Revisiting Long-term Time Series Forecasting: An Investigation on Linear Mapping

Zhe Li, Shiyi Qi, Yiduo Li, Zenglin Xu
Harbin Institute of Technology, Shenzhen
{plum271828, syqi12138, liyiduo5, zenglin}@gmail.com

Abstract

Long-term time series forecasting has gained significant attention in recent years. While there are various specialized designs for capturing temporal dependency, previous studies have demonstrated that a single linear layer can achieve competitive forecasting performance compared to other complex architectures. In this paper, we thoroughly investigate the intrinsic effectiveness of recent approaches and make three key observations: 1) linear mapping is critical to prior long-term time series forecasting efforts; 2) RevIN (reversible normalization) and CI (Channel Independent) play a vital role in improving overall forecasting performance; and 3) linear mapping can effectively capture periodic features in time series and has robustness for different periods across channels when increasing the input horizon. We provide theoretical and experimental explanations to support our findings and also discuss the limitations and future works. Our framework's code is available at `https://github.com/plumprc/RTSF`.

1 Introduction

Time series forecasting has become increasingly popular in recent years due to its applicability in various fields, such as electricity forecasting [9], weather forecasting [2], and traffic flow estimation [5]. With the advances in computing resources, data volume, and model architectures, deep learning techniques, such as RNN-based [11, 20] and CNN-based models [4, 22], have outperformed traditional statistical methods [1, 3] in terms of higher capacity and robustness.

Recently, there have been increasing interests in using Transformer-based methods to capture long-term temporal correlations in time series forecasting [31, 26, 33, 15, 21, 30]. These methods have demonstrated promising results through various attention mechanisms and non-autoregressive generation (NAR) techniques. However, a recent work LTSF-Linear family [27] has shown that these Transformer-based methods may not be as effective as previously thought and found that their reported forecasting results may mainly rely on one-pass prediction compared to autoregressive generation. Instead, the LTSF-Linear, which uses only a single linear layer, surprisingly outperformed existing complex architectures by a large margin. Built on this work, subsequent approaches [14, 18, 13, 25] discarded the encoder-decoder architecture and focused on developing temporal feature extractors and modeling the mapping between historical inputs and predictions. While these methods have achieved improved forecasting performance, they are still not significantly better than linear models. Additionally, they often require a large number of adjustable hyper-parameters and specific training tricks, such as normalization and channel-specific processing, which may potentially affect the fairness of comparison. Based on these observations, we raise the following questions: (1) Are temporal feature extractors effective for long-term time series forecasting? (2) What are the underlying mechanisms explaining the effectiveness of linear mapping in time series forecasting? and (3) What are the limits of linear models and how can we improve them?

Preprint. Under review.

In the following sections, after introducing problem definition and experimental setup, we conduct a comprehensive investigation with intensive experiments and analysis into the inner working mechanisms of recent time series forecasting models, aiming to answer these questions raised above through extensive experiments and theoretical analysis. The main contributions of this paper are:

- We investigate the efficacy of different components in recent time series forecasting models, finding that linear mapping is critical to their forecasting performance, as shown in Sec. 3.
- We demonstrate the effectiveness of linear mapping for learning periodicity in long-term time series forecasting tasks with both theoretical and experimental evidence and propose simple yet effective baselines for a fairer comparison in the future (as shown in Table 3).
- We examine the limitations of the linear mapping when dealing with multivariate time series with different periodic channels, and analysis the impact of the input horizon and a remedial technique called Channel-Independent, as shown in Figures 10 and 11.

2 Problem Definition and Experimental Setup

Problem definition. Given a historical time series observation $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{c \times n}$ with c channels and n time steps, forecasting tasks aim to predict the next m time steps $\mathbf{Y} = [\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \dots, \boldsymbol{x}_{n+m}] \in \mathbb{R}^{c \times m}$ where m denotes forecasting horizon. We need to learn a map $\mathcal{F} : \mathbf{X}^{c \times n} \mapsto \mathbf{Y}^{c \times m}$ where $\mathbf{X}$ and $\mathbf{Y}$ are consecutive in the original time series data.

Experimental setup. Our experiments are conducted on simulated time series and six public real-world datasets: (1) ETT [31] (Electricity Transformer Temperature) with four datasets with different granularity records six power load features and oil temperature from electricity transformers; (2) Weather[1] contains 21 meteorological indicators in the 2020 year from nearly 1600 locations in the U.S.; and (3) ECL[2] records the hourly electricity consumption of 321 customers from 2012 to 2014. For a fair comparison, we follow the same evaluation protocol in [18] and split all datasets into training, validation, and test sets. Our proposed baselines are trained using the L2 loss and the Adam [17] optimizer. The training process is early stopped within 20 epochs. MSE (Mean Squared Error) and MAE (Mean Absolute Error) are adopted as evaluation metrics for comparison. R-squared score is used for empirical study as it can eliminate the impact of data scale. All the models are implemented in PyTorch [19] and tested on a single Nvidia V100 32GB GPU for three times.

3 Are Temporal Feature Extractors Effective?

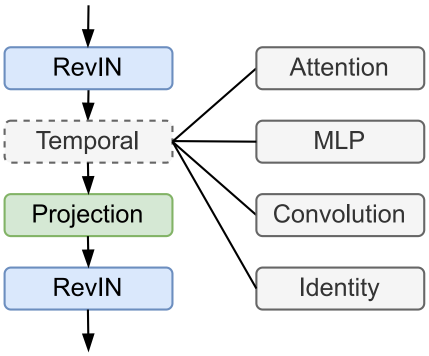

Figure 1: The general framework for time series forecasting, comprising of RevIN [10], a temporal feature extractor, and a linear projection layer.

