Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting

Haixu Wu, Jiehui Xu, Jianmin Wang, Mingsheng Long
School of Software, BNRist, Tsinghua University, China

摘要

长期序列预测是极端天气预警和长期能源消耗规划等实际应用的关键需求，本论文研究关于时间序列的长期预测问题。以往的 Tranformer 模型采用各种自注意力机制来发现长期依赖关系，然而复杂的时间模式阻碍了模型找到可靠的相关性，Tranformer 还采用了点对点自关注的稀疏形式，提高了长期效率但也造成了信息利用的瓶颈。针对这些挑战，论文中提出了一种具有自相关机制的新型分解结构 Autoformer，打破了传统序列分解的预处理惯例，将其改造成深层模型的基本内块，这种设计使 Autoformer 具有对复杂时间序列的渐进分解能力。另外，受随机过程理论的启发，作者设计了基于序列周期性的自相关机制，在子序列层次上进行依赖项发现和表示聚合。在长期预测中，Autoformer 的准确率达到了最先进的水平，在六个基准测试上相对提高了 38%，涵盖了五个实际应用：能源、交通、经济、天气和疾病。

关键词：长期序列预测；深度分解；自相关机制

1 引言

时间序列预测已广泛应用于能源消耗、交通经济规划、天气和疾病传播预测等领域，在这些实际应用中，将预报时间延长到更遥远的未来是目前一个较为迫切的需求，这对长期规划和预警具有重要意义。因此，论文研究了时间序列的长期预测问题，其特点是预测的时间序列长度大。最近的深度预测模型已经取得了很大的进展，特别是基于 Transformer 的模型，得益于自注意力机制，Transformer 在对顺序数据的长期依赖关系建模方面具有很大的优势，这使得大模型会更加强大。

然而，在长时间的条件下，预测任务极具挑战性。首先，直接从长期时间序列中发现时间依赖关系是不可靠的，因为依赖关系可能被纠缠的时间模式所掩盖。其次，由于序列长度的二次复杂度，具有自注意力机制的标准 Transformer 模型在计算上无法进行长期预测。以往基于 Transformer 的预测模型主要侧重于提高对稀疏版本的自关注，虽然性能得到了显著提高，但这些模型仍然使用逐点表示聚合。因此，在提高效率的过程中，由于稀疏的点连接，会牺牲信息的利用率，导致时间序列的长期预测遇到瓶颈。

为了获取序列中复杂的时间模式信息，作者尝试采用分解的思想，这是时间序列分析的标准方法，它可以用来处理复杂的时间序列，提取更多可预测的分量。但是，由于未来是未知的，它只能作为过去序列的预处理，这种常见的用法限制了分解的功能，并忽略了被分解的组件之间潜在的未来交互。因此，作者试图突破序列分解的预处理使用，并提出一种通用架构，使深度预测模型具有内在的渐进分解能力，可以解开时间序列的纠缠时间模式，突出时间序列的固有属性。作者利用序列的周期性来更新自注意力机制中的点连接，在同一相位位置的子序列在周期之间往往表现出相似的时间过程，基于此构建出一个序列级连接。

从以上出发，作者提出了一种新的深度分解架构模型 Autoformer 代替 Transformer 进行长期时间序列预测，引入一种自相关机制代替自注意力机制，内部的分解模块也能逐步从预测的隐藏变量中分离出长期趋势信息，最后实现更好的预测效果。

2 相关工作

2.1 Informer

Informer[1]的目标是解决长序列持续预测问题, 自注意力的计算/内存开销是随输入/输出的序列长度呈二次相关的，这导致大规模的 Transformer 模型必须使用大量计算资源，昂贵的训练和部署成本阻碍了模型的应用；同时这也会限制 Transformer 模型对于长序列数据的处理能力。

Informer 基于自注意力机制中存在的查询稀疏性（attention 的长尾分布），选择 top-u 进行 query-key 对的部分计算，提出了 ProbSparse Self-Attention 替代标准的 Self-Attention，将自注意力机制的内存和计算开销从 $\mathcal{O}(n^2)$ 减少到 $\mathcal{O}(L \log L)$。同时，Informer 提出在时序问题下使用自注意力蒸馏机制，每层 Encdoer 都将输入序列的长度减小一半，从而大大减小了 Encoder 内存开销和计算时间，在 Decoder 结构中使用生成式结构，能够一次生成全部预测序列，极大减小了预测解码耗时。

