, LIU C, SAHOO D, et al. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting[J]. ArXiv preprint arXiv:2202.01575, 2022.

[29] KIM T, KIM J, TAE Y, et al. Reversible instance normalization for accurate time-series forecasting against distribution shift[C]//International Conference on Learning Representations. 2021.

[30] AL-QANESS M A, DAHOU A, EWEES A A, et al. ResInformer: Residual Transformer-Based Artificial Time-Series Forecasting Model for PM2. 5 Concentration in Three Major Chinese Cities[J]. Mathematics, 2023, 11(2): 476.