General framework. Figure 1 illustrates the general framework of recent works [14, 32, 6, 13, 18, 25] for time series forecasting, comprising three core components: RevIN [10], a reversible normalization layer; a temporal feature extractor such as attention, MLP, or convolutional layers; and a linear projection layer that projects the final prediction results. Given the potential impact of hyper-parameter adjustment and various training tricks on comparison fairness, we first examine the effectiveness of different temporal feature extractors. Without loss of generality, we meticulously select four notable recent developments: PatchTST [18] (attention), MTS-Mixers [13] (MLP), TimesNet [25] and SCINet [14] (convolution). All of these methods follow this common framework and have achieved state-of-the-art forecasting performance, as claimed. Considering the fact that their reported forecasting accuracy is not significantly better than a single linear layer, we conduct new experiments using the ETT benchmark to check the contribution of each part in their proposed approaches. Figure 2 demonstrates the forecasting performance of different models on

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

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

ETTh1 over 4 different prediction lengths. The baseline "RLinear" refers to a linear projection layer with RevIN. The fixed random extractor means that we only initialize the temporal feature extractor randomly and do not update its parameters in the training phase. It is worth noting that the RevIN significantly improves the forecasting accuracy of these approaches. Thus, comparing one method with others which do not use RevIN may lead to unfair results due to its advantage. With the aid of RevIN, even a simple linear layer can outperform current state-of-the-art baseline PatchTST.

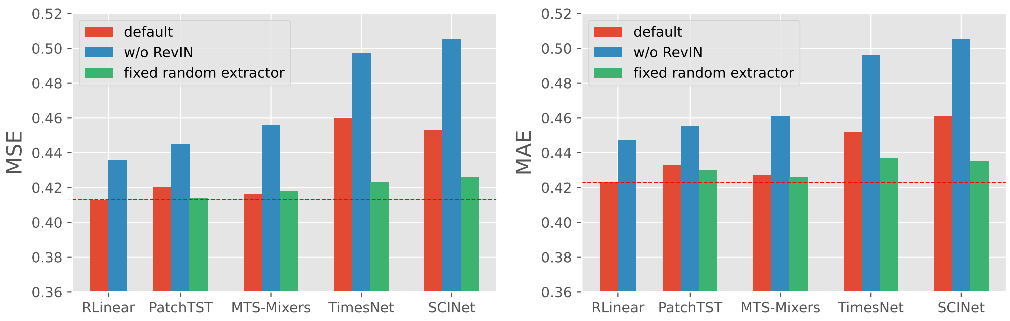

Figure 2: Forecasting results of selected models on ETTh1 [31] dataset. MSE and MAE results are averaged from 4 different prediction lengths {96, 192, 336, 720}. The lower MSE and MAE indicate the better forecasting performance.

Remarkably, our findings suggest that even using a randomly initialized temporal feature extractor with untrained parameters can induce competitive, even better forecasting results. It is necessary to consider what these feature extractors have learned from time series data. Figure 3 illustrates the weights of the final linear projection layer and different temporal feature extractors, such as MLP and attention on ETTh1. Interestingly, when the temporal feature extractor is a MLP, both MLP and projection layer learn chaotic weights, whereas the product of the two remains consistent with the weights learned from a single linear layer. On the other hand, when the temporal feature extractor is attention, it also learns about messy weights, but the weight learned by the projection layer is similar to that of a single linear layer, implying the importance of linear projection in time series forecasting.

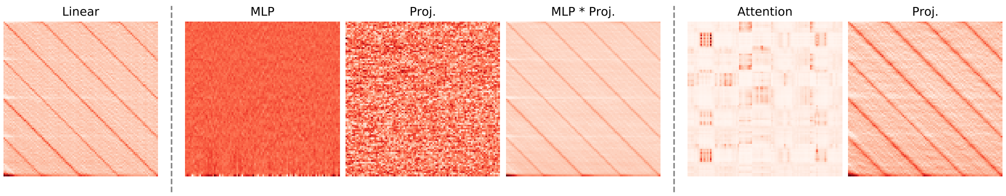

Figure 3: Weights visualization on ETTh1 where the input and output length are set as 96.

To mitigate any potential dataset-specific bias, we conducted more experiments on the full ETT benchmarks, using the same comparison protocol as [18]. Table 1 demonstrates the forecasting results of RLinear and selected models. Interestingly, the simple baseline RLinear has comparable or even better performance in most cases compared to carefully designed methods. Sometimes, these delicate models using temporal feature extractors even perform worse than an untrained prototype. It is important to note that models with a fixed random temporal feature extractor generally exhibit similar forecasting performance, and approach a single linear layer. These intriguing observations prompt us to question whether temporal feature extractors are necessary, and why linear mapping is so effective in long-term time series forecasting.

4 Theoretical and Empirical Study on the Linear Mapping

4.1 Roles of Linear Mapping in Forecasting

Linear mapping learns periodicity. Consider a single linear layer as

$$\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{b}, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n \times m}$ is the weight, also termed as the transition matrix, and $\mathbf{b} \in \mathbb{R}^{1 \times m}$ is the bias.

Table 1: Forecasting results on the full ETT benchmarks. The length of the historical horizon and the prediction length are set as 336. † indicates the temporal feature extractor with fixed random weights.

