Are Transformers Effective for Time Series Forecasting?

Ailing Zeng[1,2*], Muxi Chen[1*], Lei Zhang[2], Qiang Xu[1]

[1]The Chinese University of Hong Kong
[2]International Digital Economy Academy
{zengailing, leizhang}@idea.edu.cn,{mxchen21, qxu}@cse.cuhk.edu.hk

Abstract

Recently, there has been a surge of Transformer-based solutions for the long-term time series forecasting (LTSF) task. Despite the growing performance over the past few years, *we question the validity of this line of research in this work*. Specifically, Transformers is arguably the most successful solution to extract the semantic correlations among the elements in a long sequence. However, in time series modeling, we are to extract the temporal relations in *an ordered set of continuous points*. While employing positional encoding and using tokens to embed sub-series in Transformers facilitate preserving some ordering information, the nature of the *permutation-invariant* self-attention mechanism inevitably results in temporal information loss.

To validate our claim, we introduce a set of embarrassingly simple one-layer linear models named *LTSF-Linear* for comparison. Experimental results on nine real-life datasets show that *LTSF-Linear* surprisingly outperforms existing sophisticated Transformer-based LTSF models in all cases, and often by a large margin. Moreover, we conduct comprehensive empirical studies to explore the impacts of various design elements of LTSF models on their temporal relation extraction capability. We hope this surprising finding opens up new research directions for the LTSF task. We also advocate revisiting the validity of Transformer-based solutions for other time series analysis tasks (e.g., anomaly detection) in the future.

Introduction

Time series are ubiquitous in today's data-driven world. Given historical data, time series forecasting (TSF) is a long-standing task that has a wide range of applications, including but not limited to traffic flow estimation, energy management, and financial investment. Over the past several decades, TSF solutions have undergone a progression from traditional statistical methods (e.g., ARIMA (Ariyo, Adewumi, and Ayo 2014)) and machine learning techniques (e.g., GBRT (Friedman 2001)) to deep learning-based solutions, e.g., (Bai, Kolter, and Koltun 2018; Liu et al. 2022).

Transformer (Vaswani et al. 2017) is arguably the most successful sequence modeling architecture, demonstrating unparalleled performances in various applications, such as natural language processing (NLP) (Devlin et al. 2018), speech recognition (Dong, Xu, and Xu 2018), and computer vision (Liu et al. 2021b). Recently, there has also been a surge of Transformer-based solutions for time series analysis, as surveyed in (Wen et al. 2022). Most notable models, which focus on the less explored and challenging long-term time series forecasting (LTSF) problem, include LogTrans (Li et al. 2019) (NeurIPS 2019), Informer (Zhou et al. 2021) (AAAI 2021 Best paper), Autoformer (Xu et al. 2021) (NeurIPS 2021), Pyraformer (Liu et al. 2021a) (ICLR 2022 Oral), Triformer (Cirstea et al. 2022) (IJCAI 2022) and the recent FEDformer (Zhou et al. 2022) (ICML 2022).

The main working power of Transformers is from its multi-head self-attention mechanism, which has a remarkable capability of extracting semantic correlations among elements in a long sequence (e.g., words in texts or 2D patches in images). However, self-attention is *permutation-invariant* and "anti-order" to some extent. While using various types of positional encoding techniques can preserve some ordering information, it is still inevitable to have temporal information loss after applying self-attention on top of them. This is usually not a serious concern for semantic-rich applications such as NLP, e.g., the semantic meaning of a sentence is largely preserved even if we reorder some words in it. However, when analyzing time series data, there is usually a lack of semantics in the numerical data itself, and we are mainly interested in modeling the temporal changes among *a continuous set of points*. That is, the order itself plays the most crucial role. Consequently, we pose the following intriguing question: ***Are Transformers really effective for long-term time series forecasting?***

Moreover, while existing Transformer-based LTSF solutions have demonstrated considerable prediction accuracy improvements over traditional methods, in their experiments, all the compared (non-Transformer) baselines perform autoregressive or iterated multi-step (IMS) forecasting (Ariyo, Adewumi, and Ayo 2014; Salinas, Flunkert, and Gasthaus 2017; Bahdanau, Cho, and Bengio 2014; Taylor and Letham 2017), which are known to suffer from significant error accumulation effects for the LTSF problem. Therefore, in this work, we challenge Transformer-based LTSF solutions with direct multi-step (DMS) forecasting strategies to validate their real performance.

Not all time series are predictable, let alone long-term forecasting (e.g., for chaotic systems). We hypothesize that

*These authors contributed equally.

long-term forecasting is only feasible for those time series with a relatively clear trend and periodicity. As linear models can already extract such information, we introduce a set of embarrassingly simple models named ***LTSF-Linear*** as a new baseline for comparison. *LTSF-Linear* regresses historical time series with a one-layer linear model to forecast future time series directly. We conduct extensive experiments on nine widely-used benchmark datasets that cover various real-life applications: traffic, energy, economics, weather, and disease predictions. Surprisingly, our results show that *LTSF-Linear* outperforms existing complex Transformer-based models *in all cases, and often by a large margin (20% $\sim$ 50%)*. Moreover, we find that, in contrast to the claims in existing Transformers, most of them fail to extract temporal relations from long sequences, i.e., the forecasting errors are not reduced (sometimes even increased) with the increase of look-back window sizes. Finally, we conduct various ablation studies on existing Transformer-based TSF solutions to study the impact of various design elements in them.

