Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift

Anonymous authors
Paper under double-blind review

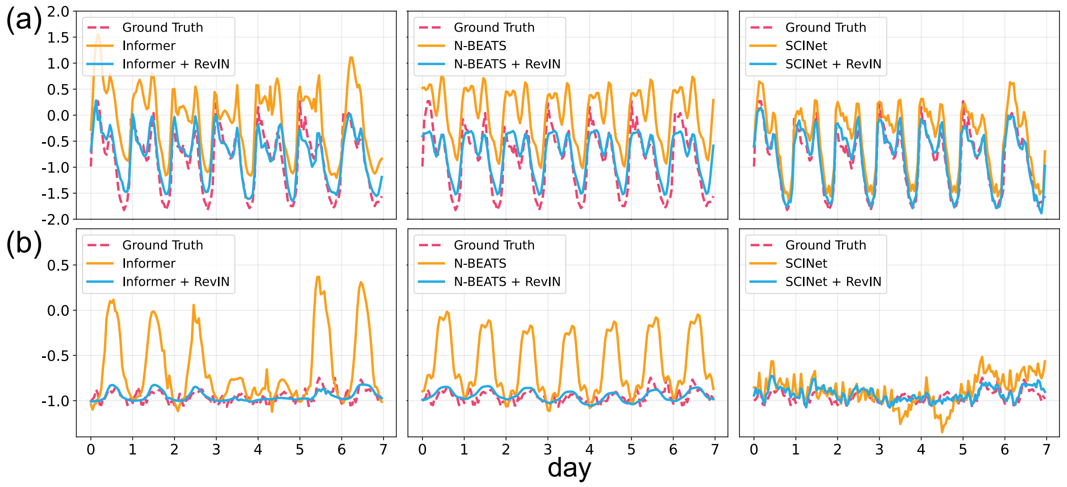

Figure 1: Multivariate time-series forecasting results of our method in comparison with the state-of-the-art baseline models, i.e., Informer (Zhou et al., 2021), N-BEATS (Oreshkin et al., 2020), and SCINet (Liu et al., 2021). The analysis is conducted on one of the variables of the electricity counting load dataset with the prediction length of seven days. The predictions of the baseline models are inaccurately (a) shifted and (b) scaled. Added to the baseline models, our method significantly improves their forecasting performance.

Abstract

The statistical properties in a time-series often change, i.e., the data distributions differ over time. This temporal distribution change is one of the main challenges that prevent accurate time-series forecasting. To address this issue, we propose a simple but effective normalization method called reversible instance normalization (RevIN). To be specific, RevIN consists of two different steps, normalization and denormalization. The former normalizes the input to fix its distribution in terms of the mean and variance, while the latter returns the output to the original distribution. In addition, RevIN is model-agnostic, generally applicable to various time-series forecasting models with significant improvements in forecasting performance. As shown in Fig. 1, RevIN effectively enhances the performance of the baselines. Our extensive experimental results verify the general applicability and performance improvements on various real-world datasets.

1 Introduction

Time-series forecasting plays a significant role in various daily problems such as health care, economics, and traffic engineering (Che et al., 2018; Bauer et al., 2016; Zhang et al., 2017). Recently, time-series forecasting models have achieved comparative performance on these problems by overcoming several challenges, e.g., long-term forecasting (Zhou et al., 2021) and missing value imputation (Tang et al., 2020). However, these time-series forecasting models often severely suffer from statistical properties that change over time, which is also widely known as a distribution shift problem; for example, the mean and variance of the time-series data distribution vary over time. Accordingly, the input sequences to the forecasting models have different underlying distributions from each other. In the case of the train and test data, they are usually divided into data before and after a specific point in time. Thus, the distributions of the two data often hardly overlap, which is commonly known as the reason for model performance degradation. Thus, we assume that if we normalize every sequence in the data to have identical mean and variance and provide the transformed data to the model, the discrepancy in the distribution will be reduced, thereby improving the model performance. However, if the input to the model is shifted and scaled, the prediction of the model can be shifted and scaled as much as the input; since this will differ from the groundtruth distribution, the accuracy of the model will decrease. Also, we further hypothesize that if the model output, shifted and scaled, can be returned to its original distribution explicitly, we can alleviate the discrepancy in the distribution of the data during the forward pass of the model as well as allow its prediction to obey the original distribution. Here, reconstructing the original distribution can be achieved by applying denormalization on the output of the model as a reversed function of the normalization applied on the input data. Inspired by these, we propose a simple yet effective normalization method, **reversible instance normalization (RevIN)**, which first normalizes the input sequences and then denormalizes the model output sequences to solve the time-series forecasting task against the distribution shift problem. RevIN is symmetrically structured to return the model output to the original distribution in the denormalization layer by scaling and shifting the model output as much as the input data is shifted and scaled in the normalization layer. Adding only the normalization and denormalization layers on the input and output of the model, respectively, RevIN can be applied to any model without modification to the model architecture. We empirically found that RevIN effectively alleviates the distribution discrepancy within the data, as well as significantly improves the model performance when applied to various state-of-the-art time-series forecasting methods. To verify the effectiveness of RevIN, we conduct extensive quantitative evaluations with various baselines: Informer (Zhou et al., 2021), N-BEATS (Oreshkin et al., 2020), and SCINet (Liu et al., 2021). We also present an in-depth analysis of the behavior of the model, including the verification of the assumptions on reversible instance normalization (See Section 3.2).

