Are Transformers Effective for Time Series Forecasting?

Ailing Zeng[1]*, Muxi Chen[1]*, Lei Zhang[2], Qiang Xu[1]
[1]The Chinese University of Hong Kong
[2]International Digital Economy Academy (IDEA)
{alzeng, mxchen21, qxu}@cse.cuhk.edu.hk
{leizhang}@idea.edu.cn

Abstract

Recently, there has been a surge of Transformer-based solutions for the time series forecasting (TSF) task, especially for the challenging long-term TSF problem. Transformer architecture relies on self-attention mechanisms to effectively extract the semantic correlations between *paired elements* in a long sequence, which is *permutation-invariant and "anti-ordering"* to some extent. However, in time series modeling, we are to extract the temporal relations among *an ordering set of continuous points*. Consequently, whether Transformer-based techniques are the right solutions for "long-term" time series forecasting is an interesting problem to investigate, despite the performance improvements shown in these studies.

In this work, we question the validity of Transformer-based TSF solutions. In their experiments, the compared (non-Transformer) baselines are mainly autoregressive forecasting solutions, which usually have a poor long-term prediction capability due to inevitable error accumulation effects. In contrast, we use an embarrassingly simple architecture named *DLinear* that conducts direct multi-step (DMS) forecasting for comparison. *DLinear* decomposes the time series into a trend and a remainder series and employs two one-layer linear networks to model these two series for the forecasting task. Surprisingly, it outperforms existing complex Transformer-based models in most cases by a large margin. Therefore, we conclude that the relatively higher long-term forecasting accuracy of Transformer-based TSF solutions shown in existing works has little to do with the temporal relation extraction capabilities of the Transformer architecture. Instead, it is mainly due to the non-autoregressive DMS forecasting strategy used in them. We hope this study also advocates revisiting the validity of Transformer-based solutions for other time series analysis tasks (e.g., anomaly detection) in the future. Code is available at `https://github.com/cure-lab/DLinear`.

1 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 [1]) and machine learning techniques (e.g., GBRT [10]) to deep learning-based solutions, e.g., recurrent neural networks (RNNs) [15] and temporal convolutional networks (TCNs) [3]. At the same time, we are dealing with increasingly complex and diverse time series data,

*Equal contribution

Preprint. Under review.

ranging from univariate time series to multivariate time series and today's big-time series in many applications, requiring advanced deep neural networks for temporal relation extraction.

Transformer [25] is arguably the most successful sequence modeling architecture, which demonstrates unparalleled performances in various artificial intelligence applications, such as natural language processing [6], speech recognition [7], and motion analysis [19]. Recently, there has also been a surge of Transformer-based solutions for time series analysis, as surveyed in [26]. Some notable models for the TSF task include: LogTrans [16] (NeurIPS 2019), Informer [28] (AAAI 2021 Best paper), Autoformer [27] (NeurIPS 2021), Pyraformer [18] (ICLR 2022 Oral), and the recent FEDformer [29] (ICML 2022).

Most of the above works focus on the less explored long-term time series forecasting (LTSF) problem, demonstrating considerable prediction accuracy improvements over traditional methods. However, in their experiments, all the compared (non-Transformer) baselines perform autoregressive forecasting [1, 21, 2, 23], which are known to suffer from significant error accumulation effects. More importantly, the main working power of the Transformer architecture is from its multi-head self-attention mechanism, which has a remarkable capability of extracting semantic correlations between *paired elements* in a long sequence (e.g., words in texts or 2D patches in images), and this procedure is *permutation-invariant*, i.e., regardless of the order. However, for time series analysis, we are mainly interested in modeling the temporal dynamics among *a continuous set of points*, wherein the order itself often plays the most crucial role. Based on the above analysis, we pose the following intriguing question: **Are Transformers really effective for long-term time series forecasting?**

To answer this question, we present an embarrassingly simple network named *DLinear* as a baseline for comparison. *DLinear* decomposes the time series into a trend and a remainder series and employs two one-layer linear networks to model these two series with direct multi-step (DMS) forecasting. We conduct extensive experiments on nine widely-used benchmarks, including several real-life applications: traffic, energy, economics, weather, and disease predictions.

Our results show that *DLinear* outperforms existing complex Transformer-based models in most cases by a large margin. In particular, for the Exchange-Rate dataset that does not have obvious periodicity, the prediction errors of the state-of-the-art method [29] are more than twice larger than those of *DLinear*. Moreover, we find that, in contrast to the claims in existing works, 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 also conduct various ablation studies on existing Transformer-based TSF solutions to study the impact of various design elements in them.

With the above, we conclude that *the temporal modeling capabilities of Transformers for time series are exaggerated, at least for the time series forecasting problem*. At the same time, while *DLinear* 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.

The remainder of this paper is organized as follows. Sec. 2 presents the preliminaries on time series forecasting. Then, we discuss existing Transformer-based solutions in Sec. 3. Next, Sec. 4 details the baseline *DLinear* architecture. Experimental results are then shown in Sec. 5. Finally, Sec. 6 concludes this paper.

2 Preliminaries

2.1 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 $T > 1$, we can learn a single-step forecaster and iteratively apply it to obtain multi-step predictions, known as iterated multi-step (IMS) forecasting [22]. Alternatively, we can directly optimize the multi-step forecasting objective at once, known as direct multi-step (DMS) forecasting [4].

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 relatively more accurate prediction results when it is hard to obtain an unbiased single-step forecasting model or T is large.

2.2 Non-Transformer-Based TSF Solutions

As a long-standing problem that has a wide range of applications, statistical approaches (e.g., autoregressive integrated moving average (ARIMA) [1], exponential smoothing [11], and structural models [13]) for time series forecasting have been used from the 1970s onward.

Generally speaking, the parametric models used in statistical methods require significant domain expertise to build. To relieve this burden, many machine learning techniques such as gradient boosting regression tree (GBRT) [10, 9] gain popularity, which learns the temporal dynamics of time series in a data-driven manner.

However, these methods still require manual feature engineering and model designs.

With the powerful representation learning capability of deep neural networks (DNNs) from abundant data, various deep learning-based TSF solutions are proposed in the literature, achieving better forecasting accuracy than traditional techniques in many cases. Besides Transformers, the other two popular DNN architectures are also applied for time series forecasting:

- Recurrent neural networks (RNNs) based methods (e.g., [20]) summarize the past information compactly in internal memory states and recursively update themselves for forecasting.
- Convolutional neural networks (CNNs) based methods (e.g., [3]), wherein convolutional filters are used to capture local temporal features.

RNN-based TSF methods belong to IMS forecasting techniques. Depending on whether the decoder is implemented in an autoregressive manner, there are either IMS or DMS forecasting techniques for CNN-based TSF methods [3, 17].

3 Transformer-Based LTSF Solutions

Transformer-based models [25] have achieved unparalleled performances in many long-standing AI tasks in natural language processing and computer vision fields, thanks to the effectiveness and efficiency of the multi-head self-attention mechanism. This has also triggered lots of research interests in Transformer-based time series modeling techniques, as surveyed in [26]. In particular, a large amount of research works are dedicated to the TSF task (e.g., [16, 18, 27–29]). Considering the ability to capture long-range dependencies with Transformer models, most of them focus on the less-explored long-term forecasting problem ($T \gg 1$).

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 [28] 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 [18, 27, 29]. We summarize the design elements of existing Transformer-based TSF solutions as follows (see Figure 1).