Dataset	ETTh1		ETTm1		ETTh2		ETTm2	
Method	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
RLinear	*0.420*	*0.423*	*0.370*	*0.383*	*0.325*	*0.386*	*0.273*	*0.326*
PatchTST	0.431	0.436	**0.366**	0.392	0.331	**0.380**	**0.276**	0.332
†PatchTST	**0.429**	**0.435**	0.371	**0.389**	**0.328**	0.384	0.280	**0.331**
MTS-Mixers	**0.414**	0.425	0.378	0.399	0.353	0.407	0.291	0.337
†MTS-Mixers	0.423	**0.424**	**0.377**	**0.392**	**0.351**	**0.405**	**0.282**	**0.334**
TimesNet	0.493	0.468	0.406	0.418	0.358	0.420	**0.304**	0.353
†TimesNet	**0.428**	**0.439**	**0.384**	**0.400**	**0.342**	**0.406**	0.306	**0.351**
SCINet	0.467	0.469	0.404	0.423	0.365	0.414	0.329	0.369
†SCINet	**0.428**	**0.431**	**0.386**	**0.398**	**0.349**	**0.403**	**0.299**	**0.345**

Assumption 1. *A general time series $x(t)$ can be disentangled into seasonality part $s(t)$ and trend part $f(t)$ with tolerable noise [8, 28], denoted as $x(t) = s(t) + f(t) + \epsilon$.*

Numerous methods [26, 33, 23, 24, 27] have been developed to decompose time series into seasonal and trend terms, leveraging neural networks to capture periodicity and supplement trend prediction. However, it's worth noting that a single linear layer can also effectively learn periodic patterns.

这里其实说了说了两件事情,
1) 季节性的时序数据可能满足平移性
2) 季节性的时序数据满足线性性质
即ax+b的这种特性

Theorem 1. *Given a seasonal time series satisfying $x(t) = s(t) = s(t-p)$ where $p \leq n$ is the period, there always exists an analytical solution for the linear model as*

$$[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \cdot \mathbf{W} + \mathbf{b} = [\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}], \tag{2}$$

$$\mathbf{W}_{ij}^{(k)} = \begin{cases} 1, & \textit{if } i = n - kp + (j \bmod p) \\ 0, & \textit{otherwise} \end{cases}, 1 \leq k \in \mathbb{Z} \leq \lfloor n/p \rfloor, b_i = 0. \tag{3}$$

Equation 3 indicates that linear mapping can predict periodic signals when the length of the input historical sequence is not less than the period, but that is not a unique solution. Since the values corresponding to each timestamp in $s(t)$ are almost impossible to be linearly independent, the solution space for the parameters of W is extensive. In particular, it is possible to obtain a closed-form solution for more potential values of $\mathbf{W}^{(k)}$ with different factor k when $n \gg p$. The linear combination of $[\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(k)}]$ with proper scaling factor also satisfies the solution of Equation 2.

Corollary 1.1. *When the given time series satisfies $x(t) = ax(t-p) + c$ where a, c are scaling and translation factors, the linear model still has a closed-form solution to Equation 2 as*

$$\mathbf{W}_{ij}^{(k)} = \begin{cases} a^k, & \textit{if } i = n - kp + (j \bmod p) \\ 0, & \textit{otherwise} \end{cases}, 1 \leq k \in \mathbb{Z} \leq \lfloor n/p \rfloor, b_i = \sum_{l=0}^{k-1} a^l \cdot c. \tag{4}$$

Now we know that a single linear layer can effectively capture periodicity in time series. Weights visualized in Figure 3 also support our viewpoint where the transition matrix from input to output shows significant periodicity (24 time steps per period). However, in practice, time series generally follow the Assumption 1 so that the trend term may affect the learning of linear models. Figure 4 illustrates the forecasting results of a linear layer on simulated seasonal and trend signals, including a sine wave, a linear function, and their sum. As expected, the linear model fits seasonality well but performs poorly on the trend, regardless of whether it has a bias term or not. Chen et al. [6] have also studied similar issues and provided an upper bound on the performance of linear models when forecasting time series with seasonal and trend components. Based on their work, we have adjusted their conclusion and derived the following theorem.

Theorem 2. *Let $x(t) = s(t) + f(t)$ where $s(t)$ is a seasonal signal with period p and $f(t)$ satisfies K-Lipschitz continuous. Then there exists a linear model as Equation 2 with input horizon size $n = p + \tau, \tau \geq 0$ such that $|x(n+j) - \hat{x}(n+j)| \leq K(p+j), j = 1, \ldots, m$.*

Proof. To simplify the proof process, we assume that the timestamp of historical data is 1 to n. Then for the j-th true value $x(n+j)$ to be predicted, we have

$$x(n+j) = x(p+\tau+j) = s(\tau+j) + f(p+\tau+j). \tag{5}$$

Supposing that the linear model can only learn periodic patterns, we can directly use Equation 3 as an approximate solution where we choose $k = 1$. Thus, the prediction for $x(n+j)$ is

$$\hat{x}(n+j) = \mathbf{XW} + \mathbf{b} = x(n-p+(j \bmod p)) = s(\tau+j) + f(\tau+(j \bmod p)). \tag{6}$$

Leveraging properties of K-Lipschitz continuous we can get

$$\begin{aligned} |x(n+j) - \hat{x}(n+j)| &= |f(p+\tau+j) - f(\tau+(j \bmod p))| \\ &\le K|p+j-(j \bmod p)| \\ &\le K(p+j). \end{aligned} \tag{7}$$

Although the forecasting error of linear models for trend terms is bounded, it can still impact prediction results as the timestamp accumulates or the trend term becomes more significant. This could potentially be why linear models prone to perform poorly in trend prediction.