RevIN is a flexible, end-to-end trainable layer that can be applied to any arbitrarily chosen layers, effectively suppressing non-stationary information (mean and standard deviation of the instance) from one layer and restoring it on the other layer at a virtually symmetric position. Despite its remarkable performance, there has been no work on generalizing and expanding the instance-wise normalization-and-denormalization as a flexibly applicable, trainable layer in the time-series domain. Recently, deep learning-based time-series forecasting approaches, such as Informer (Zhou et al., 2021) and N-BEATS (Oreshkin et al., 2020), have shown outstanding performance in time-series forecasting. However, they overlooked the importance of normalization, merely using simple global preprocessing to the model input without further exploration and expecting their end-to-end deep learning model to replace the role. Despite the simplicity of our method, there have been no cases of using such techniques in modern deep-learning-based time-series forecasting approaches (Zhou et al., 2021; Liu et al., 2021; Oreshkin et al., 2020). In this sense, we enlight the importance of the appropriate normalization method in deep-learning-based time-series approaches; we propose a carefully designed, deep-learning-friendly module, for time-series forecasting, by combining the learnable affine transformation with the method, which has been widely accepted in recent deep-learning-based normalization work (Ulyanov et al., 2016).

In summary, our contributions are as follows:

- We propose a simple and effective normalization-and-denormalization method for time-series, called RevIN, symmetrically structured to remove and restore the statistical information of a time-series instance; it is a generally-applicable layer to arbitrary deep neural networks with negligible cost and significantly improves the performance by reducing the distribution discrepancy within the data.
- By adding RevIN to the baseline, we achieve the state-of-the-art performance on four large-scale real-world datasets with a significant margin.
- We conduct extensive evaluations on RevIN with quantitative analysis and qualitative visualizations to verify its effectiveness, addressing the distribution shift problem.

2 Related Work

Time-series forecasting. Time-series forecasting methods are mainly categorized into three distinct approaches: (1) statistical methods, (2) hybrid methods, and (3) deep learning-based methods. As an example of the statistical models, exponential smoothing forecasting (Holt, 2004; Winters, 1960) is a well-established benchmark to predict future values. Statistical models have several advantages, e.g., interpretability and theoretically well guaranteed. In order to boost the performance, recent work proposed a hybrid model (Smyl, 2020) which incorporates a deep learning module with a statistical model. It achieved better performance over the statistical methods in the M4 time-series forecasting competition. The deep learning-based method basically follows the sequence-to-sequence framework to model the time-series forecasting. Initially, deep learning-based models utilized variations of recurrent neural networks (RNNs). However, to overcome the limitation of the limited receptive field, several studies utilize advanced techniques, such as the dilatation and attention module. For instance, SCINet (Liu et al., 2021) and Informer (Zhou et al., 2021) modified the sequence-to-sequence-based model to improve the performance for long sequences. However, most previous deep learning-based models are hard to interpret compared to statistical models. Thus, inspired by statistical models, N-BEATS (Oreshkin et al., 2020) designed an interpretable layer for time-series forecasting by encouraging the model to explicitly learn trend, seasonality, and residual components. This model shows superior performance on M4 competition datasets.