To sum up, the contributions of this work include:

- To the best of our knowledge, this is the first work to challenge the effectiveness of the booming Transformers for the long-term time series forecasting task.
- To validate our claims, we introduce a set of embarrassingly simple one-layer linear models, named *LTSF-Linear*, and compare them with existing Transformer-based LTSF solutions on nine benchmarks. *LTSF-Linear* can be a new baseline for the LTSF problem.
- We conduct comprehensive empirical studies on various aspects of existing Transformer-based solutions, including the capability of modeling long inputs, the sensitivity to time series order, the impact of positional encoding and sub-series embedding, and efficiency comparisons. Our findings would benefit future research in this area.

With the above, we conclude that *the temporal modeling capabilities of Transformers for time series are exaggerated, at least for the existing LTSF benchmarks*. At the same time, while *LTSF-Linear* achieves a better prediction accuracy compared to existing works, it merely serves as a simple baseline for future research on the challenging long-term TSF problem. With our findings, we also advocate revisiting the validity of Transformer-based solutions for other time series analysis tasks (e.g., anomaly detection) in the future.

Preliminaries: TSF Problem Formulation

For time series containing C variates, given historical data $\mathcal{X} = \{X_1^t, ..., X_C^t\}_{t=1}^L$, wherein L is the look-back window size and X_i^t is the value of the i_{th} variate at the t_{th} time step. The time series forecasting task is to predict the values $\hat{\mathcal{X}} = \{\hat{X}_1^t, ..., \hat{X}_C^t\}_{t=L+1}^{L+T}$ at the T future time steps. When $T > 1$, iterated multi-step (IMS) forecasting (Taieb, Hyndman et al. 2012) learns a single-step forecaster and iteratively applies it to obtain multi-step predictions. Alternatively, direct multi-step (DMS) forecasting (Chevillon 2007) directly optimizes the multi-step forecasting objective at once.

Compared to DMS forecasting results, IMS predictions have smaller variance thanks to the autoregressive estimation procedure, but they inevitably suffer from error accumulation effects. Consequently, IMS forecasting is preferable when there is a highly-accurate single-step forecaster, and T is relatively small. In contrast, DMS forecasting generates more accurate predictions when it is hard to obtain an unbiased single-step forecasting model, or T is large.

Transformer-Based LTSF Solutions

Transformer-based models (Vaswani et al. 2017) have achieved unparalleled performances in many long-standing AI tasks in natural language processing and computer vision fields, thanks to the effectiveness of the multi-head self-attention mechanism. This has also triggered lots of research interest in Transformer-based time series modeling techniques (Wen et al. 2022). In particular, a large amount of research works are dedicated to the LTSF task (e.g., (Li et al. 2019; Liu et al. 2021a; Xu et al. 2021; Zhou et al. 2021, 2022)). Considering the ability to capture long-range dependencies with Transformer models, most of them focus on the less-explored long-term forecasting problem ($T \gg 1$)[1].

When applying the vanilla Transformer model to the LTSF problem, it has some limitations, including the quadratic time/memory complexity with the original self-attention scheme and error accumulation caused by the autoregressive decoder design. Informer (Zhou et al. 2021) addresses these issues and proposes a novel Transformer architecture with reduced complexity and a DMS forecasting strategy. Later, more Transformer variants introduce various time series features into their models for performance or efficiency improvements (Liu et al. 2021a; Xu et al. 2021; Zhou et al. 2022). We summarize the design elements of existing Transformer-based LTSF solutions as follows (see Figure 1).

Time series decomposition: For data preprocessing, normalization with zero-mean is common in TSF. Besides, Autoformer (Xu et al. 2021) first applies seasonal-trend decomposition behind each neural block, which is a standard method in time series analysis to make raw data more predictable (Cleveland 1990; Hamilton 2020). Specifically, they use a moving average kernel on the input sequence to extract the *trend-cyclical* component of the time series. The difference between the original sequence and the trend component is regarded as the *seasonal* component. On top of the decomposition scheme of Autoformer, FEDformer (Zhou et al. 2022) further proposes the mixture of experts' strategies to mix the trend components extracted by moving average kernels with various kernel sizes.

Input embedding strategies: The self-attention layer in the Transformer architecture cannot preserve the positional information of the time series. However, local positional information, i.e. the ordering of time series, is important. Besides, global temporal information, such as hierarchical timestamps (week, month, year) and agnostic timestamps (holidays and events), is also informative (Zhou et al. 2021). To enhance the temporal context of time-series inputs, a practical design in the SOTA Transformer-based methods is injecting several embeddings, like a fixed positional encoding, a

[1]Due to page limit, we leave the discussion of non-Transformer forecasting solutions in the Appendix.