2.2 Reformer

Transformer 模型已经在许多任务上取得了令人欣喜的成绩，但是当扩展到更长的上下文窗口时会遇到许多限制。更大的上下文窗口意味着模型能力变得更加强大，但也会让其变得不够高效，消耗更多的内存。Reformer[2]模型结合了两个至关重要的技术来解决限制 Transformer 应用到长上下文窗口的注意力和内存分配的问题。Reformer 使用位置敏感哈希（Locality-Sensitive-Hashing, LSH) 来减少关注长序列的复杂度，使用可逆残差层来更加高效地利用可用的内存。

当将 Transformer 应用到一个非常大的文本序列时，首要的挑战就是如何处理注意力层。LSH 通过计算哈希函数解决了这个问题，LSH 把相似的向量匹配在一起，而不是在所有的可能的向量对上进行搜索。尽管 LSH 解决了注意力问题，但是仍然有内存问题。在 Reformer 中实现的第二种新颖的方法就是在反向传播期间按需重新计算每一层的输入，而不是将其存储在内存中。这可以通过使用可逆层来实现，其中来自网络最后一层的激活函数值用来恢复任何中间层的激活函数值，这相当于反向运行网络。

2.3 LogSparse Transformer

该方法同样是使用 Transformer 模型解决时间序列预测问题，解决了 Transformer 的两个弱点：（1）局部不可知：标准 Transformer 架构中的逐点点积自注意力机制对局部上下文不敏感，这会使模型容易在时间序列中出现异常；（2）内存瓶颈：标准 Transformer 随序列长度 L 的平方增长，这导致无法直接对长时间序列进行建模。

为了解决上述问题，作者提出了 LogSparse Transformer[3]架构，在自注意力层使用了卷积注意力机制，通过因果卷积生成 query 和 key，蕴含局部上下文信息的 query-key 对匹配可以帮助模型实现更低的训练损失，并进一步提高其预测精度。同时，该网络的空间复杂度只有 $\mathcal{O}(L \log L)$，打破了内存瓶颈使长时间序列建模可行，可以用更少的内存使用产生类似甚至更好的结果。

3 本文方法

3.1 本文方法概述

论文提出的 Autoformer 全面革新 Transformer 为深度分解架构，包括内部的序列分解单元、自相关机制以及对应的编-解码器，具体结构如图1所示：

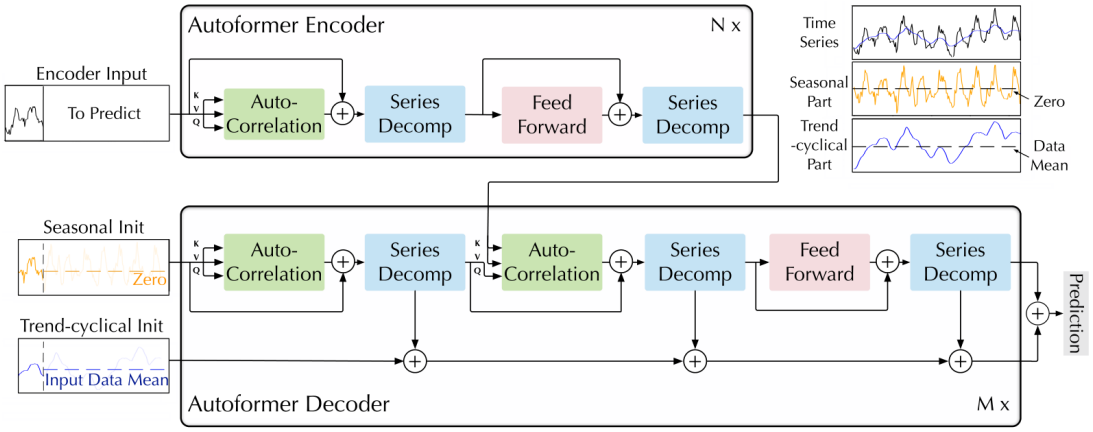

图 1: 深度分解架构

3.2 时间序列分解

时间序列分解是指将时间序列分解为几个组分，每个组分表示一类潜在的时间模式，如周期项（seasonal），趋势项（trend-cyclical）。由于预测问题中未来的不可知性，通常先对过去序列进行分解，再分别预测。但这会造成预测结果受限于分解效果，并且忽视了未来各个组分之间的相互作用。