Distribution shift. Although there are various models for time-series forecasting, they often suffer from non-stationary time-series, where the data distribution changes over time. Domain adaptation (DA) (Tzeng et al., 2017; Ganin et al., 2016; Wang et al., 2018) and domain generalization (DG) (Wang et al., 2021; Li et al., 2018; Muandet et al., 2013) are common solutions for alleviating the distribution shift. A domain adaptation algorithm attempts to reduce the distribution gap between source and target domains. A domain generalization algorithm only relies on the source domain and hopes to generalize on the target domain. Both DA and DG have a common objective that bridges the gap between source and target distributions. However, in non-stationary time-series, defining a domain is not as straightforward since the data distribution shifts over time. Recently, Du et al. (Du et al., 2021) proposed Adaptive RNNs (AdaRNNs) to handle the distribution shift problems for non-stationary time-series data. It first characterizes the distribution information by splitting the training data into periods. Then, it matches distributions of the discovered periods to generalize the model. However, unlike AdaRNNs that are costly expensive, RevIN is simple yet effective and model-agnostic since the method can be adopted easily to any deep-learning model.

3 Proposed Method

In this section, we propose reversible instance normalization (RevIN) for alleviating the distribution shift problem in time-series, which causes the discrepancy among the input sequences of the training data, as well as the discrepancy between train and test data distributions. Section 3.1 describes RevIN in detail and Section 3.2 discusses how RevIN alleviates the distribution discrepancy in time-series throughout qualitative visualization.

3.1 Reversible Instance Normalization

We consider the multivariate time-series forecasting task in discrete time, with a sliding window size of T_x for input data. Here, N, K, T_x, and T_y indicate the number of the sequences, the number of variables, the input sequence length, and the prediction length. Let $\mathcal{X} = \{x^{(i)}\}_{i=1}^{N}$ and $\mathcal{Y} = \{y^{(i)}\}_{i=1}^{N}$ denote the input data and corresponding target, respectively. In RevIN, the input sequence length T_x and the prediction length T_y can be different since the observations are normalized and denormalized across the temporal dimension, as will be explained below. Given an input sequence $x^{(i)} \in \mathbb{R}^{K \times T_x}$, we aim to solve the time-series forecasting problem, which is to predict the subsequent values $y^{(i)} \in \mathbb{R}^{K \times T_y}$. Our proposed method, RevIN, symmetrically transforms the input data $x^{(i)}$ and the prediction output $\tilde{y}^{(i)}$ of the network, as illustrated in Fig. 2. First, we normalize the input data $x^{(i)}$ with the instance-specific mean and standard deviation, which is widely accepted as instance normalization (Ulyanov et al., 2016). The mean and standard deviation are computed for