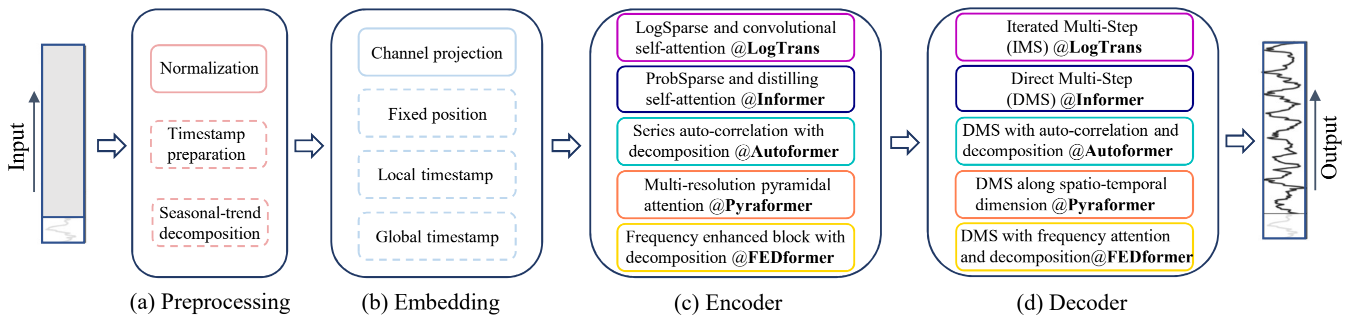
Input
Normalization
Timestamp preparation
Seasonal-trend decomposition
Channel projection
Fixed position
Local timestamp
Global timestamp
LogSparse and convolutional self-attention @LogTrans
ProbSparse and distilling self-attention @Informer
Series auto-correlation with decomposition @Autoformer
Multi-resolution pyramidal attention @Pyraformer
Frequency enhanced block with decomposition @FEDformer
Iterated Multi-Step (IMS) @LogTrans
Direct Multi-Step (DMS) @Informer
DMS with auto-correlation and decomposition @Autoformer
DMS along spatio-temporal dimension @Pyraformer
DMS with frequency attention and decomposition@FEDformer
Output
(a) Preprocessing
(b) Embedding
(c) Encoder
(d) Decoder

Figure 1: The pipeline of existing Transformer-based TSF solutions. In (a) and (b), the solid boxes are essential operations, and the dotted boxes are applied optionally. (c) and (d) are distinct for different methods (Li et al. 2019; Zhou et al. 2021; Xu et al. 2021; Liu et al. 2021a; Zhou et al. 2022).

channel projection embedding, and learnable temporal embeddings into the input sequence. Moreover, temporal embeddings with a temporal convolution layer (Li et al. 2019) or learnable timestamps (Xu et al. 2021) are introduced.

Self-attention schemes: Transformers rely on the self-attention mechanism to extract the semantic dependencies between paired elements. Motivated by reducing the $O\left(L^2\right)$ time and memory complexity of the vanilla Transformer, recent works propose two strategies for efficiency. On the one hand, LogTrans and Pyraformer explicitly introduce a sparsity bias into the self-attention scheme. Specifically, LogTrans uses a Logsparse mask to reduce the computational complexity to $O\left(LlogL\right)$ while Pyraformer adopts pyramidal attention that captures hierarchically multi-scale temporal dependencies with an $O\left(L\right)$ time and memory complexity. On the other hand, Informer and FEDformer use the low-rank property in the self-attention matrix. Informer proposes a ProbSparse self-attention mechanism and a self-attention distilling operation to decrease the complexity to $O\left(LlogL\right)$, and FEDformer designs a Fourier enhanced block and a wavelet enhanced block with random selection to obtain $O\left(L\right)$ complexity. Lastly, Autoformer designs a series-wise auto-correlation mechanism to replace the original self-attention layer.

Decoders: The vanilla Transformer decoder outputs sequences in an autoregressive manner, resulting in a slow inference speed and error accumulation effects, especially for long-term predictions. Informer designs a generative-style decoder for DMS forecasting. Other Transformer variants employ similar DMS strategies. For instance, Pyraformer uses a fully-connected layer concatenating Spatio-temporal axes as the decoder. Autoformer sums up two refined decomposed features from trend-cyclical components and the stacked auto-correlation mechanism for seasonal components to get the final prediction. FEDformer also uses a decomposition scheme with the proposed frequency attention block to decode the final results.

The premise of Transformer models is the semantic correlations between paired elements, while the self-attention mechanism itself is permutation-invariant, and its capability of modeling temporal relations largely depends on positional encodings associated with input tokens. Considering the raw numerical data in time series (e.g., stock prices or electricity values), there are hardly any point-wise semantic correlations between them. In time series modeling, we are mainly interested in the temporal relations among a continuous set of points, and the order of these elements instead of the paired relationship plays the most crucial role. While employing positional encoding and using tokens to embed sub-series facilitate preserving some ordering information, the nature of the permutation-invariant self-attention mechanism inevitably results in temporal information loss. Due to the above observations, we are interested in revisiting the effectiveness of Transformer-based LTSF solutions.

An Embarrassingly Simple Baseline for LTSF

In the experiments of existing Transformer-based LTSF solutions ($T \gg 1$), all the compared (non-Transformer) baselines are IMS forecasting techniques, which are known to suffer from significant error accumulation effects. We hypothesize that the performance improvements in these works are largely due to the DMS strategy used in them.

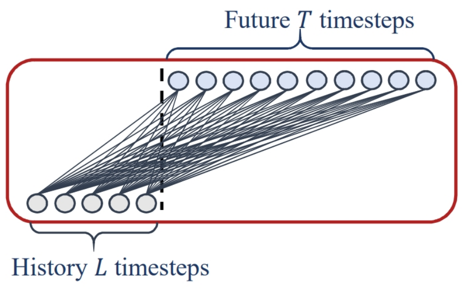

Figure 2: Illustration of the basic linear model.