Time series decomposition: For data preprocessing, normalization with zero-mean is common in TSF. Besides, Autoformer [27] first applies seasonal-trend decomposition behind each neural block, which is a standard method in time series analysis to make raw data more predictable [5, 12]. 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 [29] further proposes the mixture of experts strategy 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

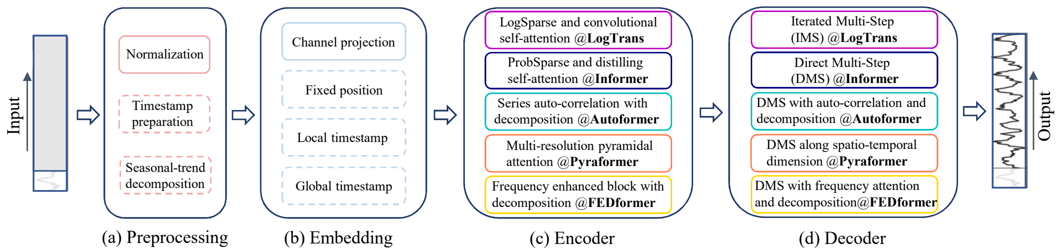


Figure 1: The pipeline of existing Transformer-based TSF solutions. In (a) and (b), the solid boxes are essential operations and the dotted boxes are applied optionally. (c) and (d) are distinct for different methods [16, 28, 27, 18, 29].

time series, is important. Besides, global temporal information, such as hierarchical timestamps (week, month, year) and agnostic timestamps (holidays, and events), is also informative [28]. 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 channel projection embedding, and learnable temporal embeddings into the input sequence. Moreover, temporal embeddings with a temporal convolution layer [16] or learnable timestamps [27] are introduced.

Self-attention schemes: Transformers rely on the self-attention mechanism to extract the semantic dependencies between paired elements. To reduce the $O\left(L^2\right)$ time and memory complexity of the vanilla Transformer, two strategies are used for efficiency improvements. On the one hand, LogTrans [16] and Pyraformer [18] explicitly introduce a sparsity bias into the self-attention scheme. To be specific, LogTrans [16] uses a Logsparse mask to reduce the computational complexity to $O\left(LlogL\right)$ while Pyraformer [18] 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 [28] and FEDformer [29] use the low-rank property in the self-attention matrix. Informer [28] proposes a ProbSparse self-attention mechanism and a self-attention distilling operation to decrease the complexity to $O\left(LlogL\right)$ and FEDformer [29] designs a Fourier enhanced block and a wavelet enhanced block with random selection to obtain $O\left(L\right)$ complexity. Lastly, Autoformer [27] designs a series-wise auto-correlation mechanism to replace the original self-attention layer.

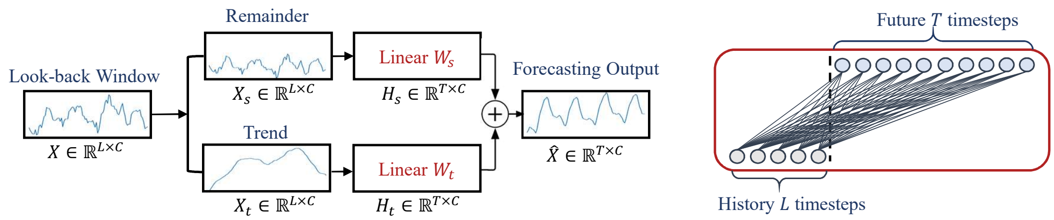
Decoders: The original Transformer decoder outputs sequences in an autoregressive manner, resulting in a slow inference speed and error accumulation effects, especially for long-term predictions. Informer [28] designs a generative-style decoder for DMS forecasting. Other Transformer variants employ similar DMS strategies. For instance, Pyraformer [18] uses a fully-connected layer concatenating spatio-temporal axes as the decoder. Autoformer [27] 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 [29] 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. 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.

4 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 shown in these works are largely due to the DMS strategy used in them. To validate this hypothesis, we present an embarrassingly simple linear model with time series decomposition, named *DLinear*, as a baseline for comparison. There are mainly two observations that inspire our design.

First, an one-layer linear network is arguably the simplest network to aggregate historical information for future prediction. Second, from the experiments in previous works [26, 27, 29], decomposition can largely enhance the performance of Transformer-based methods in time series forecasting, which is model-agnostic and may boost other models, such as a linear model. Accordingly, *DLinear* is a combination of a decomposition scheme and a linear network. It first decomposes a time-series data into a trend component $X_t \in \mathbb{R}^{L \times C}$ and a remainder data $X_s = X - X_t$. Then, two one-layer linear networks are applied to the two series.



(a) The whole structure of *DLinear* (b) One Linear Layer

Figure 2: Illustration of the Decomposition Linear Model.

The overall structure of *DLinear* is shown in Figure 2 (a). The whole process is: $\hat{X} = H_s + H_t$, where $H_s = W_s X_s \in \mathbb{R}^{T \times C}$ and $H_t = W_t X_t \in \mathbb{R}^{T \times C}$ are the decomposed trend and remainder features. $W_s \in \mathbb{R}^{T \times L}$ and $W_t \in \mathbb{R}^{T \times L}$ are two linear layers, as illustrated in Figure 2(b).

Note that, if the variates of the dataset have different characteristics, i.e., different seasonality and trend, *DLinear* with shared weights across different variates might not perform well. Therefore, we have two designs in *DLinear*. We name the *DLinear* network that every variate shares the same linear layer as *DLinear*-S and the *DLinear* network that has a linear layer for each variate individually as *DLinear*-I. We use *DLinear*-S by default.

Although *DLinear* is simple, it has some compelling characteristics:

- **An $O(1)$ maximum signal traversing path length**: The shorter the path, the better the dependencies are captured [18], making *DLinear* capable of capturing both short-range and long-range temporal relations.
- **High-efficiency:** As each branch has only one linear layer, it costs much lower memory and fewer parameters and has a faster inference speed than existing Transformers (see Table 8).
- **Interpretability:** After training, we can visualize weights from the seasonality and trend branches to have some insights on the predicted values [8].
- **Easy-to-use:** *DLinear* can be obtained easily without tuning model hyper-parameters.

5 Experiments

5.1 Experimental Settings

Dataset. We conduct extensive experiments on nine widely-used real-world datasets, including ETT (Electricity Transformer Temperature) [28] (ETTh1, ETTh2, ETTm1, ETTm2), Traffic, Electricity, Weather, ILI, Exchange-Rate [15]. All of them are multivariate time series. We leave *the detailed data descriptions* in the supplementary material.

Table 1: The statistics of the nine benchmark datasets.

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

Evaluation metric. Following previous works [28, 27, 29], we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the core metrics to compare performance.

Table 2: Long-term forecasting errors in terms of MSE and MAE, the lower the better. Among them, four datasets are with look-back window size $L = 96$ and forecasting horizon $T \in \{96, 192, 336, 720\}$. For the ILI dataset, $L = 36$ and $T \in \{24, 36, 48, 60\}$. Repeat-C repeats the last value of the look-back window. The best results are highlighted in **red bold** and the second best results are highlighted with a blue underline.