深度分解架构将序列分解作为 Autoformer 的一个内部单元，嵌入到编-解码器中。在预测过程中，模型交替进行预测结果优化和序列分解，即从隐变量中逐步分离趋势项与周期项，实现渐进式分解。

序列分解单元（series decomposition block）基于滑动平均思想，平滑周期项、突出趋势项：

$$\begin{aligned} \mathcal{X}_{\mathrm{t}} &= \mathrm{AvgPool}(\ \mathrm{Padding}\ (\mathcal{X})) \\ \mathcal{X}_{\mathrm{s}} &= \mathcal{X} - \mathcal{X}_{\mathrm{t}}, \end{aligned} \tag{1}$$

其中，$\mathcal{X}$ 为待分解的隐变量，$\mathcal{X}_{\mathrm{t}}$ 和 $\mathcal{X}_{\mathrm{s}}$ 分别为趋势项和周期项，公式记为 $\mathcal{X}_t, \mathcal{X}_s = \mathrm{SeriesDecomp}(\mathcal{X})$，将上述序列分解单元嵌入 Autoformer 层间。

3.3 编-解码器

在编码器部分，我们逐步消除趋势项（这部分会在 jie'ma'q 中通过累积得到），得到周期项 $\mathcal{S}_{\mathrm{en}}^{l,1}$，$\mathcal{S}_{\mathrm{en}}^{l,2}$。而基于这种周期性，我们设计自相关机制，聚合不同周期的相似子过程，实现信息聚合：

$$\begin{aligned} \mathcal{S}_{\mathrm{en}}^{l,1}, \_ &= \mathrm{SeriesDecomp}\left(\ \mathrm{AutoCorrelation}\ \left(\mathcal{X}_{\mathrm{en}}^{l-1}\right) + \mathcal{X}_{\mathrm{en}}^{l-1}\right) \\ \mathcal{S}_{\mathrm{en}}^{l,2}, \_ &= \mathrm{SeriesDecomp}\left(\ \mathrm{FeedForward}\ \left(\mathcal{S}_{\mathrm{en}}^{l,1}\right) + \mathcal{S}_{\mathrm{en}}^{l,1}\right), \end{aligned} \tag{2}$$

在解码器部分，我们对趋势项与周期项分别建模。其中，对于周期项，自相关机制利用序列的周期性质，聚合不同周期中具有相似过程的子序列；对于趋势项，我们使用累积的方式，逐步从预测的

隐变量中提取出趋势信息（最后一行）：

$$
\begin{aligned}
\mathcal{S}_{\text{de}}^{l,1}, \mathcal{T}_{\text{de}}^{l,1} &= \text{SeriesDecomp}\left(\text{AutoCorrelation}\left(\mathcal{X}_{\text{de}}^{l-1}\right) + \mathcal{X}_{\text{de}}^{l-1}\right) \\
\mathcal{S}_{\text{de}}^{l,2}, \mathcal{T}_{\text{de}}^{l,2} &= \text{SeriesDecomp}\left(\text{AutoCorrelation}\left(\mathcal{S}_{\text{de}}^{l,1}, \mathcal{X}_{\text{en}}^{N}\right) + \mathcal{S}_{\text{de}}^{l,1}\right) \\
\mathcal{S}_{\text{de}}^{l,3}, \mathcal{T}_{\text{de}}^{l,3} &= \text{SeriesDecomp}\left(\text{FeedForward}\left(\mathcal{S}_{\text{de}}^{l,2}\right) + \mathcal{S}_{\text{de}}^{l,2}\right) \\
\mathcal{T}_{\text{de}}^{l} &= \mathcal{T}_{\text{de}}^{l-1} + \mathcal{W}_{l,1} * \mathcal{T}_{\text{de}}^{l,1} + \mathcal{W}_{l,2} * \mathcal{T}_{\text{de}}^{l,2} + \mathcal{W}_{l,3} * \mathcal{T}_{\text{de}}^{l,3},
\end{aligned}
\tag{3}
$$

基于上述渐进式分解架构，模型可以在预测过程中逐步分解隐变量，并通过自相关机制、累积的方式分别得到周期、趋势组分的预测结果，实现分解、预测结果优化的交替进行、相互促进。

3.4 自相关机制

作者提出自相关机制来实现高效的序列级连接，从而扩展信息效用。观察到，不同周期的相似相位之间通常表现出相似的子过程，可以利用这种序列固有的周期性来设计自相关机制，其中，包含基于周期的依赖发现（Period-based dependencies）和时延信息聚合（Time delay aggregation）。