To validate this hypothesis, we present the simplest DMS model via a temporal linear layer, named *LTSF-Linear*, as a baseline for comparison. The basic formulation of *LTSF-Linear* directly regresses historical time series for future prediction via a weighted sum operation (as illustrated in Figure 2). The mathematical expression is $\hat{X}_i = WX_i$, where

Datasets	ETTh1&ETTh2	ETTm1 &ETTm2	Traffic	Electricity	Exchange-Rate	Weather	ILI
Variates	7	7	862	321	8	21	7
Timesteps	17,420	69,680	17,544	26,304	7,588	52,696	966
Granularity	1hour	5min	1hour	1hour	1day	10min	1week

Table 1: The statistics of the nine popular datasets for the LTSF problem.

Methods		IMP.	Linear*		NLinear*		DLinear*		FEDformer		Autoformer		Informer		Pyraformer*		*Repeat**	
Metric		MSE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	27%	**0.140**	**0.237**	0.141	**0.237**	**0.140**	**0.237**	0.193	0.308	0.201	0.317	0.274	0.368	0.386	0.449	1.588	0.946
	192	24%	**0.153**	0.250	0.154	**0.248**	**0.153**	0.249	0.201	0.315	0.222	0.334	0.296	0.386	0.386	0.443	1.595	0.950
	336	21%	**0.169**	0.268	0.171	**0.265**	**0.169**	0.267	0.214	0.329	0.231	0.338	0.300	0.394	0.378	0.443	1.617	0.961
	720	17%	**0.203**	0.301	0.210	**0.297**	**0.203**	0.301	0.246	0.355	0.254	0.361	0.373	0.439	0.376	0.445	1.647	0.975
Exchange	96	45%	0.082	0.207	0.089	0.208	**0.081**	0.203	0.148	0.278	0.197	0.323	0.847	0.752	0.376	1.105	**0.081**	**0.196**
	192	42%	0.167	0.304	0.180	0.300	**0.157**	0.293	0.271	0.380	0.300	0.369	1.204	0.895	1.748	1.151	0.167	**0.289**
	336	34%	0.328	0.432	0.331	0.415	**0.305**	0.414	0.460	0.500	0.509	0.524	1.672	1.036	1.874	1.172	**0.305**	**0.396**
	720	46%	0.964	0.750	1.033	0.780	**0.643**	**0.601**	1.195	0.841	1.447	0.941	2.478	1.310	1.943	1.206	0.823	0.681
Traffic	96	30%	**0.410**	0.282	**0.410**	**0.279**	**0.410**	0.282	0.587	0.366	0.613	0.388	0.719	0.391	2.085	0.468	2.723	1.079
	192	30%	**0.423**	0.287	**0.423**	**0.284**	**0.423**	0.287	0.604	0.373	0.616	0.382	0.696	0.379	0.867	0.467	2.756	1.087
	336	30%	0.436	0.295	**0.435**	**0.290**	0.436	0.296	0.621	0.383	0.622	0.337	0.777	0.420	0.869	0.469	2.791	1.095
	720	26%	0.466	0.315	**0.464**	**0.307**	0.466	0.315	0.626	0.382	0.660	0.408	0.864	0.472	0.881	0.473	2.811	1.097
Weather	96	19%	**0.176**	0.236	0.182	**0.232**	**0.176**	0.237	0.217	0.296	0.266	0.336	0.300	0.384	0.896	0.556	0.259	0.254
	192	21%	**0.218**	0.276	0.225	**0.269**	0.220	0.282	0.276	0.336	0.307	0.367	0.598	0.544	0.622	0.624	0.309	0.292
	336	23%	**0.262**	0.312	0.271	**0.301**	0.265	0.319	0.339	0.380	0.359	0.395	0.578	0.523	0.739	0.753	0.377	0.338
	720	20%	0.326	0.365	0.338	**0.348**	**0.323**	0.362	0.403	0.428	0.419	0.428	1.059	0.741	1.004	0.934	0.465	0.394
ILI	24	48%	1.947	0.985	**1.683**	**0.858**	2.215	1.081	3.228	1.260	3.483	1.287	5.764	1.677	1.420	2.012	6.587	1.701
	36	36%	2.182	1.036	**1.703**	**0.859**	1.963	0.963	2.679	1.080	3.103	1.148	4.755	1.467	7.394	2.031	7.130	1.884
	48	34%	2.256	1.060	**1.719**	**0.884**	2.130	1.024	2.622	1.078	2.669	1.085	4.763	1.469	7.551	2.057	6.575	1.798
	60	34%	2.390	1.104	**1.819**	**0.917**	2.368	1.096	2.857	1.157	2.770	1.125	5.264	1.564	7.662	2.100	5.893	1.677
ETTh1	96	1%	0.375	0.397	**0.374**	**0.394**	0.375	0.399	0.376	0.419	0.449	0.459	0.865	0.713	0.664	0.612	1.295	0.713
	192	4%	0.418	0.429	0.408	**0.415**	**0.405**	0.416	0.420	0.448	0.500	0.482	1.008	0.792	0.790	0.681	1.325	0.733
	336	7%	0.479	0.476	**0.429**	**0.427**	0.439	0.443	0.459	0.465	0.521	0.496	1.107	0.809	0.891	0.738	1.323	0.744
	720	13%	0.624	0.592	**0.440**	**0.453**	0.472	0.490	0.506	0.507	0.514	0.512	1.181	0.865	0.963	0.782	1.339	0.756
ETTh2	96	20%	0.288	0.352	**0.277**	**0.338**	0.289	0.353	0.346	0.388	0.358	0.397	3.755	1.525	0.645	0.597	0.432	0.422
	192	20%	0.377	0.413	**0.344**	**0.381**	0.383	0.418	0.429	0.439	0.456	0.452	5.602	1.931	0.788	0.683	0.534	0.473
	336	26%	0.452	0.461	**0.357**	**0.400**	0.448	0.465	0.496	0.487	0.482	0.486	4.721	1.835	0.907	0.747	0.591	0.508
	720	14%	0.698	0.595	**0.394**	**0.436**	0.605	0.551	0.463	0.474	0.515	0.511	3.647	1.625	0.963	0.783	0.588	0.517
ETTm1	96	21%	0.308	0.352	0.306	0.348	**0.299**	**0.343**	0.379	0.419	0.505	0.475	0.672	0.571	0.543	0.510	1.214	0.665
	192	21%	0.340	0.369	0.349	0.375	**0.335**	**0.365**	0.426	0.441	0.553	0.496	0.795	0.669	0.557	0.537	1.261	0.690
	336	17%	0.376	0.393	0.375	0.388	**0.369**	**0.386**	0.445	0.459	0.621	0.537	1.212	0.871	0.754	0.655	1.283	0.707
	720	22%	0.440	0.435	0.433	0.422	**0.425**	**0.421**	0.543	0.490	0.671	0.561	1.166	0.823	0.908	0.724	1.319	0.729
ETTm2	96	18%	0.168	0.262	**0.167**	**0.255**	**0.167**	0.260	0.203	0.287	0.255	0.339	0.365	0.453	0.435	0.507	0.266	0.328
	192	18%	0.232	0.308	**0.221**	**0.293**	0.224	0.303	0.269	0.328	0.281	0.340	0.533	0.563	0.730	0.673	0.340	0.371
	336	16%	0.320	0.373	**0.274**	**0.327**	0.281	0.342	0.325	0.366	0.339	0.372	1.363	0.887	1.201	0.845	0.412	0.410
	720	13%	0.413	0.435	**0.368**	**0.384**	0.397	0.421	0.421	0.415	0.433	0.432	3.379	1.338	3.625	1.451	0.521	0.465