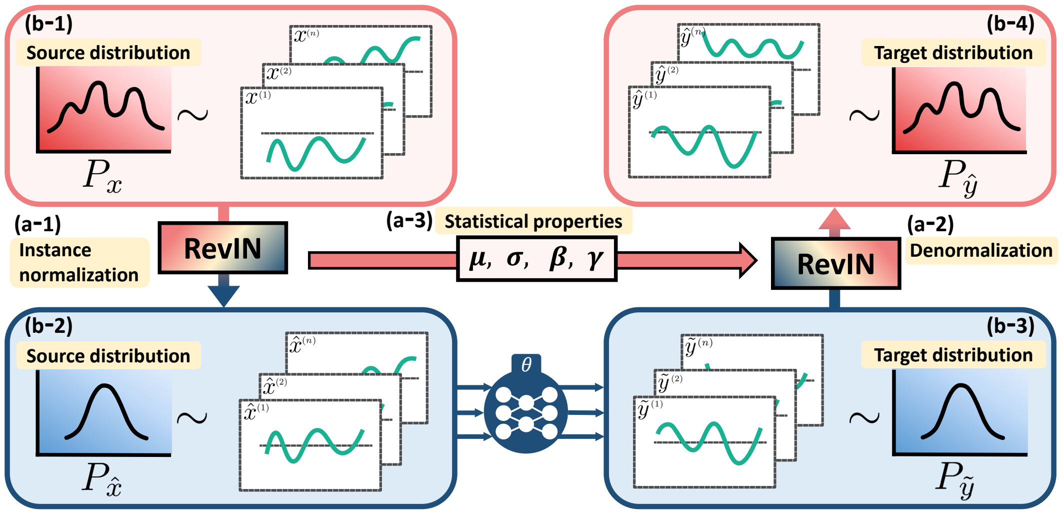

Figure 2: **Overview of the proposed method.** We present an example of a univariate case, where $x^{(i)} \in \mathbb{R}^{1\times T}$ (See Section 3.1). The input data $x^{(i)}$ is actually multivariate. In RevIN, (a-1) instance normalization and (a-2) denormalization are symmetrically structured, applied to the input and output of the model, respectively. (a-3) The statistical properties, mean μ, variance σ^2, and learnable affine parameters γ, β, are obtained from the input data for normalization, and then again used for denormalization. The instance normalization applied on the input data transforms (b-1) the original data distribution into (b-2) a mean-centered distribution, where the distribution discrepancy between instances is reduced. Given the transformed input $\hat{x}$ as the input, instead of x, the model predicts the subsequent values y, which follows (b-3) the normalized distribution similar to input $\hat{x}$. In order to reconstruct the distribution of the output back to (b-4) the original distribution, RevIN denormalizes the model output, reversing the normalization applied to the input data.

every instance $x_{k\cdot}^{(i)}$ of the input data (Fig. 2(a-3)) as

$$\mathbb{E}_t[x_{kt}^{(i)}] = \frac{1}{T_x}\sum_{j=1}^{T_x} x_{kj}^{(i)} \quad \text{and} \quad \mathrm{Var}[x_{kt}^{(i)}] = \frac{1}{T_x}\sum_{j=1}^{T_x}\left(x_{kj}^{(i)} - \mathbb{E}_t[x_{kt}^{(i)}]\right)^2. \tag{1}$$

Using these statistics, we normalize the input data $x^{(i)}$ as (Fig. 2(a-1))

$$\hat{x}_{kt}^{(i)} = \gamma_k\left(\frac{x_{kt}^{(i)} - \mathbb{E}_t[x_{kt}^{(i)}]}{\sqrt{\mathrm{Var}[x_{kt}^{(i)}] + \epsilon}}\right) + \beta_k, \tag{2}$$

where $\gamma, \beta \in \mathbb{R}^K$ are learnable affine parameter vectors. The normalized sequences have the same mean and variance, where the non-stationary information is reduced. As a result, the denormalization layer allows the model to accurately predict the local dynamics within the sequence while receiving inputs of consistent distributions in terms of the mean and variance.

The model then receives the transformed data $\hat{x}^{(i)}$ as input and forecasts their future values. However, the input data have different statistics from the original distribution, and only observing the normalized input $\hat{x}^{(i)}$, it is difficult to capture the original distribution of the input $x^{(i)}$. Thus, to make this easier for the model, we explicitly restore the non-stationary properties removed from the input data to the model output by reversing the normalization step (Fig. 2(a-3)). Such denormalization step can return the model output to the original time-series value as well (Ogasawara et al., 2010). Accordingly, we denormalize the model output $\tilde{y}^{(i)}$ by applying the reciprocal of the normalization in Eq. 2 (Fig. 2(a-2)) as

$$\hat{y}_{kt}^{(i)} = \sqrt{\mathrm{Var}[x_{kt}^{(i)}] + \epsilon}\cdot\left(\frac{\tilde{y}_{kt}^{(i)} - \beta}{\gamma}\right) + \mathbb{E}_t[x_{kt}^{(i)}], \tag{3}$$