基于周期的依赖发现：基于随机过程理论，对于实离散时间过程 $\{\mathcal{X}_t\}$，可以如下计算其自相关系数 $\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau)$：

$$
\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau) = \lim_{L\to\infty} \frac{1}{L} \sum_{t=0}^{L-1} \mathcal{X}_t \mathcal{X}_{t-\tau} \tag{4}
$$

其中，自相关系数 $\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau)$ 表示序列 $\{\mathcal{X}_t\}$ 与它的 τ 延迟 $\{\mathcal{X}_{t-\tau}\}$ 之间的相似性。我们将这种时延相似性看作未归一化的周期估计的置信度，即周期长度 τ 为的置信度为 $\mathcal{R}(\tau)$。

时延信息聚合：为了实现序列级连接，需要将相似的子序列信息进行聚合。依据估计出的周期长度，首先使用 Roll() 操作进行信息对齐，再进行信息聚合，这里依然使用 query、key、value 的形式，从而可以无缝替代自注意力机制。

$$
\begin{aligned}
\tau_1, \cdots, \tau_k &= \arg\operatorname*{Topk}_{\tau\in\{1,\cdots,L\}} \left(\mathcal{R}_{\mathcal{Q},\mathcal{K}}(\tau)\right) \\
\widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}\left(\tau_1\right), \cdots, \widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}\left(\tau_k\right) &= \text{SoftMax}\left(\mathcal{R}_{\mathcal{Q},\mathcal{K}}\left(\tau_1\right), \cdots, \mathcal{R}_{\mathcal{Q},\mathcal{K}}\left(\tau_k\right)\right) \\
\text{AutoCorrelation}\left(\mathcal{Q},\mathcal{K},\mathcal{V}\right) &= \sum_{i=1}^{k} \text{Roll}\left(\mathcal{V},\tau_k\right) \widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}\left(\tau_k\right)
\end{aligned}
\tag{5}
$$