- Methods* are implemented by us; Other results are from FEDformer (Zhou et al. 2022).

Table 2: Multivariate long-term forecasting errors in terms of MSE and MAE, the lower the better. Among them, ILI dataset is with forecasting horizon $T \in \{24, 36, 48, 60\}$. For the others, $T \in \{96, 192, 336, 720\}$. The best results are highlighted in bold and the best results of Transformers are highlighted with an underline. IMP. is the best result of linear models compared to the results of Transformer-based solutions.

$W \in \mathbb{R}^{T \times L}$ is a linear layer along the temporal axis. $\hat{X}_i$ and X_i are the prediction and input for each i_{th} variate. Note that *LTSF-Linear* shares weights across different variates and does not model any spatial correlations.

LTSF-Linear is a set of linear models. *Vanilla Linear* is a one-layer linear model. To handle time series across different domains (e.g., finance, traffic, and energy domains), we further introduce two variants with two preprocessing methods, named *DLinear* and *NLinear*.

- Specifically, *DLinear* is a combination of a *Decomposition* scheme used in Autoformer and FEDformer with linear layers. It first decomposes a raw data input into a trend component by a moving average kernel and a remainder (seasonal) component. Then, two one-layer linear layers are applied to each component, and we sum up the two features to get the final prediction. By explicitly handling trend, *DLinear* enhances the performance of a vanilla linear when there is a clear trend in the data.

- Meanwhile, to boost the performance of *LTSF-Linear* when there is a distribution shift in the dataset, *NLinear* first subtracts the input by the last value of the sequence. Then, the input goes through a linear layer, and the subtracted part is added back before making the final prediction. The subtraction and addition in *NLinear* are a simple normalization for the input sequence.

Experiments

Experimental Settings

Dataset. We conduct extensive experiments on nine widely-used real-world datasets, including ETT (Electricity Transformer Temperature) (Zhou et al. 2021) (ETTh1, ETTh2, ETTm1, ETTm2), Traffic, Electricity, Weather, ILI, Exchange-Rate (Lai et al. 2017). All of them are multivariate time series. We leave *data descriptions* in the Appendix.

Evaluation metric. Following previous works (Zhou et al. 2021; Xu et al. 2021; Zhou et al. 2022), we use Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Compared methods. We include four recent Transformer-based methods: FEDformer (Fourier) (Zhou et al. 2022), Autoformer (Xu et al. 2021), Informer (Zhou et al. 2021), Pyraformer (Liu et al. 2021a). Besides, we include a naive DMS method: Closest Repeat (*Repeat*), which repeats the last value in the look-back window.

Comparison with Transformers

Quantitative results. In Table 2, we extensively evaluate all mentioned Transformers on nine benchmarks, following the experimental setting of previous work (Xu et al. 2021; Zhou et al. 2022, 2021). Surprisingly, the performance of *LTSF-Linear* surpasses the SOTA FEDformer in most cases by 20% $\sim$ 50% improvements on the *multivariate forecasting*, where *LTSF-Linear* even does not model correlations among variates. For different time series benchmarks, *NLinear* and *DLinear* show the superiority to handle the distribution shift and trend-seasonality features. We also provide results for *univariate forecasting* of ETT datasets in the Appendix, where *LTSF-Linear* still consistently outperforms Transformer-based LTSF solutions by a large margin. In general, these results reveal that existing complex Transformer-based LTSF solutions are not seemingly effective on the existing nine benchmarks while *LTSF-Linear* can be a powerful baseline. Another interesting observation is that even though the naive *Repeat* method shows worse results when predicting long-term seasonal data (e.g., Electricity), it surprisingly outperforms all Transformers on Exchange-Rate (around 45%). This is mainly caused by the wrong prediction of trends in Transformer-based solutions, which may overfit toward sudden change noises in the training data, resulting in significant accuracy degradation.