Methods	Metric	Electricity				Exchange-Rate				Traffic				Weather				ILI			
		96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720	24	36	48	60
DLinear-S*	MSE	0.194	0.193	0.206	0.242	**0.078**	0.159	0.274	**0.558**	0.650	**0.598**	**0.605**	0.645	0.196	0.237	0.283	0.345	**2.398**	**2.646**	2.614	2.804
	MAE	0.276	0.280	0.296	0.329	0.197	0.292	0.391	**0.574**	0.396	**0.370**	0.373	0.394	0.255	0.296	0.335	0.381	**1.040**	1.088	1.086	1.146
DLinear-I*	MSE	**0.184**	**0.184**	**0.197**	**0.234**	0.084	**0.157**	**0.236**	0.626	0.647	0.602	0.607	0.646	**0.164**	**0.209**	**0.263**	**0.338**	3.015	2.737	**2.577**	2.821
	MAE	**0.270**	**0.273**	**0.289**	**0.323**	0.216	0.298	**0.379**	0.634	0.403	0.375	0.377	0.398	**0.237**	**0.282**	**0.327**	**0.380**	1.192	**1.036**	1.043	**1.091**
FEDformer	MSE	0.193	0.201	0.214	0.246	0.148	0.271	0.460	1.195	**0.587**	0.604	0.621	**0.626**	0.217	0.276	0.339	0.403	3.228	2.679	2.622	2.857
	MAE	0.308	0.315	0.329	0.355	0.278	0.380	0.500	0.841	**0.366**	0.373	0.383	**0.382**	0.296	0.336	0.380	0.428	1.260	1.080	1.078	1.157
Autoformer	MSE	0.201	0.222	0.231	0.254	0.197	0.300	0.509	1.447	0.613	0.616	0.622	0.660	0.266	0.307	0.359	0.419	3.483	3.103	2.669	**2.770**
	MAE	0.317	0.334	0.338	0.361	0.323	0.369	0.524	0.941	0.388	0.382	**0.337**	0.408	0.336	0.367	0.395	0.428	1.287	1.148	1.085	1.125
Informer	MSE	0.274	0.296	0.300	0.373	0.847	1.204	1.672	2.478	0.719	0.696	0.777	0.864	0.300	0.598	0.578	1.059	5.764	4.755	4.763	5.264
	MAE	0.368	0.386	0.394	0.439	0.752	0.895	1.036	1.310	0.391	0.379	0.420	0.472	0.384	0.544	0.523	0.741	1.677	1.467	1.469	1.564
Pyraformer*	MSE	0.386	0.378	0.376	0.376	1.748	1.874	1.943	2.085	0.867	0.869	0.881	0.896	0.622	0.739	1.004	1.420	7.394	7.551	7.662	7.931
	MAE	0.449	0.443	0.443	0.445	1.105	1.151	1.172	1.206	0.468	0.467	0.469	0.473	0.556	0.624	0.753	0.934	2.012	2.031	2.057	2.100
LogTrans	MSE	0.258	0.266	0.280	0.283	0.968	1.040	1.659	1.941	0.684	0.685	0.734	0.717	0.458	0.658	0.797	0.869	4.480	4.799	4.800	5.278
	MAE	0.357	0.368	0.380	0.376	0.812	0.851	1.081	1.127	0.384	0.390	0.408	0.396	0.490	0.589	0.652	0.675	1.444	1.467	1.468	1.560
Reformer	MSE	0.312	0.348	0.350	0.340	1.065	1.188	1.357	1.510	0.732	0.733	0.742	0.755	0.689	0.752	0.639	1.130	4.400	4.783	4.832	4.882
	MAE	0.402	0.433	0.433	0.420	0.829	0.906	0.976	1.016	0.423	0.420	0.420	0.423	0.596	0.638	0.596	0.792	1.382	1.448	1.465	1.483
Repeat-C*	MSE	1.588	1.595	1.617	1.647	0.081	0.167	0.305	0.823	2.723	2.756	2.791	2.811	0.259	0.309	0.377	0.465	6.587	7.130	6.575	5.893
	MAE	0.946	0.950	0.961	0.975	**0.196**	**0.289**	0.396	0.681	1.079	1.087	1.095	1.097	0.254	0.292	0.338	0.394	1.701	1.884	1.798	1.677

- Methods∗ are implemented by us; Other results are from FEDformer [29].

Table 3: Long-term forecasting errors on four ETT benchmarks with look-back window size $L = 96$ and forecasting horizon $T \in \{96, 192, 336, 720\}$. The best results are highlighted in **red bold** and the second best results are highlighted with a blue underline.

Methods		*DLinear*-S*		FEDformer		Autoformer		Informer		Pyraformer*		LogTrans		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh1$	96	0.386	**0.400**	**0.376**	0.419	0.449	0.459	0.865	0.713	0.664	0.612	0.878	0.740	0.837	0.728
	192	0.437	**0.432**	**0.420**	0.448	0.500	0.482	1.008	0.792	0.790	0.681	1.037	0.824	0.923	0.766
	336	0.481	**0.459**	**0.459**	0.465	0.521	0.496	1.107	0.809	0.891	0.738	1.238	0.932	1.097	0.835
	720	0.519	0.516	**0.506**	**0.507**	0.514	0.512	1.181	0.865	0.963	0.782	1.135	0.852	1.257	0.889
$ETTh2$	96	**0.295**	**0.352**	0.346	0.388	0.358	0.397	3.755	1.525	0.645	0.597	2.116	1.197	2.626	1.317
	192	0.452	0.462	**0.429**	**0.439**	0.456	0.452	5.602	1.931	0.788	0.683	4.315	1.635	11.12	2.979
	336	0.504	0.490	0.496	0.487	**0.482**	**0.486**	4.721	1.835	0.907	0.747	1.124	1.604	9.323	2.769
	720	0.577	0.538	**0.463**	**0474**	0.515	0.511	3.647	1.625	0.963	0.783	3.188	1.540	3.874	1.697
$ETTm1$	96	**0.345**	**0.372**	0.379	0.419	0.505	0.475	0.672	0.571	0.543	0.510	0.600	0.546	0.538	0.528
	192	**0.380**	**0.389**	0.426	0.441	0.553	0.496	0.795	0.669	0.557	0.537	0.837	0.700	0.658	0.592
	336	**0.413**	**0.413**	0.445	0.459	0.621	0.537	1.212	0.871	0.754	0.655	1.124	0.832	0.898	0.721
	720	**0.474**	**0.453**	0.543	0.490	0.671	0.561	1.166	0.823	0.908	0.724	1.153	0.820	1.102	0.841
$ETTm2$	96	**0.183**	**0.273**	0.203	0.287	0.255	0.339	0.365	0.453	0.435	0.507	0.768	0.642	0.658	0.619
	192	**0.260**	**0.325**	0.269	0.328	0.281	0.340	0.533	0.563	0.730	0.673	0.989	0.757	1.078	0.827
	336	0.336	0.367	**0.325**	**0.366**	0.339	0.372	1.363	0.887	1.201	0.845	1.334	0.872	1.549	0.972
	720	**0.415**	0.423	0.421	**0.415**	0.433	0.432	3.379	1.338	3.625	1.451	3.048	1.328	2.631	1.242

- Methods* are implemented by us; Other results are from FEDformer [29].

Compared methods. We include six state-of-the-art Transformer-based methods: FEDformer [29], Autoformer [27], Informer [28], Pyraformer [18], LogTrans [16], and Reformer [14]. Besides, we include the naive DMS method: Closest Repeat (Repeat-C), which naively repeats the last value in the look-back window, as another simple baseline. Since there are two variants of FEDformer, we compare with the one with better accuracy (FEDformer-f via Fourier transform) shown in their paper. More implementation details are in the supplementary materials.

5.2 Comparison of Multivariate Forecasting with Transformers

To verify whether existing Transformer-based solutions are effective for TSF tasks, we present both quantitative and qualitative comparisons with *DLinear*.

Quantitative results. In Table 2 and 3, we extensively evaluate all mentioned Transformers on nine benchmarks, following the experimental setting of previous work [27, 29, 28]. As can be observed, the performance of *DLinear* is better than Transformer-based methods in most cases. Specifically, *DLinear* outperforms all the Transformer-based methods by over 25% on Weather and around 50% on

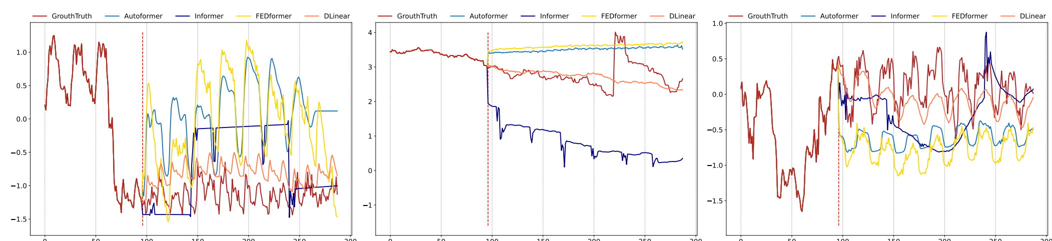
Exchange-Rate. These results indicate that existing Transformer-based TSF solutions are not effective for temporal relation extraction and *DLinear* is a powerful baseline for the long-term forecasting task.