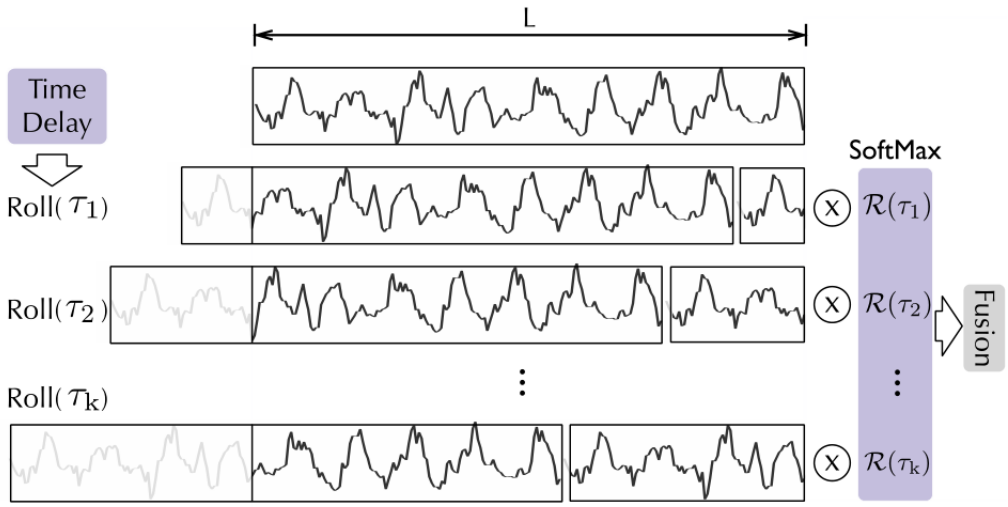

图 2: 时延信息聚合

其中，挑选出最有可能的 $k = \lfloor c \times \log L \rfloor$ 个周期长度，用于避免挑选到无关、甚至相反的相位。

在 Autoformer 中，还使用了多头（multi-head）版本, 如图3：。

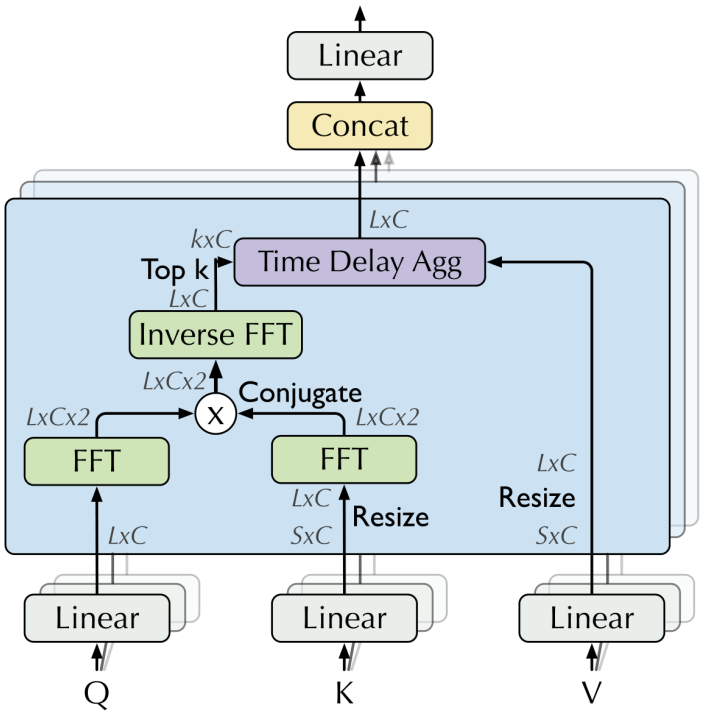

图 3: 多头自相关机制

为了高效计算，基于 Wiener-Khinchin 理论，自相关系数 $\mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau)$ 使用快速傅立叶变换（FFT）得到，计算过程（图 3）如下：

$$\begin{aligned} \mathcal{S}_{\mathcal{X}\mathcal{X}}(f) &= \mathcal{F}\left(\mathcal{X}_t\right) \mathcal{F}^*\left(\mathcal{X}_t\right) = \int_{-\infty}^{\infty} \mathcal{X}_t e^{-i2\pi tf} \, \mathrm{d}t \int_{-\infty}^{\infty} \mathcal{X}_t e^{-i2\pi tf} \, \mathrm{d}t \\ \mathcal{R}_{\mathcal{X}\mathcal{X}}(\tau) &= \mathcal{F}^{-1}\left(\mathcal{S}_{\mathcal{X}\mathcal{X}}(f)\right) = \int_{-\infty}^{\infty} \mathcal{S}_{\mathcal{X}X}(f) e^{i2\pi f\tau} \mathrm{d}f \end{aligned} \tag{6}$$

4 复现细节

4.1 与已有开源代码对比

论文开源了代码https://github.com/thuml/Autoformer。

代码中不止包含论文中提出的 Autoformer 网络，作者还集合了 Informer、Reformer、Transformer 模型。我参考源代码和论文框架图，把 Autoformer 的部分按自己的理解和代码习惯单独重新构建，去掉冗杂的例如混合精度训练、蒸馏等机制，将命令行输入参数改为用字典存储，直接在程序中修改并删除了不必要的参数设置，使模型更为简洁。

4.2 实验环境搭建

创建 conda 环境，安装 python=3.7，pytorch=1.12.1，以及各种所需要的包：matplotlib、numpy、pandas、scikit-learn 等，实验使用单 GPU 训练，显卡为 NVIDIA GeForce RTX 3070 Ti。

4.3 创新点

论文的时序拆解模块本质来自于传统的时序预测算法，对于时间序列进行拆解，并赋予拆解的子项以不同的物理意义例如：趋势项，季节项，残差项等，论文也使用最普遍的加法模型，用一个均值滤波器处理时序得到趋势项，并仅仅拆解出两个子项：趋势项和季节项，这一部分有很多可以深挖的部分，例如使用更复杂的混合模型，以及拆解出多个子项。

我尝试使用基于 LOESS（局部加权回归）的 STL 分解将时间序列分解成趋势项、季节项和余项。方法是在预测变量附近取一个数据子集，然后对该子集进行线性回归或二次回归，回归时采用加权最小二乘法，即越靠近估计点的值其权重越大，最后利用得到的局部回归模型来估计响应变量的值，用这种方法进行逐点运算得到整条拟合曲线。在模型中后续对余项的处理和季节项一样，最终效果没有较好的提升反而有所下降。