Qualitative results. As shown in Figure 3, we plot the prediction results on three selected time series datasets with Transformer-based solutions and *LTSF-Linear*: Electricity (Sequence 1951, Variate 36) , where it has different temporal patterns. When the input length is 96 steps, and the output horizon is 336 steps, Transformers fail to capture the scale and bias of the future data . Moreover, they can hardly predict a proper trend on aperiodic data.

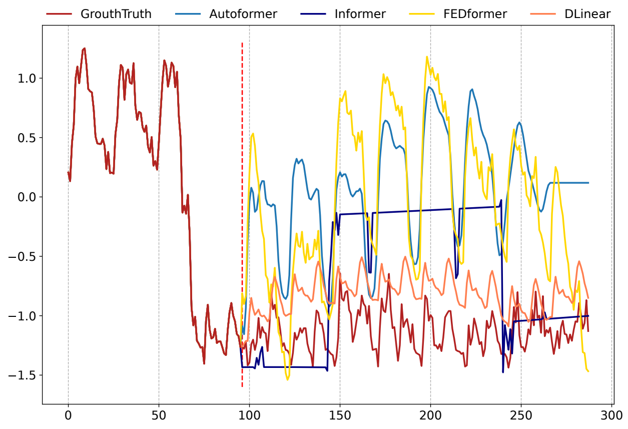

Figure 3: Illustration of the long-term forecasting outputs (Y-axis) of five models with an input length L=96 and output length T=192 (X-axis) on Electricity.

More Analyses on Transformer-Based Solutions

Can existing LTSF-Transformers extract temporal relations well from longer input sequences? The size of the look-back window greatly impacts forecasting accuracy as it determines how much we can learn from historical data. Generally speaking, a powerful TSF model with a strong temporal relation extraction capability should be able to achieve better results with larger look-back window sizes.

To study the impact of input look-back window sizes, we conduct experiments with $L \in \{24, ..., 720\}$ for long-term forecasting (T=720). Similar to the observations from previous studies (Zhou et al. 2021; Wen et al. 2022), existing Transformers' performance deteriorates or stays stable when the look-back window size increases. In contrast, the performances of all *LTSF-Linear* are significantly boosted with the increase of look-back window size. Thus, existing solutions tend to overfit temporal noises instead of extracting temporal information if given a longer sequence, and the input size 96 is exactly suitable for most Transformers.

What can be learned for long-term forecasting? While the temporal dynamics in the look-back window significantly impact the forecasting accuracy of short-term time series forecasting, we hypothesize that long-term forecasting depends on whether *models can capture the trend and periodicity well only.* That is, the farther the forecasting horizon, the less impact the look-back window itself has.

Methods	FEDformer		Autoformer	
Input	*Close*	*Far*	*Close*	*Far*
Electricity	0.251	0.265	0.255	0.287
Traffic	0.631	0.645	0.677	0.675

Table 3: The MSE comparisons of different input sequences.

To validate the above hypothesis, in Table 3, we com-

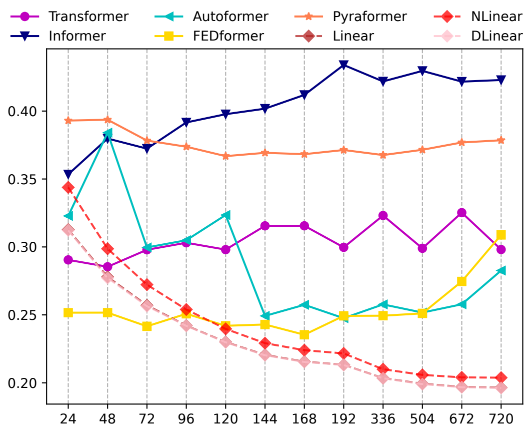

Figure 4: The MSE results (Y-axis) of models with different look-back window sizes (X-axis) of long-term forecasting (T=720) on Electricity.

pare the forecasting accuracy for the same future 720 time steps with data from two different look-back windows: (i). the original input L=96 setting (called *Close*) and (ii). the far input L=96 setting (called *Far*) that is before the original 96 time steps. The performance of the SOTA Transformers drops slightly, indicating these models only capture similar temporal information from the adjacent time series sequence. Since capturing the intrinsic characteristics of the dataset generally does not require a large number of parameters, i,e. one parameter can represent the periodicity. Using too many parameters will even cause overfitting.

Are the self-attention scheme effective for LTSF? We verify whether these complex designs in the existing Transformer (e.g., Informer) are essential. In Table 4, we gradually transform Informer to Linear. First, we replace each self-attention layer with a linear layer, called *Att.-Linear*, since a self-attention layer can be regarded as a fully-connected layer where weights are dynamically changed. Furthermore, we discard other auxiliary designs (e.g., FFN) in Informer to leave embedding layers and linear layers, named *Embed + Linear*. Finally, we simplify the model to one linear layer. As can be observed, the performance of Informer grows with the gradual simplification, thereby challenging the necessity of these modules.