The recent FEDformer achieves relatively high forecasting accuracy, especially for the ETT benchmark, as shown in Table 3. This might be due to the fact that FEDformer does not rely much on the self-attention mechanism in Transformers. Rather, it employs classical time series analysis techniques such as Fourier transformation, which plays an important role in temporal feature extraction.

It is worth noting that, while FEDformer outperforms *DLinear* in some cases, it is achieved under the setting of $T = 96$. We study the impact of different look-back window sizes (see Sec. 5.4), and our results show that *DLinear* continues to improve with increased T, eventually outperforming FEDformer by a large margin.

Another interesting observation is that even though the naive Repeat-C method shows worse results when predicting long-term seasonal data (e.g., Electricity and Traffic), it surprisingly outperforms all the Transformer-based methods on the Exchange-Rate (over 30%) and Weather (over 10%) dataset. This is mainly caused by the wrong prediction of trends in Transformer-based solutions, resulting in significant accuracy degradation (see Figure 3(b)).

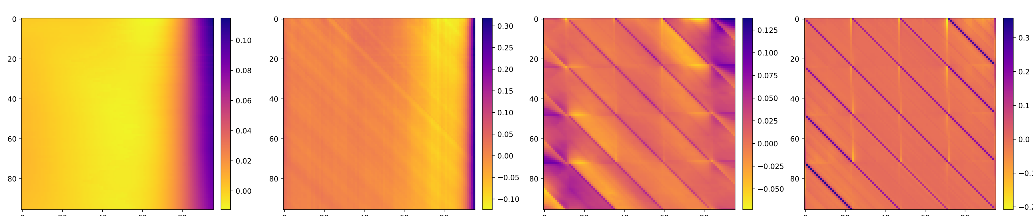
Qualitative results. As shown in Figure 3, we plot the prediction results on three selected time series datasets with Transformer-based solutions and *DLinear*: Electricity (Sequence 1951, Variate 36), Exchange-Rate (Sequence 676, Variate 3), and ETTh2 (Sequence 1241, Variate 2), where these data have different temporal patterns. When the input length is 96 steps and the output horizon is 336 steps, Transformers [28, 27, 29] fail to capture the scale and bias of the future data on Electricity and ETTh2. Moreover, they can hardly predict a proper trend on aperiodic data such as Exchange-Rate. These phenomena further indicate the inadequacy of existing Transformers for the TSF task.



(a) Electricity (b) Exchange-Rate (c) ETTh2

Figure 3: Illustration of the long-term forecasting output (Y-axis) of five models with an input length L=96 and output length T=192 (X-axis) on Electricity, Exchange-Rate, and ETTh2, respectively.

The interpretability of *DLinear*.



(a) Trend-Exchange (b) Remainder-Exchange (c) Trend-Electricty (d) Remainder-Electricty

Figure 4: Visualization of the weights(T*L) of *DLinear*. (a) and (c) are weights in the remainder layer. (b) and (d) are weights in the trend layer. Models are trained with a look-back window L=96 (X-axis) and a forecasting length T=96 (Y-axis) on the Exchange-Rate and Electricity datasets, respectively.

Because *DLinear* is a linear model, its weights can directly reveal how *DLinear* works. Figure 4(a) and (b) visualize the weights of the trend and the remainder layers on the Exchange-Rate dataset. Due to the lack of periodicity and seasonality in financial data, it is hard to observe clear patterns, but the trend layer reveals greater weights of information closer to the outputs, representing their larger contributions to the predicted values.

In contrast, for periodical data, Figure 4 (d) shows a clear seasonality. We can count the interval/periodicity of adjacent purple lines (weights with high value) as 24 (for one day). Specifically, the value of the first output time step strongly depends on the 1_{th}, 25_{th}, 49_{th}... time steps of the input. A similar phenomenon appears in the trend layer shown in Figure 4(c).

5.3 Comparison under a Large Look-Back Window Size

Multivariate forecasting. In Table 2, we follow the settings of existing Trasnformer-based TSF solutions and fix the look-back window size as **96** for a fair comparison. The results of FEDformer are on par with those of *DLinear* on some datasets, i.e., Traffic, Electricity, and ETTh1. In this experiment, we set the look-back window size to be 336. As shown in Table 4, the performance of *DLinear* can largely surpass the SOTA Transformer-based methods (FEDformer and Autoformer). Specifically, *DLinear* outperforms FEDformer by over 40% on Exchange rate, around 30% on Traffic, Electricity, and Weather, and around 25% on ETTm1 and Weather.

Table 4: Multivariate long sequence time-series forecasting results on nine benchmarks with a long look-back window size $L = 336$ of *DLinear* and forecasting horizon $T \in \{96, 192, 336, 720\}$. Since the SOTA methods FEDformer and Autoformer obtain better performance with an input length $L = 96$ as shown in Figure 7, we keep their best settings for comparison. FEDformer-f and FEDformer-w represent the FEDformer with fourier enhanced blocks and wavelet enhanced blocks, respectively. The best results are highlighted in **red bold** and the second best results are highlighted with a blue underline.

Methods		DLinear-S		DLinear-I		FEDformer-f		FEDformer-w		Autoformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.375**	0.399	0.377	**0.397**	0.376	0.419	0.395	0.424	0.449	0.459
	192	**0.405**	**0.416**	0.413	0.421	0.420	0.448	0.469	0.470	0.500	0.482
	336	**0.439**	0.443	0.440	**0.439**	0.459	0.465	0.530	0.499	0.521	0.496
	720	**0.472**	0.490	0.476	**0.481**	0.506	0.507	0.598	0.544	0.514	0.512
ETTh2	96	**0.289**	**0.353**	0.438	0.451	0.346	0.388	0.394	0.414	0.358	0.397
	192	**0.383**	**0.418**	0.615	0.517	0.429	0.439	0.439	0.445	0.456	0.452
	336	**0.448**	**0.465**	0.603	0.525	0.496	0.487	0.482	0.480	0.482	0.486
	720	0.605	0.551	1.082	0.723	**0.463**	**0.474**	0.500	0.509	0.515	0.511
ETTm1	96	0.299	0.343	**0.286**	**0.334**	0.379	0.419	0.378	0.418	0.505	0.475
	192	0.335	0.365	**0.327**	**0.358**	0.426	0.441	0.464	0.463	0.553	0.496
	336	0.369	0.386	**0.367**	**0.383**	0.445	0.459	0.508	0.487	0.621	0.537
	720	**0.425**	0.421	0.429	**0.418**	0.543	0.490	0.561	0.515	0.671	0.561
ETTm2	96	**0.167**	**0.260**	0.195	0.288	0.203	0.287	0.204	0.288	0.255	0.339
	192	**0.224**	**0.303**	0.332	0.367	0.269	0.328	0.316	0.363	0.281	0.340
	336	**0.281**	**0.342**	0.545	0.476	0.325	0.366	0.359	0.387	0.339	0.372
	720	**0.397**	0.421	0.697	0.546	0.421	**0.415**	0.433	0.432	0.422	0.419
Eletricity	96	0.140	0.237	**0.133**	**0.230**	0.193	0.308	0.183	0.297	0.201	0.317
	192	0.153	0.249	**0.148**	**0.245**	0.201	0.315	0.195	0.308	0.222	0.334
	336	0.169	0.267	**0.164**	**0.263**	0.214	0.329	0.212	0.313	0.231	0.338
	720	0.203	0.301	**0.201**	**0.297**	0.246	0.355	0.231	0.343	0.254	0.361
Exchange	96	**0.081**	**0.203**	0.091	0.220	0.148	0.278	0.139	0.276	0.197	0.323
	192	**0.157**	**0.293**	0.184	0.323	0.271	0.380	0.256	0.369	0.300	0.369
	336	**0.305**	**0.414**	0.328	0.436	0.460	0.500	0.426	0.464	0.509	0.524
	720	**0.643**	**0.601**	0.975	0.781	1.195	0.841	1.090	0.800	1.447	0.941
Traffic	96	**0.410**	**0.282**	0.440	0.308	0.587	0.366	0.562	0.349	0.613	0.388
	192	**0.423**	**0.287**	0.451	0.314	0.604	0.373	0.562	0.346	0.616	0.382
	336	**0.436**	**0.296**	0.464	0.321	0.621	0.383	0.570	0.323	0.622	0.337
	720	**0.466**	**0.315**	0.494	0.340	0.626	0.382	0.596	0.368	0.660	0.408
Weather	96	0.176	0.237	**0.146**	**0.213**	0.217	0.296	0.227	0.304	0.266	0.336
	192	0.220	0.282	**0.191**	**0.258**	0.276	0.336	0.295	0.363	0.307	0.367
	336	0.265	0.319	**0.244**	**0.301**	0.339	0.380	0.381	0.416	0.359	0.395
	720	0.323	0.362	**0.317**	**0.359**	0.403	0.428	0.424	0.434	0.419	0.428
ILI	96	2.108	1.074	**2.007**	**0.953**	3.228	1.260	2.203	0.963	3.483	1.287
	192	2.332	1.078	**2.175**	0.979	2.679	1.080	2.272	**0.976**	3.103	1.148
	336	**2.088**	1.011	2.202	0.999	2.622	1.078	2.209	**0.981**	2.669	1.085
	720	**2.299**	1.078	2.348	**1.066**	2.857	1.157	2.545	1.061	2.770	1.125