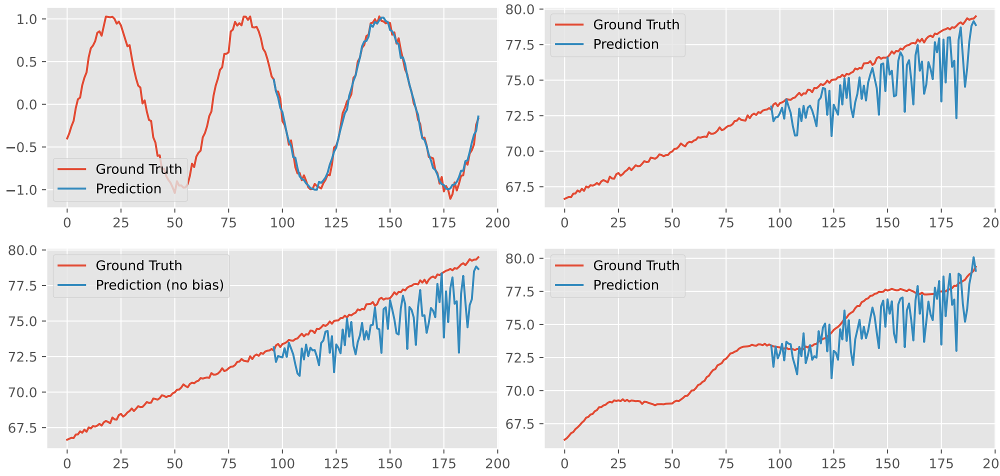

Figure 4: Forecasting visualization of a linear model on simulated seasonal and trend signals.

4.2 Disentanglement and Normalization

Problems in Disentanglement. If the trend term can be eliminated or separated from the seasonal term, forecasting performance can be improved. Previous works [8, 28, 26, 23, 24, 33, 29, 27] have focused on disentangling time series into seasonal and trend components to predict them individually. In general, they utilized the moving average implemented by an average pooling layer with a sliding window of a proper size to get trend information from the input time series. Then, they identified seasonal features from periodic signals obtained by subtracting trend items from the original data. However, these disentanglement methods have some problems as reported in [12]. Firstly, the sliding window size should be larger than the maximum period of seasonality parts, or the decoupling will be inadequate. Secondly, due to the usage of the average pooling layer, alignment requires padding on both ends of the input time series, which inevitably distorts the sequence at the head and tail. Besides, even if the signals are completely disentangled, or they only have trend terms, the issue of under-fitting trend terms persists. Therefore, while disentanglement may improve forecasting performance, it still has a gap with some recent advanced models.

Turning trend into seasonality. The key to disentanglement is subtracting the moving average from the original time series, which is related to normalization. Kim et al. [10] recognized that some statistical information of time series, such as mean and variance, continuously change over time due to the distribution shift problem. To address this challenge, they developed RevIN, a method that first normalizes the input historical time series and feeds them into forecasting modules before denormalizing the prediction for final results. Previous works [16, 7, 25, 18] have attributed the

文章主要就是说明了我们线性的分类器能够对季节项有很好的捕捉，但是对于趋势项很难捕捉
一些文章使用了AvgPooLing来提取这个趋势项，实际上是有很多问题的
本文提出了一种方式，如何让我们的trend转成季节项

RevIN就是通过归一化的方式来解决这种数据偏移的效果
可以看到Linear在Batchnorm, Disentagelement, RevIN的效果上差的很多，通过RevIN可以很好的解决问题

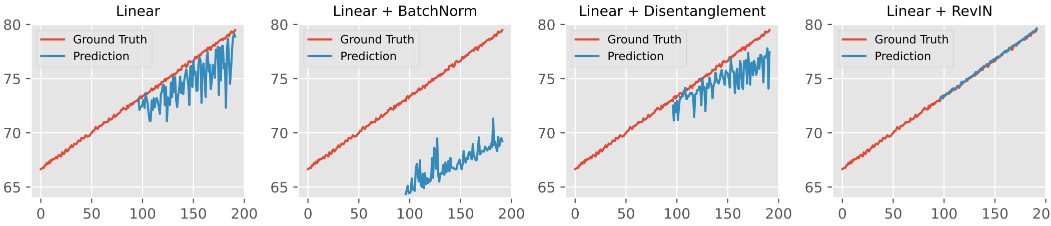

Figure 5: Forecasting results on the trend signal with different normalization methods.

effectiveness of RevIN more to normalization for alleviating distribution shift problems. However, the range and size of values in time series are also meaningful in real-world scenarios. Directly applying normalization to input data may erase this statistical information and lead to poor predictions. Figure 5 illustrates the forecasting results on the simulated trend signal with two channels using different normalization methods. It is challenging to fit trend changes solely using a linear layer. Applying batch normalization even induces worse results, and layer normalization results in meaningless prediction close to zero. Disentangling the simulated time series also does not work. However, with the help of RevIN, a single linear layer can accurately predict trend terms.

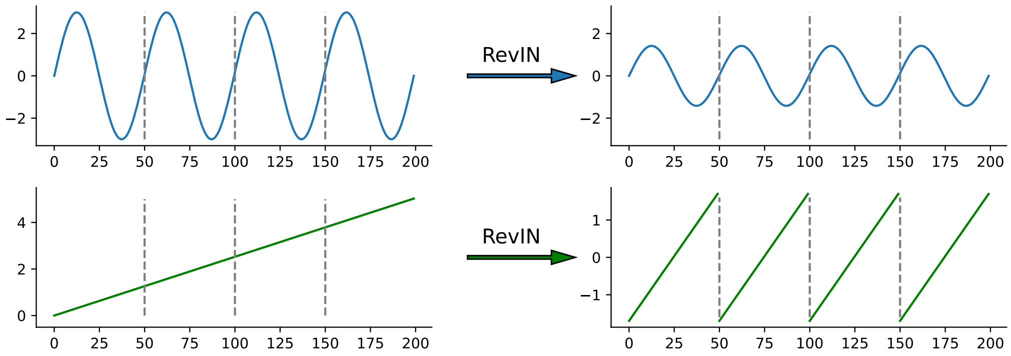

Figure 6: The effect of RevIN applied to seasonal and trend signals. Each segment separated by a dashed line contains the input historical time series X and the predicted sequence Y.