Methods		Informer	*Att.-Linear*	*Embed + Linear*	Linear
Exchange	96	0.847	1.003	0.173	0.084
	192	1.204	0.979	0.443	0.155
	336	1.672	1.498	1.288	0.301
	720	2.478	2.102	2.026	0.763
ETTh1	96	0.865	0.613	0.454	0.400
	192	1.008	0.759	0.686	0.438
	336	1.107	0.921	0.821	0.479
	720	1.181	0.902	1.051	0.515

Table 4: The MSE comparisons of gradually transforming Informer to a Linear from the left to right columns.

Can existing LTSF-Transformers preserve temporal order well? Self-attention is inherently permutation-invariant, i.e., regardless of the order. However, in time-series forecasting, the sequence order often plays a crucial role. We argue that even with positional and temporal embeddings, existing Transformer-based methods still suffer from temporal information loss. In Table 5, we shuffle the raw input before the embedding strategies. Two shuffling strategies are presented: *Shuf.* randomly shuffles the whole input sequences and *Half-Ex.* exchanges the first half of the input sequence with the second half. Interestingly, compared with the original setting (*Ori.*) on the Exchange Rate, the performance of all Transformer-based methods does not fluctuate even when the input sequence is randomly shuffled. By contrary, the performance of *LTSF-Linear* is damaged significantly. These indicate that LTSF-Transformers with different positional and temporal embeddings preserve quite limited temporal relations and are prone to overfit on noisy financial data, while the simple *LTSF-Linear* can model the order naturally and avoid overfitting with fewer parameters.

For the ETTh1 dataset, FEDformer and Autoformer introduce time series inductive bias into their models, making them can extract certain temporal information when the dataset has more clear temporal patterns (e.g., periodicity) than the Exchange Rate. Therefore, the average drops of the two Transformers are 73.28% and 56.91% under the *Shuf.* setting, where it loses the whole order information. Moreover, Informer still suffers less from both *Shuf.* and *Half-Ex.* settings due to its no such temporal inductive bias. Overall, the average drops of *LTSF-Linear* are larger than Transformer-based methods for all cases, indicating the existing Transformers do not preserve temporal order well.

How effective are different embedding strategies? In Table 6, the forecasting errors of Informer largely increase without positional embeddings (wo/Pos.). Without timestamp embeddings (wo/Temp.) will gradually damage the performance of Informer as the forecasting lengths increase. Since Informer uses a single time step for each token, it is necessary to introduce temporal information in tokens.

Rather than using a single time step in each token, FEDformer and Autoformer input a sequence of timestamps to embed the temporal information. Hence, they can achieve comparable or even better performance without fixed positional embeddings. However, without timestamp embeddings, the performance of Autoformer declines rapidly because of the loss of global temporal information. Instead, thanks to the frequency-enhanced module proposed in FEDformer to introduce temporal inductive bias, it suffers less from removing any position/timestamp embeddings.

Is training data size a limiting factor for existing LTSF-Transformers? Some may argue that the poor performance of Transformer-based solutions is due to the small sizes of the benchmark datasets. Unlike computer vision or natural language processing tasks, TSF is performed on collected time series, and it is difficult to scale up the training data size. In fact, the size of the training data would indeed have a significant impact on the model performance. Accordingly, we conduct experiments on Traffic, comparing the performance of the model trained on a full dataset (17,544*0.7 hours), named *Ori.*, with that training on a shortened dataset

Methods		Linear			FEDformer			Autoformer			Informer		
Predict Length		*Ori.*	*Shuf.*	*Half-Ex.*	*Ori.*	*Shuf.*	*Half-Ex.*	*Ori.*	*Shuf.*	*Half-Ex.*	*Ori.*	*Shuf.*	*Half-Ex.*
Exchange	96	0.080	0.133	0.169	0.161	0.160	0.162	0.152	0.158	0.160	0.952	1.004	0.959
	192	0.162	0.208	0.243	0.274	0.275	0.275	0.278	0.271	0.277	1.012	1.023	1.014
	336	0.286	0.320	0.345	0.439	0.439	0.439	0.435	0.430	0.435	1.177	1.181	1.177
	720	0.806	0.819	0.836	1.122	1.122	1.122	1.113	1.113	1.113	1.198	1.210	1.196
	Average Drop	N/A	27.26%	46.81%	N/A	-0.09%	0.20%	N/A	0.09%	1.12%	N/A	-0.12%	-0.18%
ETTh1	96	0.395	0.824	0.431	0.376	0.753	0.405	0.455	0.838	0.458	0.974	0.971	0.971
	192	0.447	0.824	0.471	0.419	0.730	0.436	0.486	0.774	0.491	1.233	1.232	1.231
	336	0.490	0.825	0.505	0.447	0.736	0.453	0.496	0.752	0.497	1.693	1.693	1.691
	720	0.520	0.846	0.528	0.468	0.720	0.470	0.525	0.696	0.524	2.720	2.716	2.715
	Average Drop	N/A	81.06%	4.78%	N/A	73.28%	3.44%	N/A	56.91%	0.46%	N/A	1.98%	0.18%

Table 5: The MSE comparisons of models when shuffling the raw input sequence. *Shuf.* randomly shuffles the input sequence. *Half-EX.* randomly exchanges the first half of the input sequences with the second half. We run five times.