Univariate forecasting. We also present the univariate time series forecasting on the four ETT datasets. *DLinear* represents the same *DLinear*-S and *DLinear*-I, since they have the same structure for univariate forecasting. From Table 4, the mean errors of *DLinear* are also lower FEDformer and Autoformer at most cases.

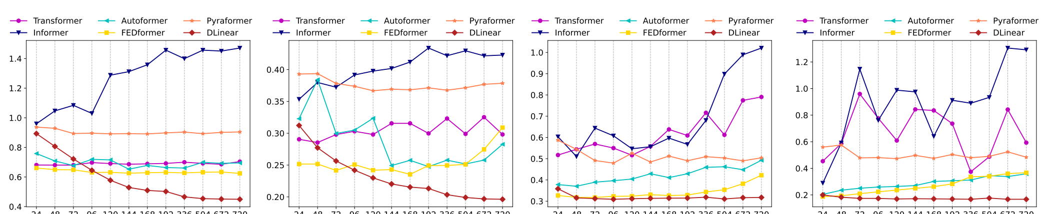
Table 5: Univariate long-term time-series forecasting results on ETT full benchmark with a long look-back window size $L = 336$ of *DLinear* and forecasting horizon $T \in \{96, 192, 336, 720\}$. Since the SOTA methods FEDformer and Autoformer obtain better performance with an input length $L = 96$ as shown in Figure 7, we keep their best settings for comparison. FEDformer-f and FEDformer-w represent the FEDformer with fourier enhanced blocks and wavelet enhanced blocks, respectively. The best results are highlighted in **red bold** and the second best results are highlighted with a blue underline.

Methods		*DLinear*		FEDformer-f		FEDformer-w		Autoformer		Informer		LogTrans		Reformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh1$	96	**0.056**	**0.180**	0.079	0.215	0.080	0.214	0.071	0.206	0.193	0.377	0.283	0.468	0.532	0.569
	192	**0.071**	**0.204**	0.104	0.245	0.105	0.256	0.114	0.262	0.217	0.395	0.234	0.409	0.568	0.575
	336	**0.098**	**0.244**	0.119	0.270	0.120	0.269	0.107	0.258	0.202	0.381	0.386	0.546	0.635	0.589
	720	**0.119**	**0.274**	0.142	0.299	0.127	0.280	0.126	0.283	0.183	0.355	0.475	0.628	0.762	0.666
$ETTh2$	96	0.131	0.279	**0.128**	**0.271**	0.156	0.306	0.153	0.306	0.213	0.373	0.217	0.379	1.411	0.838
	192	**0.176**	**0.329**	0.185	0.330	0.238	0.380	0.204	0.351	0.227	0.387	0.281	0.429	5.658	1.671
	336	**0.209**	**0.367**	0.231	0.378	0.271	0.412	0.246	0.389	0.242	0.401	0.293	0.437	4.777	1.582
	720	0.276	0.426	0.278	0.420	0.288	0.438	**0.268**	**0.409**	0.291	0.439	0.218	0.387	2.042	1.039
$ETTm1$	96	**0.028**	**0.123**	0.033	0.140	0.036	0.149	0.056	0.183	0.109	0.277	0.049	0.171	0.296	0.355
	192	**0.045**	**0.156**	0.058	0.186	0.069	0.206	0.081	0.216	0.151	0.310	0.157	0.317	0.429	0.474
	336	**0.061**	**0.182**	0.084	0.231	0.071	0.209	0.076	0.218	0.427	0.591	0.289	0.459	0.585	0.583
	720	**0.080**	**0.210**	0.102	0.250	0.105	0.248	0.110	0.267	0.438	0.586	0.430	0.579	0.782	0.730
$ETTm2$	96	**0.063**	**0.183**	0.067	0.198	**0.063**	0.189	0.065	0.189	0.088	0.225	0.075	0.208	0.076	0.214
	192	**0.092**	**0.227**	0.102	0.245	0.110	0.252	0.118	0.256	0.132	0.283	0.129	0.275	0.132	0.290
	336	**0.119**	**0.261**	0.130	0.279	0.147	0.301	0.154	0.305	0.180	0.336	0.154	0.302	0.160	0.312
	720	**0.175**	**0.320**	0.178	0.325	0.219	0.368	0.182	0.335	0.300	0.435	0.160	0.321	0.168	0.335

5.4 More Analyses on Transformer-Based Solutions

Impact of the look-back window size. The size of the look-back window has a high impact on 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 make a fair comparison, we follow the experimental settings in Autoformer [27] and FEDformer [29] and set a fixed look-back window size (36 for ILI and 96 for other datasets) in previous experiment. To study the impact of look-back window sizes, we conduct experiments with $L \in \{24, 48, 72, 96, 120, 144, 168, 192, 336, 504, 672, 720\}$ to forecast either short-term time series (24 steps) or long-term time series (720 steps).



(a) **720** steps-**Traffic** (b) **720** steps-**Electricity** (c) **24** steps-**ETTh1** (d) **24** steps-**ETTh2**

Figure 5: The MSE results (Y-axis) of models with different look-back window sizes (X-axis) of the Long Forecasting (**720** time steps) and the Short Forecasting (**24** time steps) on four datasets. Please refer to Fig. 7 for comparisons on other benchmarks.

Figure 5 demonstrates the MSE results on four datasets. Please also refer to Fig. 7 for comparisons on other benchmarks. Similar to the observations from previous studies [28, 26], the performance of existing Transformer-based models will deteriorate or keep stable when the look-back window sizes increase. In contrast, the performances of *DLinear* are significantly boosted with the increase of look-back window size. The input size 96 is suitable for most Transformers, and FEDformer can obtain a comparable performance with *DLinear* under this setting. Note that, we provide more quantitative results in the supplementary materials, and our conclusion holds in almost all cases.