可逆归一化的核心在于可逆性
1) 消耗了统计引起的变化，并保留了用于复原的统计信息
2)对于季节性，只是缩放范围，不改变周期性
3)对于趋势项，将其转成了周期性的模式

这就极大改善了我们Linear的性能了，因为Linear对于这种趋势项不好预测，转成了周期性好，就很nice

其实文章做了两件事情，
1. 否定了各种MLP和Transformer这种特征提取器，说它们学的很混乱
2. 说明我们线性提取器好，因为最后学习的权重可以看到趋势项
3. 但是线性提取器对于季节提取敏感，对于权重提取不行
4. 现有的Linear线性映射层，使用了Autoformer里面的这种AvgPool，提取。其实是有缺陷的
5. 我们最重要的就是区别季节和趋势项，但是如果将RevIN的模型应用到Linear上，会得到一个更好的结果

The core of reversible normalization lies in reversibility. It eliminates trend changes caused by moment statistics while preserving statistical information that can be used to restore final forecasting results. Figure 6 illustrates how RevIN affects seasonal and trend terms. For the seasonal signal, RevIN scales the range but does not change the periodicity. For the trend signal, RevIN scales each segment into the same range and exhibits periodic patterns. RevIN is capable of turning some trends into seasonality, making models better learn or memorize trend terms. Figure 7 showcases forecasting results of the linear model with RevIN on simulated time series with seasonal and trend terms. RevIN converts continuously changing trends into multiple segments with a fixed and similar trend, demonstrating periodic characteristics. As a result, errors in trend prediction caused by accumulated timesteps in the past can be alleviated, leading to more accurate forecasting results.

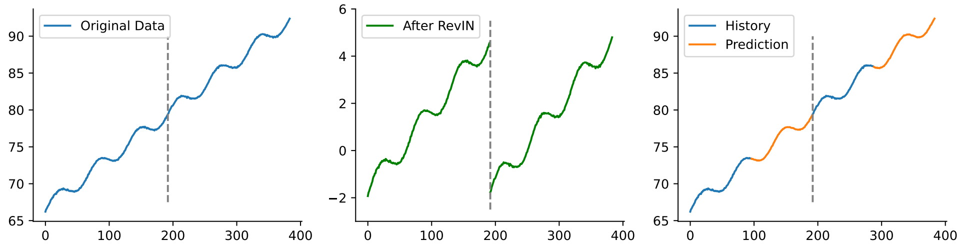

Figure 7: Forecasting results of a linear layer with RevIN on simulated time series with seasonal and trend terms. Each segment separated by a dashed line consists of historical and prediction sequences.

5 Experimental Evaluation

In this section, we first evaluate the performance of different models on real-world datasets, and then examine the scenarios with multiple periods among various channels.

5.1 Comparison on Real-world Datasets

Table 2 provides statistical information of those six real-world datasets. We perform experiments using three latest competitive baselines: PatchTST [18] (ICLR 2023), TimesNet [25] (ICLR 2023), and DLinear [27] (AAAI 2023). Given that RevIN significantly improves the forecasting performance, we add two simple baselines, RLinear and RMLP with two linear layers and a ReLU activation, for a fairer comparison. Table 3 provides an overview of forecasting results for all benchmarks.

Table 2: Statistical information of all datasets for time series forecasting.

Dataset	ETTh1/h2	ETTm1/m2	Weather	ECL
#Channel	7	7	21	321
Timesteps	17,420	69,680	52,696	26,304
Granularity	1 hour	15 minutes	10 minutes	1 hour
Data Partition	6:2:2 (month)		7:2:1	

Table 3: Time series forecasting results. The length of the historical horizon is 336 and prediction lengths are {96, 192, 336, 720}. The best results are in **bold** and the second one is underlined.

Method		RLinear		RMLP		PatchTST		TimesNet		DLinear	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.366**	**0.391**	0.390	0.410	0.381	0.405	0.398	0.418	0.375	0.399
	192	**0.404**	**0.412**	0.430	0.432	0.416	0.423	0.447	0.449	0.405	0.416
	336	**0.420**	**0.423**	0.441	0.441	0.431	0.436	0.493	0.468	0.439	0.443
	720	**0.442**	**0.456**	0.506	0.495	0.450	0.466	0.518	0.504	0.472	0.490
ETTm1	96	0.301	**0.342**	0.298	0.345	**0.293**	0.345	0.335	0.380	0.306	0.349
	192	**0.335**	**0.363**	0.344	0.375	**0.335**	0.372	0.358	0.388	0.339	0.368
	336	0.370	**0.383**	0.390	0.410	**0.366**	0.392	0.406	0.418	0.374	0.390
	720	0.425	**0.414**	0.445	0.441	**0.420**	0.424	0.449	0.443	0.428	0.423
ETTh2	96	**0.262**	**0.331**	0.288	0.352	0.276	0.337	0.348	0.392	0.289	0.353
	192	**0.319**	**0.374**	0.343	0.387	0.339	0.379	0.362	0.404	0.383	0.418
	336	**0.325**	0.386	0.353	0.402	0.331	**0.380**	0.358	0.420	0.448	0.465
	720	**0.372**	**0.421**	0.410	0.440	0.379	0.422	0.442	0.463	0.605	0.551
ETTm2	96	**0.164**	**0.253**	0.174	0.259	0.165	0.256	0.188	0.267	0.167	0.260
	192	**0.219**	**0.290**	0.236	0.303	0.238	0.305	0.252	0.308	0.224	0.303
	336	**0.273**	**0.326**	0.291	0.338	0.276	0.332	0.304	0.353	0.281	0.342
	720	**0.366**	**0.385**	0.371	0.391	0.369	0.391	0.405	0.409	0.397	0.421
Weather	96	0.175	0.225	**0.149**	**0.202**	0.155	0.205	0.172	0.220	0.176	0.237
	192	0.218	0.260	**0.194**	**0.242**	0.199	0.245	0.219	0.261	0.220	0.282
	336	0.265	0.294	**0.243**	**0.282**	0.249	0.284	0.280	0.306	0.265	0.319
	720	0.329	0.339	**0.316**	**0.333**	0.319	0.335	0.365	0.359	0.326	0.363
ECL	96	0.140	0.235	**0.129**	**0.224**	0.133	0.226	0.168	0.272	0.140	0.237
	192	0.154	0.248	**0.147**	**0.240**	0.149	0.242	0.184	0.289	0.153	0.249
	336	0.171	0.264	**0.164**	**0.257**	0.167	0.260	0.198	0.300	0.169	0.267
	720	0.209	0.297	**0.203**	**0.291**	0.205	0.293	0.220	0.320	0.210	0.310