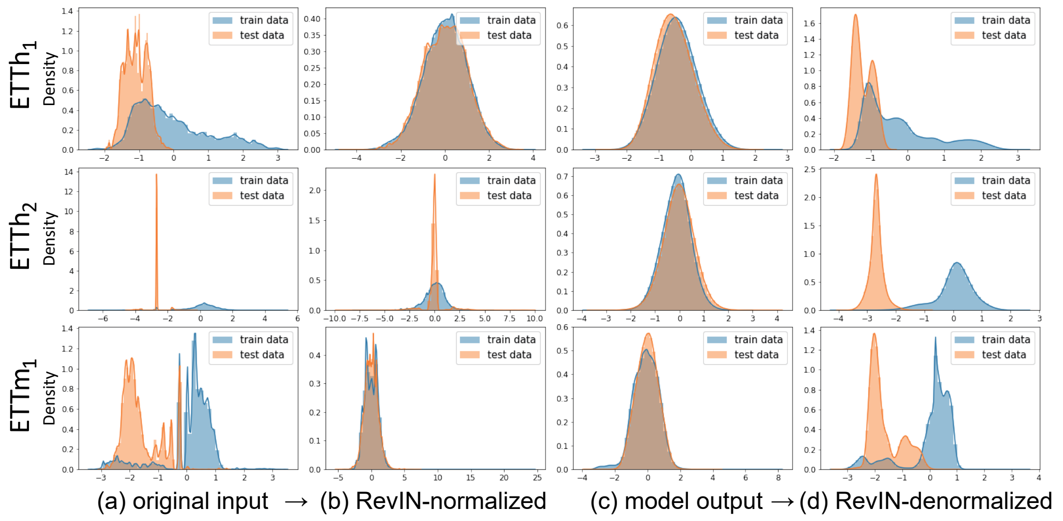

Figure 3: **Effect of RevIN on distribution discrepancy between train and test data.** From left to right columns, we compare the train and test data distributions on each step of the sequential process in RevIN: (a) the original input x, (b) the input normalized by RevIN, $\hat{x}$, (c) the model prediction output $\tilde{y}$, and (d) the output denormalized by RevIN, $\hat{y}$, which is also the final prediction. The analysis is conducted on ETT datasets using SCINet as the baseline.

where the same statistics used in the normalization step in Eq. (2) are used for the scaling and shifting. Now, $\hat{y}^{(i)}$ is the final prediction of the model instead of $\tilde{y}^{(i)}$.

Simply added to the input and output of a network, RevIN can be easily applied to any network with negligible cost while effectively improving time-series forecasting performance. Additionally, each step of RevIN, normalization and denormalization, can be applied to any activations of the intermediate layers in the network. However, our method is most effective when applied to the layers of symmetric encoder-decoder structure. In a typical time-series forecasting model, the boundary between the encoder and the decoder is often unclear. Therefore, we apply normalization and denormalization to the input and output of a forecasting network, which can be interpreted as an encoder-decoder structure, generating subsequent values given input data. Additional analysis on selecting layers to apply RevIN is provided in Section 4.2.3.

3.2 Effect of Reversible Instance Normalization on Distribution Shift

This section verifies that our method indeed alleviates the discrepancy in distributions during the forward pass of the network, and then reconstructs it in the network output layer through the denormalization step. We analyze the distributions of the train and test data before and after each step of our method, i.e., the normalization on input and denormalization on output, as shown in Fig. 3.

As a result, we observe that RevIN significantly reduces the discrepancy between the train and test data distributions in the normalization step (Fig. 3(b)) compared to the original data (Fig. 3(a)). In Fig. 3(a), the distributions of the train and test data hardly overlap, especially in the ETTh$_2$ dataset, which is caused by the distribution shift problem. Also, the distributions have multiple peaks, which can suggest that the sequences in the data have different distributions from each other.

However, in our method, the normalization step transforms each data distribution into mean-centered distributions. Thus, it can be said that the multimodal distribution in the input data ((Fig. 3(a)) is caused by the different distributions in sequences. Also, the train and test data distributions become similar to each other. It demonstrates that the normalization step of RevIN can alleviate the distribution shift problem, reducing the distribution discrepancy between train and test datasets. Moreover, the baseline model, where RevIN is applied, still maintains the similarly-shaped distributions in the prediction output (Fig. 3(c)). As expected, these are then returned back to the original distribution by the denormalization step of RevIN (Fig. 3(d)). Without denormalization, the model needs to reconstruct the values that follow the original distributions (Fig. 3(d)) only using the input with the transformed distributions (Fig. 3(b)). Additionally, we hypothesize that the distribution discrepancy is also reduced in the intermediate layers of the model when RevIN is applied at the input and output layers; RevIN does not need to be applied on the intermediate layers, which will be discussed in Section 4.2.3. As a result, this procedure of RevIN can be considered as making problems easier first and then turning them back rather than directly solving the challenging problem where the distribution shift problem exists.