Impact of different embedding strategies. In Table 6, we study the benefits brought by embedding position/timestamp information into Transformer-based methods. As can be observed, the forecasting errors of Informer are largely increased without the positional embedding or timestamp embeddings, since Informer uses a single time step as a token for self-attention, indicating the necessity of introducing temporal information in the model. Rather than using a single time step in each token, FEDformer [29] and Autoformer [27] input a sequence of timestamps to introduce more positional information. They can achieve comparable or even better performance without the fixed positional embedding (wo/Pos.). Without timestamp embedding, the performance of Autoformer declines rapidly due to the loss of global temporal information. However, thanks to the frequency enhanced module proposed in FEDformer, it suffers less from removing any of the specific position/timestamp embeddings.

Table 6: The impact of different embedding strategies on Transformer-based methods with look-back window size $L = 96$ and forecasting horizon $T \in \{96, 192, 336, 720\}$. The metric used is MSE.

Methods	Embedding	Electricity 96	192	336	720	Traffic 96	192	336	720
FEDformer	All	**0.189**	**0.198**	**0.210**	0.248	0.597	0.606	0.627	0.649
	wo/Pos.	0.193	0.201	0.214	**0.246**	**0.587**	**0.604**	**0.621**	**0.626**
	wo/Temp.	0.209	0.213	0.222	0.265	0.613	0.623	0.650	0.677
	wo/Pos.-Temp.	0.203	0.211	0.222	0.250	0.613	0.622	0.648	0.663
Autoformer	All	**0.193**	0.227	0.252	0.252	0.629	0.647	0.676	**0.638**
	wo/Pos.	**0.193**	**0.201**	**0.214**	**0.246**	**0.613**	**0.616**	**0.622**	0.660
	wo/Temp.	0.206	0.243	0.303	0.302	0.681	0.665	0.908	0.769
	wo/Pos.-Temp.	0.215	0.308	0.506	0.729	0.672	0.811	1.133	1.300
Informer	All	**0.274**	**0.296**	**0.300**	**0.373**	**0.719**	**0.696**	**0.777**	**0.864**
	wo/Pos.	0.436	0.599	0.599	0.670	1.035	1.186	1.307	1.472
	wo/Temp.	0.332	0.357	0.363	0.407	0.754	0.780	0.903	1.259
	wo/Pos.-Temp.	0.524	0.651	0.801	0.954	1.038	1.351	1.491	1.512

Impact of training data size. 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 given time series and it is difficult to scale up the training data size. At the same time, the size of the training data would indeed have a significant impact on the model performance. Therefore, we conduct an experiment on the Traffic dataset, comparing the performance of the model trained with a full dataset (17,544*0.7 hours), named *Ori.*, and that trained with a shortened dataset (8,760 hours, i.e., 1 year), called *Short*. As can be observed in Table 7, the prediction errors with reduced training data are lower in most cases. While we cannot conclude that we should use less data for training, it demonstrates that the training data size is not the limiting factor for the performances of Autoformer and FEDformer.

Table 7: Comparison of forecasting errors (MSE) with different training sizes.

Methods	*DLinear*		FEDformer		Autoformer	
Dataset	*Ori.*	*Short*	*Ori.*	*Short*	*Ori.*	*Short*
96	0.650	**0.631**	0.587	**0.568**	0.613	**0.594**
192	0.598	**0.582**	0.604	**0.584**	**0.616**	0.621
336	0.605	**0.589**	0.621	**0.601**	0.622	**0.621**
720	0.645	**0.628**	0.626	**0.608**	0.660	**0.650**

Efficiency comparison. Existing Transformer-based TSF methods focus on reducing the $O\left(L^2\right)$ complexity of the vanilla Transformer. Although they prove to be able to improve the theoretical time and memory complexity, 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., we use one NVIDIA Titan Xp here).

In Table 8, we compare both the theoretical and practical efficiency. Interestingly, compared with the vanilla Transformer (with the same DMS decoder), most Transformer variants incur similar or even worse inference time and parameters in practice, due to the additional design elements introduced in their model. Moreover, the memory cost of the vanilla Transformer is practically acceptable, even for output length $L = 720$, which weakens the importance of deriving a memory-efficient Transformer-based method.

5.5 Ablation study on the *DLinear*

Impact of the decomposition strategy. In this experiment, we explore the impact of the decomposition operation of *DLinear* on different benchmarks. Accordingly, we remove the decomposition scheme, and the remained model is simply a one-layer linear network to transform the historical time

Table 8: Comparison of method efficiency on the Electricity dataset with a look-back window size of 96 and forecasting horizon of 720 steps. MACs are the number of multiply-accumulate operations. The inference time is an average result of 5 runs.

Method	MACs	Parameter	Time	Memory	Time	Memory	Test Step
DLinear	0.04G	139.7K	0.4ms	687MiB	$O(L)$	$O(L)$	1
Transformer×	4.03G	13.61M	26.8ms	6091MiB	$O(L^2)$	$O(L^2)$	1
Informer	3.93G	14.39M	49.3ms	3869MiB	$O(LlogL)$	$O(LlogL)$	1
Autoformer	4.41G	14.91M	164.1ms	7607MiB	$O(LlogL)$	$O(LlogL)$	1
Pyraformer	0.80G	241.4M*	3.4ms	7017MiB	$O(L)$	$O(L)$	1
FEDformer	4.41G	20.68M	40.5ms	4143MiB	$O(L)$	$O(L)$	1

- $\times$ is modified into the same one-step decoder, which is implemented in the source code from Autoformer.
- $*$ 236.7M parameters of Pyraformer come from its linear decoder.

series into the forecasting results. In Table 9, we observe that the decomposition scheme is beneficial when there is a clear trend in the specific dataset, e.g., Exchange Rate, ILI, and ETT, and it is not effective for those datasets without a clear trend such as Traffic and Weather, which is expected.

Table 9: The impact of the decomposition strategy on *DLinear*. wo/D. means without decomposition (only one linear layer). T indicates the forecasting length. The metric used is MSE. The look-back window size is $L = 36$ for ILI and $L = 96$ for the others.

Dateset	ETTh1		ETTh2		ETTm1		ETTm2		Exchange-Rate		Electricity		Traffic		Weather		ILI		
T	*DLinear*	wo/D.	*DLinear*	wo/D.	*DLinear*	wo/D.	*DLinear*	wo/D.	*DLinear*	wo/D.	*DLinear*	wo/D.	*DLinear*	wo/D.	*DLinear*	wo/D.	T	*DLinear*	wo/D.
96	**0.386**	0.434	**0.295**	0.314	**0.345**	0.354	**0.183**	0.189	**0.078**	0.080	**0.194**	0.195	0.650	**0.649**	**0.196**	0.199	24	**2.398**	2.655
192	**0.437**	0.490	**0.452**	0.458	**0.380**	0.423	0.260	**0.256**	**0.159**	0.168	**0.193**	0.194	**0.598**	**0.598**	**0.237**	0.239	36	**2.642**	2.702
336	**0.481**	0.529	**0.504**	0.516	**0.413**	0.486	**0.336**	0.346	0.274	**0.271**	**0.206**	0.207	**0.605**	**0.605**	0.283	**0.282**	48	**2.614**	2.684
720	**0.519**	0.647	**0.577**	0.705	**0.474**	0.547	**0.415**	0.674	**0.558**	0.603	**0.242**	**0.242**	**0.645**	**0.645**	**0.345**	0.348	60	2.804	**2.627**

Weight visualization. The weight visualization of *DLinear* can reveal certain characteristics in the data used for forecasting. In addition to the visualizations given in the main paper, we visualize the trend and remainder weights of other datasets with a fixed input length of 96 and four different forecasting horizons.