However, these well-designed models are not better than our proposed two simple baselines. It is likely that the success of these models is due to the learning of periodicity via linear mapping and efficiency of reversible normalization. Interestingly, we have noticed that RLinear does not perform significantly better than complex models on datasets with a large number of channels, such as Weather and ECL, which will be studied in the next section.

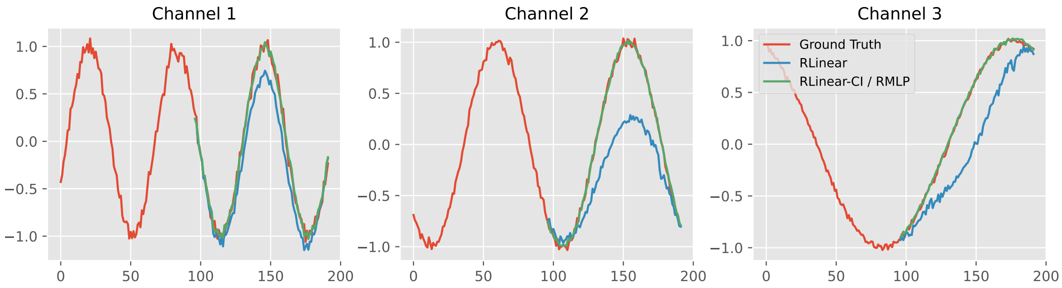

Figure 8: Forecasting results on simulated time series with three channels of different periods.

5.2 When Linear Meets Multiple Periods among Channels

Although linear mapping is capable of learning periodicity in time series, it faces challenges when dealing with multi-channel datasets. To address this issue, a possible solution is to use Channel Independent [18] (CI) modeling, which treats each channel in the time series independently. While this approach can improve forecasting accuracy, it also significantly increases computational overhead. Figure 8 illustrates the forecasting results of different models applied to simulated time series with three distinct periodic channels. It is observed that RLinear-CI and RMLP are able to fit curves, while RLinear fails. This suggests that a single linear layer may struggle to learn different periods within channels. Nonlinear units or CI modeling may be useful in enhancing the robustness of the model for multivariate time series with different periodic channels. Table 4 provides forecasting results on Weather and ECL of RLinear using CI, which achieves comparable performance with RMLP, confirming that a single linear layer may be vulnerable to varying periods among channels.

Table 4: Forecasting results on Weather and ECL of RLinear using CI where the input horizon is 336.

Dataset		Weather				ECL			
Method	Metric	96	192	336	720	96	192	336	720
RLinear	MSE MAE	0.175 0.225	0.218 0.260	0.265 0.294	0.329 0.339	0.140 0.235	0.154 0.248	0.171 **0.264**	0.209 0.297
RLinear-CI	MSE MAE	**0.146** **0.194**	**0.189** **0.235**	**0.241** **0.275**	**0.314** **0.327**	**0.134** **0.232**	**0.149** **0.246**	**0.166** 0.265	**0.202** **0.293**

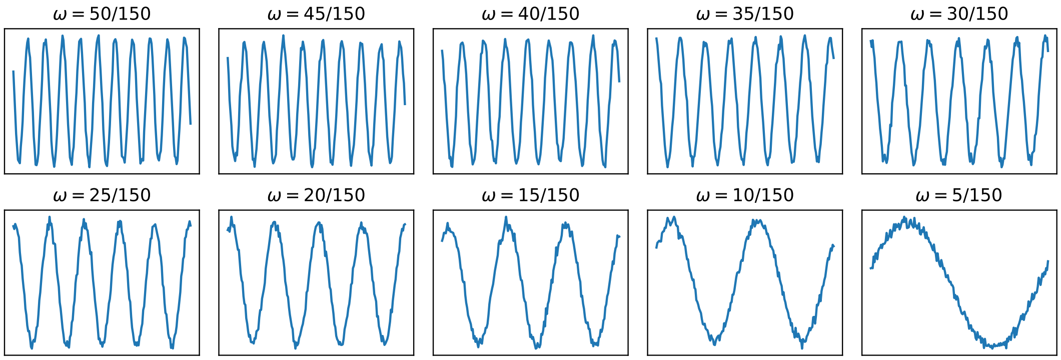

Figure 9: Simulated sine waves with angular frequency ranges from 1/30 to 1/3 and the length of 200.

To further investigate the effect of linear mapping on multivariate time series, we conduct simulations using a series of sine waves with angular frequencies ranging from 1/30 to 1/3 and the length of 3000. Figure 10 demonstrates the forecasting results under different settings. Our findings indicate that the linear model consistently performs well on time series with two channels, regardless of whether the difference in periodicity is small or large. However, as the number of channels with different periods increases, the linear model gradually performs worse, while models with nonlinear units or

CI continue to perform well. Additionally, increasing the input horizon can effectively alleviate the forecasting performance of the linear model on multi-channel datasets. These observations suggest that existing models may focus on learning seasonality, and that the differences in periodicity among different channels in multivariate time series are key factors that constrain forecasting performance. Theorem 3 provides an explanation of linear models in forecasting multivariate time series.