Methods	Embedding	Traffic			
		96	192	336	720
FEDformer	All	0.597	0.606	0.627	0.649
	wo/Pos.	**0.587**	**0.604**	**0.621**	**0.626**
	wo/Temp.	0.613	0.623	0.650	0.677
	wo/Pos.-Temp.	0.613	0.622	0.648	0.663
Autoformer	All	0.629	0.647	0.676	**0.638**
	wo/Pos.	**0.613**	**0.616**	**0.622**	0.660
	wo/Temp.	0.681	0.665	0.908	0.769
	wo/Pos.-Temp.	0.672	0.811	1.133	1.300
Informer	All	**0.719**	**0.696**	**0.777**	**0.864**
	wo/Pos.	1.035	1.186	1.307	1.472
	wo/Temp.	0.754	0.780	0.903	1.259
	wo/Pos.-Temp.	1.038	1.351	1.491	1.512

Table 6: The MSE comparisons of different embedding strategies on Transformer-based methods with look-back window size 96 and forecasting lengths $\{96, 192, 336, 720\}$.

(8,760 hours, i.e., 1 year), called *Short*. Unexpectedly, Table 7 presents that the prediction errors with reduced training data are usually lower. This might be because the whole-year data maintain clearer temporal features than a longer but incomplete data size. While we cannot conclude that we should use fewer data for training, it demonstrates that the training data scale is not the limiting reason.

Methods	FEDformer		Autoformer	
Dataset	*Ori.*	*Short*	*Ori.*	*Short*
96	0.587	**0.568**	0.613	**0.594**
192	0.604	**0.584**	**0.616**	0.621
336	0.621	**0.601**	0.622	**0.621**
720	0.626	**0.608**	0.660	**0.650**

Table 7: The MSE comparisons of two training data sizes.

Is efficiency really a top-level priority? Existing LTSF-Transformers claim that the $O\left(L^2\right)$ complexity of the vanilla Transformer is unaffordable for the LTSF problem. Although they prove to be able to improve the theoretical time and memory complexity from $O\left(L^2\right)$ to $O\left(L\right)$, it is unclear whether *1) the actual inference time and memory cost on devices are improved, and 2) the memory issue is unacceptable and urgent for today's GPU (e.g., an NVIDIA Titan XP here).* In Table 8, we compare the average practical efficiencies with 5 runs. Interestingly, compared with the vanilla Transformer (with the same DMS decoder), most Transformer variants incur similar or worse inference time and parameters in practice. These follow-ups introduce more additional design elements to make practical costs high.

Method	MACs	Parameter	Time	Memory
DLinear	**0.04G**	**139.7K**	**0.4ms**	**687MiB**
Transformer$\times$	4.03G	13.61M	26.8ms	6091MiB
Informer	3.93G	14.39M	49.3ms	3869MiB
Autoformer	4.41G	14.91M	164.1ms	7607MiB
Pyraformer	0.80G	241.4M	3.4ms	7017MiB
FEDformer	4.41G	20.68M	40.5ms	4143MiB

$\times$ the same one-step decoder.

Table 8: Comparison of practical efficiency of LTSF-Transformers under L=96 and T=720 on the Electricity. MACs are the number of multiply-accumulate operations. The inference time averages 5 runs.

Conclusion and Future Work

Conclusion. This work questions the effectiveness of emerging favored Transformer-based solutions for the long-term time series forecasting problem. We use an embarrassingly simple linear model *LTSF-Linear* as a DMS forecasting baseline to verify our claims. Note that our contributions do not come from proposing a linear model but rather from throwing out an important question, showing surprising comparisons, and demonstrating why LTSF-Transformers are not as effective as claimed in these works through various perspectives. We sincerely hope our comprehensive studies can benefit future work in this area.

Future work. *LTSF-Linear* has a limited model capacity, and it merely serves a simple yet competitive baseline with strong interpretability for future research. Consequently, we believe there is great potential for new model designs, data processing, and benchmarks to tackle LTSF.

Acknowledgments

This work was supported in part by Alibaba Group Holding Ltd. under Grant No. TA2015393. We thank the anonymous reviewers for their constructive comments and suggestions.

References

Ariyo, A. A.; Adewumi, A. O.; and Ayo, C. K. 2014. Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 106–112. IEEE.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv: Computation and Language arXiv:1409.0473*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Chevillon, G. 2007. Direct multi-step estimation and forecasting. *Journal of Economic Surveys*, 21(4): 746–785.

Cirstea, R.-G.; Guo, C.; Yang, B.; Kieu, T.; Dong, X.; and Pan, S. 2022. Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting–Full Version. *arXiv preprint arXiv:2204.13767*.

Cleveland, R. B. 1990. STL : A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Office Statistics*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, L.; Xu, S.; and Xu, B. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5884–5888. IEEE.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Hamilton, J. D. 2020. *Time series analysis*. Princeton university press.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2017. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. *international acm sigir conference on research and development in information retrieval*.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32.

Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; and Xu, Q. 2022. SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction. *Thirty-sixth Conference on Neural Information Processing Systems*.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2021a. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021b. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

Salinas, D.; Flunkert, V.; and Gasthaus, J. 2017. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting*.

Taieb, S. B.; Hyndman, R. J.; et al. 2012. *Recursive and direct multi-step forecasting: the best of both worlds*, volume 19. Citeseer.

Taylor, S. J.; and Letham, B. 2017. Forecasting at Scale. *PeerJ Prepr.*

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2022. Transformers in Time Series: A Survey. *arXiv preprint arXiv:2202.07125*.

Xu, J.; Wang, J.; Long, M.; et al. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, 11106–11115. AAAI Press.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*.