Overall, we find that *DLinear* utilizes different characteristics of the corresponding dataset for forecasting. To be specific, Figure 6(a), Figure 6(b), and Figure 6(d) show the periodicity of all weights is 24, where these data have hourly granularity. In Figure 6(e), the periodicity of all weights is 144 (1 day is $144 * 10$ minutes). Besides, the visualization can also reveal the periodicity of different temporal scales. For Traffic data, as shown in Figure 6(d), the model gives high weights to the latest time step of the look-back window for the 0,23,47...719 forecasting steps. Among these forecasting time steps, the 0, 167, 335, 503, 671 time steps have higher weights. Note that, 24 time steps are a day and 168 time steps is a week. This indicates that Traffic has a daily periodicity and a weekly periodicity.

6 Conclusion and Future Work

Conclusion. This work questions the effectiveness of emerging favored Transformer-based solutions for long-term time series forecasting. We introduce an embarrassingly simple linear model *DLinear* as a DMS forecasting baseline for comparison. Our results show that *DLinear* outperforms existing Transformer-based solutions on nine widely-used benchmarks in most cases, often by a large margin.

Future work. *DLinear* model has limited model capacity, and it merely serves a simple yet competitive baseline with strong interpretability for future research. For example, the one-layer linear network cannot capture the temporal dynamics caused by change points [24]. Consequently, we believe there is much potential for new model designs to tackle the challenging LTSF problem on complex time series. Moreover, with Transformers' questionable temporal relation extraction capability, we advocate revisiting the validity of Transformer-based solutions for other time series analysis tasks (e.g., classification and anomaly detection) in the future.

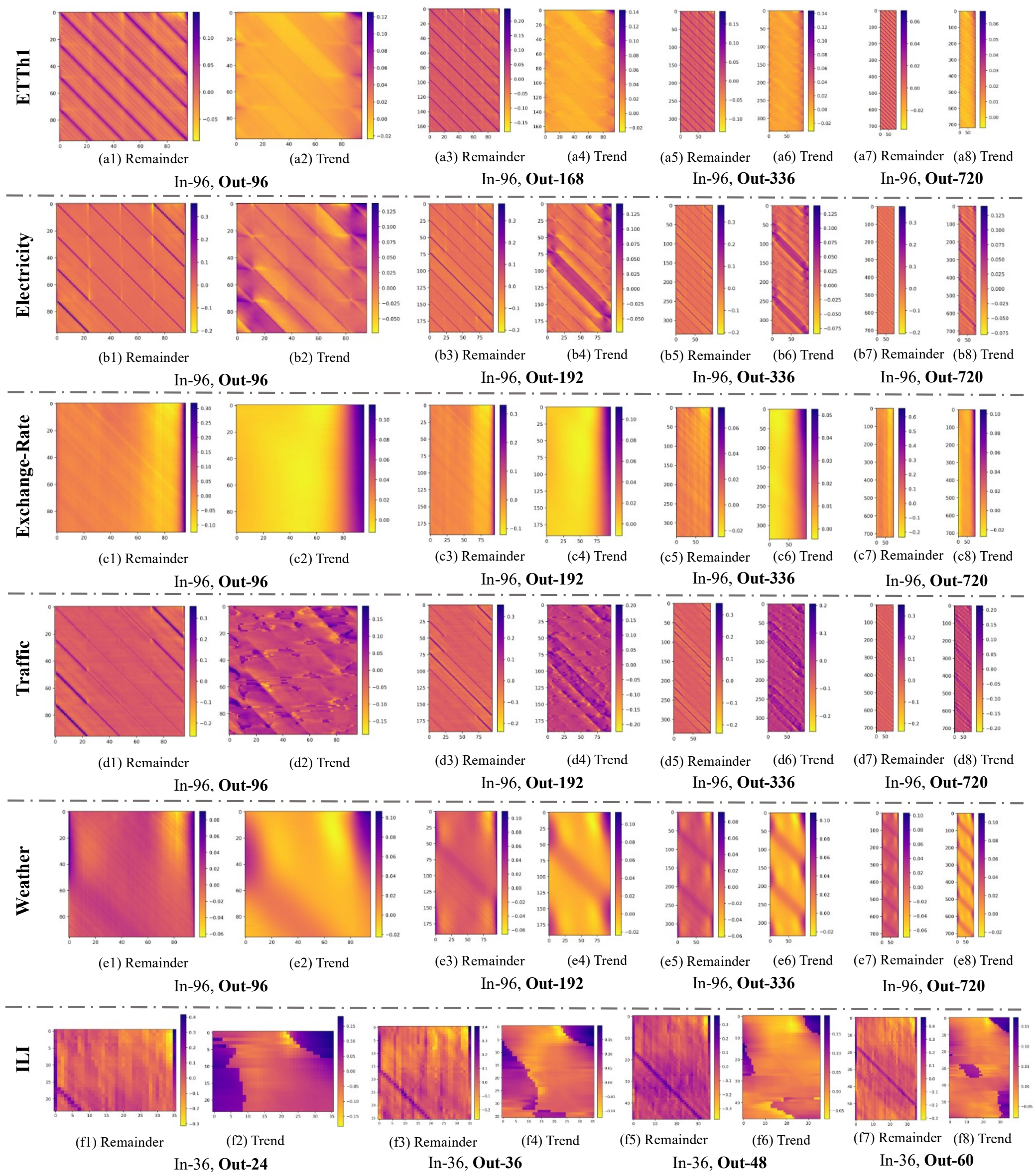


Figure 6: Visualization of the weights(T*L) of *DLinear* on several benchmarks. Models are trained with a look-back window L (X-axis) and different forecasting time steps T (Y-axis). We show weights in the remainder and trend layer.

Appendix:

In this appendix, we provide detailed experimental settings in Section A, more comparisons under different look-back window sizes in Section B.

A Experimental Details

A.1 Data Descriptions

We use nine wildly-used benchmarks in the main paper. We detail them in the following.

- ETT (Electricity Transformer Temperature) [28][2] consists of two hourly-level datasets (ETTh) and two 15-minute-level datasets (ETTm). Each of them contains seven oil and load features of electricity transformers from July 2016 to July 2018.
- Traffic[3] describes the road occupancy rates. It contains the hourly data recorded by the sensors of San Francisco freeways from 2015 to 2016.
- Electricity[4] collects the hourly electricity consumption of 321 clients from 2012 to 2014.
- Exchange-Rate [15][5] collects the daily exchange rates of 8 countries from 1990 to 2016.
- Weather[6] includes 21 indicators of weather, such as air temperature, and humidity. Its data is recorded every 10 min for 2020 in Germany.
- ILI[7] describes the ratio of patients seen with influenza-like illness and the total number of the patients. It includes the weekly data from the Centers for Disease Control and Prevention of the United States from 2002 to 2021.

A.2 Implementation Details

For existing Transformer-based TSF solutions: the implementation of Autoformer [27], Informer [28], and the vanilla Transformer [25] are all taken from the Autoformer work [27]; the implementation of FEDformer [29] and Pyraformer [18] are from their respective code repository. We also adopt their default hyper-parameters to train the models. For *DLinear*, to obtaining a smooth weight with a clear pattern, we initialize the weights of the linear layers in *DLinear* as $1/L$ rather than random initialization. That is, we use the same weight for every forecasting time step in the look-back window at the start of training. For more hyper-parameters of *DLinear*, please refer to our code. The decomposition scheme of *DLinear* is the same as Autoformerm, where the moving-average kernel size is 25.