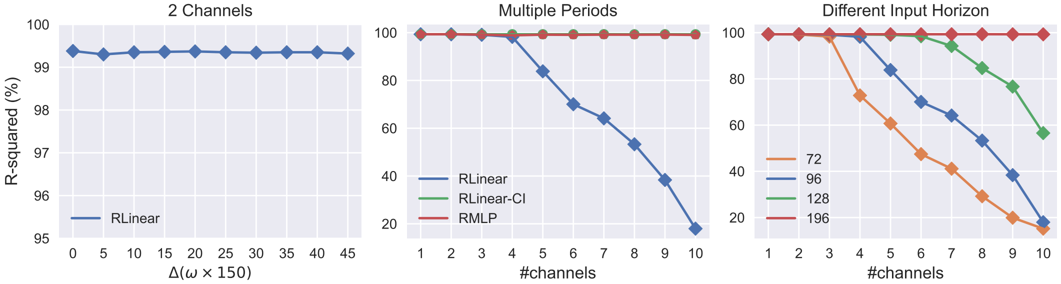

Figure 10: **Left**: Forecasting resutls on simulated 2-variate time series. $\Delta\omega$ denotes the difference in angular frequency between channels. **Middle**: Forecasting results of different models on simulated datasets with different periodic channels. **Right**: Impact of input horizon on forecasting performance.

Theorem 3. *Let* $\mathbf{X} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_c]^\top \in \mathbb{R}^{c \times n}$ *be the input historical multivariate time series with* c *channels and the length of* n. *If each signal* $\boldsymbol{s}_i$ *has a corresponding period* p_i, *there must be a linear model* $\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{b}$ *that can predict the next* m *time steps when* $n \geq lcm(p_1, p_2 \ldots, p_c)$.

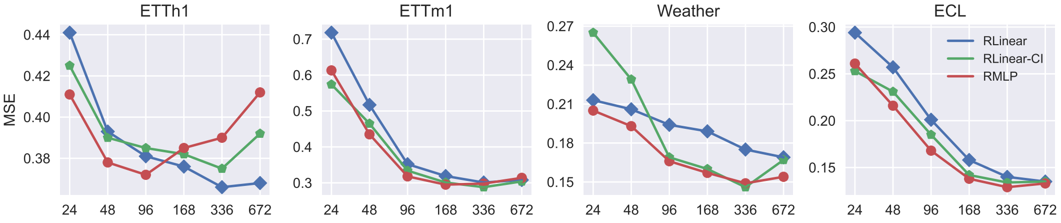

Figure 11: Impact of input horizon on forecasting results. Lower MSE indicates better performance.

As shown in Figure 11, increasing the input horizon can lead to a significant improvement in forecasting performance. This is because a longer input horizon covers more potential periods, minimizing the performance gap between linear models and those with nonlinear units. However, it is worth noting that RLinear-CI and RMLP perform worse on the ETTh1 dataset when the input horizon is longer, which may be due to the small volume of this particular dataset. Furthermore, it should be noted that there is an upper limit to the performance improvement achieved by increasing the input horizon. This limit may be highly dependent on the periodic patterns present in datasets.

6 Conclusion

This paper systematically investigate the effect of linear mapping in long-term time series forecasting, with the following important takeaways: (1) linear mapping is critical to prior long-term time series forecasting methods, where they generally prone to learn similar affine transform, which corresponds to specific periodic patterns, from input historical observation to output prediction; (2) RevIN (reversible normalization) and CI (Channel Independent) improve overall forecasting performance via simplifying learning about periodicity; and (3) linear mapping has robustness to fit multivariate time series with different periodic channels when increasing input horizon, while it may induce under-fitting of short period features. We provide theoretical explanations and conduct extensive experiments on both simulated and real-world datasets to support our findings.

Limitations and future work. Long-term time series benchmarks often display consistent seasonal patterns. To improve the model's generalization ability, it is worthwhile to study how it performs when the seasonality changes. It would also be valuable to explore the applicability of our theories to other tasks, such as short-term time series forecasting. We acknowledge that these explorations will be left for future work.

References

[1] Oliver D. Anderson, George E. P. Box, and Gwilym M. Jenkins. Time series analysis: Forecasting and control. *The Statistician*, 27(3/4):265, September 1978.

[2] Rafal A. Angryk, Petrus C. Martens, Berkay Aydin, Dustin J. Kempton, Sushant S. Mahajan, Sunitha Basodi, Azim Ahmadzadeh, Xumin Cai, Soukaina Filali Boubrahimi, Shah Muhammad Hamdi, Michael A. Schuh, and Manolis K. Georgoulis. Multivariate time series dataset for space weather data analytics. *Scientific data*, 7:227, 2020.

[3] 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.

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271, 2018.

[5] Chao Chen, Karl F. Petty, Alexander Skabardonis, Pravin Pratap Varaiya, and Zhanfeng Jia. Freeway performance measurement system: Mining loop detector data. *Transportation Research Record*, 1748:96–102, 2001.

[6] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *ArXiv*, abs/2303.06053, 2023.

[7] Wei Fan, Pengyang Wang, Dongkun Wang, Dongjie Wang, Yuanchun Zhou, and Yanjie Fu. Dish-ts: A general paradigm for alleviating distribution shift in time series forecasting. *ArXiv*, abs/2302.14829, 2023.

[8] Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20:5–10, 2004.

[9] Zulfiqar Ahmad Khan, Tanveer Hussain, Amin Ullah, Seungmin Rho, Mi Young Lee, and Sung Wook Baik. Towards efficient electricity forecasting in residential and commercial buildings: A novel hybrid cnn with a lstm-ae based framework. *Sensors*, 20:1399, 2020.

[10] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022.