4 EXPERIMENTS

This section describes the experimental setup and presents extensive experiments on RevIN.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate our methods on two large-scale real-world time-series datasets. **(i) Electricity transformer temperature (ETT)**[1] is electrical transformer temperature data collected from two different counties in China for two years. The data consists of seven features: oil temperature and six power load features. Following the same protocol of Informer (Zhou et al., 2021), we split the dataset into three datasets: ETTh_1, ETTh_2, and ETTm_1. ETTh_1 and ETTh_2 data are measured every hour at different locations. ETTm_1 data are measured every 15 minutes. For all three datasets, we use the first 12 months, the middle four months, and the last four months as training, validation, and test data, respectively. **(ii) Electricity Counting Load (ECL)**[2] data contains the electricity counting load collected from 321 clients, and the data measured from each client is used as a feature value. We convert the dataset to hourly basis following the prior work (Zhou et al., 2021). For the ECL dataset, we use 15, 3, and 4 months as train, validation, and test data, respectively.

Experimental details. We set prediction window lengths as 1 day (1d), 2d, 7d, 14d, 30d, and 40d for the datasets measured on an hourly basis, ETTh_1, ETTh_2, and ECL. For the ETTm_1 dataset, which is measured every 15 minutes, we chose 6 hours (6h), 12h, 3d, 7d, and 14d as prediction window lengths. We evaluate the time-series forecasting performance in terms of mean squared error (MSE) and mean absolute error (MAE). Following the same evaluation procedure of the previous study (Zhou et al., 2021), we compute the MSE and MAE on normalized data, since the values are not on the same scale. More details on experimental settings, including training details and hyperparameters, are provided in Appendix A.11.

Compared baselines. RevIN is a model-agnostic method that can be easily added to any network with further improvements. In this paper, we evaluate RevIN by adopting it on three time-series forecasting models: Informer (Zhou et al., 2021), N-BEATS (Oreshkin et al., 2020), and SCINet (Liu et al., 2021). All the models forecast the time-series sequence non-autoregressively. The exact training details of the models are reported in Appendix A.12. For all experiments, we compare RevIN and the baselines under the same hyperparameter settings, including the input and prediction lengths.

4.2 RESULTS AND ANALYSIS

This section presents quantitative analyses and qualitative visualizations of the experimental results of RevIN when it is adopted to the state-of-the-art time-series forecasting models.

4.2.1 EFFECTIVENESS OF REVERSIBLE INSTANCE NORMALIZATION ON VARIOUS TIME-SERIES FORECASTING MODELS

Table 1 demonstrates how much RevIN improves the forecasting accuracy of the baseline models. We compare the performance between RevIN and three baseline models, Informer, N-BEATS, and SCINet, on the four datasets with increasing prediction lengths. When adopted to various baselines, RevIN consistently outperforms their performance with a large margin, achieving the state-of-the-art performance on the four datasets. Moreover, the effectiveness of RevIN is more evident in the long sequence prediction, where it remarkably reduces the errors of the baselines. For example, when

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

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

Table 1: **Comparison of the forecasting errors of the baselines and RevIN.** The analysis is conducted on the four datasets by increasing the prediction sequence length from 24 to 960/1344. We report the average errors for five runs and the complete results are provided in Appendix A.14, including standard deviation and the original reported errors for the baselines.