B Additional Comparison with Transformers

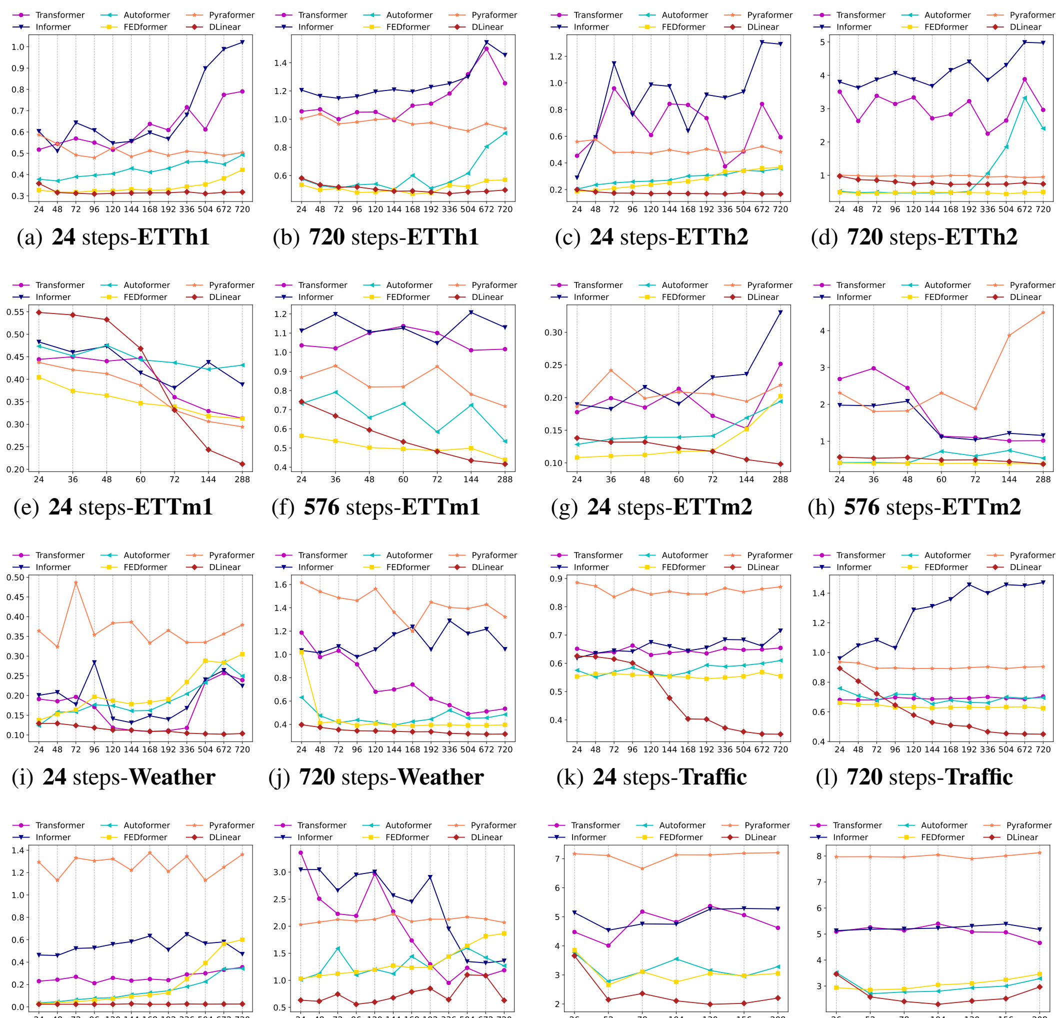
In Figure 5 of the main paper, we demonstrate that existing Transformers fail to exploit large look-back window sizes with two examples. Here, we give comprehensive comparisons between *DLinear* and Transformer-based TSF solutions under various look-back window sizes *on all benchmarks*.

B.1 Comparison under Different Look-back Windows

For hourly granularity datasets (ETTh1, ETTh2, Traffic, and Electricity), we set the look-back window size as {24, 48, 72, 96, 120, 144, 168, 192, 336, 504, 672, 720}, which represents {1, 2, 3, 4, 5, 6, 7, 8, 14, 21, 28, 30} days. The forecasting steps is {24, 720}, which means {1, 30} days. For 5-minute granularity datasets (ETTm1 and ETTm2), we set the look-back window size as {24, 36, 48, 60, 72, 144, 288}, which represents {2, 3, 4, 5, 6, 12, 24} hours. For 10-minute granularity datasets (Weather), we set the look-back window size as {24, 48, 72, 96, 120, 144, 168, 192, 336, 504, 672, 720}, which represents {4, 8, 12, 16, 20, 24, 28, 32, 56, 84, 112, 120} hours. The forecasting steps is {24, 720}, which are {4, 120} hours. For weekly granularity dataset (ILI), we set the look-back window size as {26, 52, 78, 104, 130, 156, 208}, which represents {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4} years. The forecasting steps is {26, 208}, which are {0.5, 4} years.

As can be observed in Figure 7, with increased look-back window sizes, the performance of *DLinear* is significantly boosted for most datasets, while this is not the case for Transformer-based TSF solutions. The results of Exchange-Rate and ILI datasets do not show improved results with a long look-back window, and we attribute it to the lack of seasonal information in these two datasets.

[2] `https://github.com/zhouhaoyi/ETDataset`
[3] `http://pems.dot.ca.gov`
[4] `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`
[5] `https://github.com/laiguokun/multivariate-time-series-data`
[6] `https://www.bgc-jena.mpg.de/wetter/`
[7] `https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html`



(a) **24** steps-**ETTh1** (b) **720** steps-**ETTh1** (c) **24** steps-**ETTh2** (d) **720** steps-**ETTh2**

(e) **24** steps-**ETTm1** (f) **576** steps-**ETTm1** (g) **24** steps-**ETTm2** (h) **576** steps-**ETTm2**

(i) **24** steps-**Weather** (j) **720** steps-**Weather** (k) **24** steps-**Traffic** (l) **720** steps-**Traffic**

(m) **24** steps-**Exchange** (n) **720** steps-**Exchange** (o) **24** steps-**ILI** (p) **60** steps-**ILI**

Figure 7: The MSE results (Y-axis) of models with different look-back window sizes (X-axis) of the long-term forecasting (e.g., **720**-time steps) and the short-term forecasting (e.g., **24** time steps) on different benchmarks.

References

[1] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv: Computation and Language*, 2014.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

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

[5] R. B. Cleveland. Stl : A seasonal-trend decomposition procedure based on loess. *Journal of Office Statistics*, 1990.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[8] Ruijun Dong and Witold Pedrycz. A granular time series approach to long-term forecasting and trend forecasting. *Physica A: Statistical Mechanics and its Applications*, 387(13):3253–3270, 2008.

[9] Shereen Elsayed, Daniela Thyssens, Ahmed Rashed, Hadi Samer Jomaa, and Lars Schmidt-Thieme. Do we really need deep learning models for time series forecasting? *arXiv preprint arXiv:2101.02118*, 2021.

[10] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[11] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

[12] James Douglas Hamilton. *Time series analysis*. Princeton university press, 2020.

[13] Andrew C Harvey. Forecasting, structural time series models and the kalman filter. 1990.

[14] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

[15] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. *international acm sigir conference on research and development in information retrieval*, 2017.

[16] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] Minhao Liu, Ailing Zeng, Zhijian Xu, Qiuxia Lai, and Qiang Xu. Time series is a special sequence: Forecasting with sample convolution and interaction. *arXiv preprint arXiv:2106.09305*, 2021.

[18] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2021.

[19] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586, 2021.

[20] Gábor Petneházi. Recurrent neural networks for time series forecasting. *arXiv preprint arXiv:1901.00069*, 2019.

[21] David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2017.

[22] Souhaib Ben Taieb, Rob J Hyndman, et al. *Recursive and direct multi-step forecasting: the best of both worlds*, volume 19. Citeseer, 2012.

[23] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *PeerJ Prepr.*, 2017.

[24] Gerrit JJ van den Burg and Christopher KI Williams. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[27] Jiehui Xu, Jianmin Wang, Mingsheng Long, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34, 2021.

[28] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 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, pages 11106–11115. AAAI Press, 2021.

[29] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 2022.