Autoformer 的核心是序列分解模块和 Auto-correlation 机制，Auto-correlation 关注的是寻找时序数据的周期特性，它希望模型能够更好的记住时序的周期特性，从而预测时序的未来值。STL 分解可能与 Auto-correlation 机制结合效果不好，余项的加入引入了更多的噪声，但这思路方向是正确的，找到一个更好的序列分解模型，能更好突出时序的周期特性，实现更准确的时序预测。

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

实验复现结果如图4所示，

Models		Autoformer		Reproduce		Input-192		Input-336	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	96	0.255	0.339	0.225	0.308	0.247	0.326	0.312	0.379
	192	0.281	0.340	0.293	0.345	0.309	0.363	0.393	0.435
	336	0.339	0.372	0.327	0.365	0.365	0.400	0.380	0.420
	720	0.422	0.419	0.436	0.430	0.468	0.453	-	-
Electricity	96	0.201	0.319	0.200	0.315	0.195	0.313	0.197	0.311
	192	0.222	0.334	0.220	0.330	0.204	0.318	0.220	0.336
	336	0.231	0.338	0.235	0.341	0.216	0.331	0.231	0.348
	720	0.254	0.361	0.253	0.357	0.247	0.352	-	-
Exchange	96	0.197	0.323	0.140	0.270	0.313	0.416	0.377	0.475
	192	0.300	0.369	0.309	0.404	0.500	0.542	0.526	0.562
	336	0.509	0.524	0.472	0.515	0.608	0.598	0.653	0.629
	720	1.447	0.941	1.195	0.857	1.365	0.908	-	-
Traffic	96	0.613	0.388	0.624	0.392	0.612	0.378	0.622	0.390
	192	0.616	0.382	0.633	0.393	0.624	0.386	0.667	0.422
	336	0.622	0.337	0.634	0.392	0.634	0.392	0.621	0.383
	720	0.660	0.408	0.715	0.443	-	-	-	-
Weather	96	0.266	0.336	0.258	0.392	0.272	0.351	0.318	0.406
	192	0.307	0.367	0.316	0.374	0.327	0.391	0.371	0.421
	336	0.359	0.395	0.365	0.401	0.398	0.426	0.442	0.461
	720	0.419	0.428	0.408	0.419	0.471	0.482	-	-
ILI	24	3.483	1.287	3.603	1.311	-	-	-	-
	36	3.103	1.148	3.338	1.231	-	-	-	-
	48	2.669	1.085	2.638	1.063	-	-	-	-
	60	2.770	1.125	2.806	1.128	-	-	-	-

图 4: 实验结果

其中“-”是因为内存不足无法计算出结果，越低的 MSE 和 MAE 表示预测效果越好。

复现实验得到的结果大致上与论文给出的结果相符，输入序列 ILI 为 36，其余为 96，第二列表示

的是输出序列的长度。我还将输入序列的长度加长，分别得到输入序列为 192 和 336 的结果，我们的直觉认为，输入序列越长，序列中的时间信息就越多，模型理应做出更好的预测，实际上却刚好相反，加大输入序列长度反而使误差变得更大。分析原因可能是因为序列加长导致的序列中的噪声也变得更多，对模型预测的影响超过加上序列提供的额外信息量，导致预测效果变差。

6 总结与展望

针对长时序列预测中的复杂时间模式难以处理与运算效率高的问题，基于深度分解架构和自相关机制的 Autoformer 模型通过渐进式分解和序列级连接，大幅提高了长时间预测效率。同时，Autoformer 在能源、交通、经济、气象、疾病五大主流领域均表现出了优秀的长时预测结果，模型具有良好的效果鲁棒性，具有很强的应用落地价值。

该论文发表在 2021 年的 NeurIPS 会议上，新的一年时间里有很多学者在长期时序检测领域有新的科研突破，例如阿里达摩院新提出的基于频率增强分解的 FEDformer，效果全面超越 SOTA，以及本论文作者实验室清华大学软件学院机器学习组在 2022 时间序列预测方向也有新的工作，提出了一种非平稳时间序列的通用预测框架 NSTransformers，还有很多发布在顶会上的其他模型，这些都是我后续要去拜读学习的前沿科研成果，希望我能从中汲取灵感，有所启发，运用在自己的科研项目之中去。

参考文献

[1] 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.

[2] KITAEV N, KAISER Ł, LEVSKAYA A. Reformer: The efficient transformer[J]. arXiv preprint arXiv:2001.04451, 2020.

[3] 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.