[11] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[12] Zhe Li, Zhongwen Rao, Lujia Pan, Pengyun Wang, and Zenglin Xu. Ti-mae: Self-supervised masked time series autoencoders. *ArXiv*, abs/2301.08871, 2023.

[13] Zhe Li, Zhongwen Rao, Lujia Pan, and Zenglin Xu. Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing. *ArXiv*, abs/2302.04501, 2023.

[14] Minhao Liu, Ailing Zeng, Mu-Hwa Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

[15] 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*, 2022.

[16] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Neural Information Processing Systems*, 2022.

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[18] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[20] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[21] Mohammad Amin Shabani, Amir H. Abdi, Lili Meng, and Tristan Sylvain. Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

[22] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, 8:876, 2019.

[23] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*, 2022.

[24] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *ArXiv*, abs/2202.01381, 2022.

[25] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[27] Ailing Zeng, Mu-Hwa Chen, L. Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *ArXiv*, abs/2205.13504, 2022.

[28] G Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160:501–514, 2005.

[29] Xiyuan Zhang, Xiaoyong Jin, Karthick Gopalswamy, Gaurav Gupta, Youngsuk Park, Xingjian Shi, Hongya Wang, Danielle C. Maddix, and Yuyang Wang. First de-trend then attend: Rethinking attention for time-series forecasting. *ArXiv*, abs/2212.08151, 2022.

[30] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

[31] 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[32] Tian Zhou, Ziqing Ma, Xue Wang, Qingsong Wen, Liang Sun, Tao Yao, and Rong Jin. Film: Frequency improved legendre memory model for long-term time series forecasting. *ArXiv*, abs/2205.08897, 2022.

[33] 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*, pages 27268–27286. PMLR, 2022.

A Proofs

For better readability, we have re-listed the unproven theorems as follows.

Theorem 1. *Given a seasonal time series satisfying $x(t) = s(t) = s(t-p)$ where $p \leq n$ is the period, there always exists an analytical solution for the linear model as*

$$[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \cdot \mathbf{W} + \mathbf{b} = [\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}], \tag{8}$$

$$\mathbf{W}_{ij}^{(k)} = \begin{cases} 1, & \text{if } i = n - kp + (j \bmod p) \\ 0, & \text{otherwise} \end{cases}, 1 \leq k \in \mathbb{Z} \leq \lfloor n/p \rfloor, b_i = 0. \tag{9}$$

Proof. $\forall \boldsymbol{x}_{n+j}, 1 \leq j \leq m, z \in \mathbb{Z}^+, \boldsymbol{x}_{n+j} = \boldsymbol{x}_{n-zp+(j \bmod p)}$ according to $x(t) = x(t-p)$, thus we have $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \cdot \mathbf{W}^{(k)} = [\boldsymbol{x}_{n-kp+1}, \boldsymbol{x}_{n-kp+2}, \ldots, \boldsymbol{x}_{n-kp+m}] = [\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}]$. □

Corollary 1.1. *When the given time series satisfies $x(t) = ax(t-p) + c$ where a, c are scaling and translation factors, the linear model still has a closed-form solution to Equation 8 as*

$$\mathbf{W}_{ij}^{(k)} = \begin{cases} a^k, & \text{if } i = n - kp + (j \bmod p) \\ 0, & \text{otherwise} \end{cases}, 1 \leq k \in \mathbb{Z} \leq \lfloor n/p \rfloor, b_i = \sum_{l=0}^{k-1} a^l \cdot c. \tag{10}$$

Proof. $\forall \boldsymbol{x}_{n+j}, 1 \leq j \leq m, z \in \mathbb{Z}^+, \boldsymbol{x}_{n+j} = a^z \boldsymbol{x}_{n-zp+(j \bmod p)} + \sum_{l=0}^{z-1} a^l \cdot c$ according to $x(t) = ax(t-p)+c$, thus we have $[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \cdot \mathbf{W}^{(k)} + \mathbf{b} = [a^k \boldsymbol{x}_{n-kp+1} + \sum_{l=0}^{k-1} a^l \cdot c, \ldots, a^k \boldsymbol{x}_{n-kp+m} + \sum_{l=0}^{k-1} a^l \cdot c] = [\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+m}]$. □

Theorem 3. *Let $\mathbf{X} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_c]^\top \in \mathbb{R}^{c \times n}$ be the input historical multivariate time series with c channels and the length of n. If each signal $\boldsymbol{s}_i$ has a corresponding period p_i, there must be a linear model $\mathbf{Y} = \mathbf{XW} + \mathbf{b}$ that can predict the next m time steps when $n \geq lcm(p_1, p_2 \ldots, p_c)$.*

Proof. Apparently $p = \text{lcm}(p_1, p_2 \ldots, p_c)$ is the least common period for all channels. According to Equation 9, there must be a linear model satisfying Equation 8 for each channel $\boldsymbol{s}_c \in \mathbb{R}^{1 \times n}$. □

B Experimental details

Reproduction. We implemented the reversible normalization module using the default setting from RevIN [10]. Our baseline RLinear model consists of ReVIN and a single linear layer that maps the input to the output time series. The RMLP model, on the other hand, comprises RevIN, an MLP for temporal interaction, and a linear projection layer. The MLP includes two linear layers with ReLU activation, and we set the number of hidden states to 512. The baseline RLinear-CI includes c RLinears for modeling c channels individually.

Details on benchmarks and baselines. We adopt the same pre-processing protocol in [18]. Across all benchmarks, we set the initial learning rate to 0.005 and the batch size to 128. The results of PatchTST, MTS-Mixers, TimesNet, SCINet, and DLinear are based on our reproduction. For models with a fixed random temporal feature extractor setting, we initialize these models with the fixed random seed 1024. We use the hyper-parameters suggested in their original papers for each model.