Methods		Informer		**+ RevIN**		N-BEATS		**+ RevIN**		SCINet		**+ RevIN**	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	0.550	0.536	**0.504**	**0.472**	0.478	0.505	**0.330**	**0.373**	0.338	0.373	**0.308**	**0.347**
	48	0.772	0.668	**0.646**	**0.547**	0.536	0.542	**0.372**	**0.400**	0.436	0.459	**0.365**	**0.389**
	168	1.138	0.853	**0.655**	**0.561**	1.005	0.782	**0.466**	**0.452**	0.459	0.461	**0.406**	**0.416**
	336	1.278	0.909	**1.058**	**0.758**	0.932	0.743	**0.515**	**0.483**	0.527	0.513	**0.467**	**0.471**
	720	1.357	0.945	**0.926**	**0.717**	1.389	0.926	**0.576**	**0.534**	0.596	0.571	**0.507**	**0.505**
	960	1.470	0.990	**0.902**	**0.715**	1.383	0.932	**0.678**	**0.575**	0.604	0.574	**0.545**	**0.526**
$ETTh_2$	24	0.450	0.520	**0.238**	**0.325**	0.403	0.472	**0.192**	**0.276**	0.199	0.295	**0.180**	**0.263**
	48	2.171	1.200	**0.361**	**0.404**	1.330	0.918	**0.254**	**0.320**	0.350	0.422	**0.231**	**0.302**
	168	8.157	2.558	**0.859**	**0.649**	7.174	2.329	**0.410**	**0.418**	0.559	0.518	**0.337**	**0.378**
	336	4.746	1.844	**0.890**	**0.673**	4.859	1.863	**0.449**	**0.447**	0.664	0.583	**0.357**	**0.403**
	720	3.190	1.529	**0.576**	**0.546**	5.656	2.012	**0.496**	**0.482**	1.546	0.944	**0.411**	**0.445**
	960	2.972	1.441	**0.600**	**0.570**	6.408	2.077	**0.471**	**0.481**	1.862	1.066	**0.438**	**0.462**
$ETTm_1$	24	0.330	0.382	**0.309**	**0.352**	0.443	0.437	**0.403**	**0.392**	0.130	0.231	**0.106**	**0.196**
	48	0.499	0.486	**0.390**	**0.391**	0.453	0.472	**0.328**	**0.371**	0.155	0.262	**0.135**	**0.222**
	96	0.605	0.554	**0.405**	**0.411**	0.603	0.581	**0.379**	**0.406**	0.195	0.291	**0.162**	**0.247**
	288	0.906	0.738	**0.563**	**0.502**	0.849	0.702	**0.451**	**0.445**	0.361	0.419	**0.265**	**0.321**
	672	0.943	0.760	**0.663**	**0.550**	0.860	0.726	**0.555**	**0.511**	1.020	0.756	**0.357**	**0.380**
	1344	1.095	0.823	**0.824**	**0.632**	14.613	1.948	**0.631**	**0.556**	1.841	1.044	**0.412**	**0.422**
ECL	24	0.250	0.358	**0.148**	**0.257**	0.279	0.372	**0.176**	**0.285**	0.138	0.246	**0.112**	**0.207**
	48	0.300	0.386	**0.171**	**0.279**	0.309	0.388	**0.194**	**0.301**	0.163	0.265	**0.126**	**0.222**
	168	0.345	0.423	**0.261**	**0.354**	0.333	0.410	**0.218**	**0.320**	0.177	0.281	**0.153**	**0.249**
	336	0.429	0.473	**0.356**	**0.414**	0.326	0.406	**0.241**	**0.337**	0.202	0.308	**0.162**	**0.262**
	720	0.851	0.719	**0.834**	**0.700**	0.420	0.467	**0.303**	**0.383**	0.234	0.333	**0.183**	**0.281**
	960	0.930	0.750	**0.894**	**0.741**	0.399	0.455	**0.325**	**0.398**	0.235	0.330	**0.200**	**0.292**

Table 2: **Comparison on long sequence forecasting.** We analyze the forecasting error of the baseline models and RevIN by increasing the prediction length from 48 to 960 with the input sequence length fixed to 48. The experiment is conducted on $ETTh_1$, and the average errors for five runs are reported. The complete results, including standard deviation, are provided in Appendix A.14.

Prediction length	48		168		336		720		960	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Informer	0.687	0.628	0.982	0.795	1.212	0.893	1.157	0.863	1.203	0.888
+ RevIN	**0.540**	**0.481**	**0.680**	**0.574**	**0.939**	**0.696**	**1.021**	**0.752**	**1.061**	**0.775**
N-BEATS	0.512	0.523	0.804	0.690	1.001	0.773	1.022	0.765	0.901	0.728
+ RevIN	**0.365**	**0.389**	**0.454**	**0.438**	**0.526**	**0.477**	**0.568**	**0.514**	**0.638**	**0.544**
SCINet	0.376	0.396	0.600	0.556	0.841	0.695	0.875	0.721	0.900	0.737
+ RevIN	**0.349**	**0.370**	**0.445**	**0.426**	**0.509**	**0.461**	**0.533**	**0.494**	**0.557**	**0.510**

the prediction length increases from 24 to 960 on the $ETTh_2$ dataset, the forecasting error of N-BEATS significantly degrades from 0.403 to 6.408. In contrast, RevIN shows only a slight increase in the error, i.e., from 0.192 to 0.471; it makes the baseline model become robust to the prediction length. A similar tendency appears on the other prediction lengths, datasets, and baseline models as well. These results demonstrate that RevIN is a simple-yet-effective, model-agnostic method that can easily be added to any existing model while effectively improving the forecasting performance.

We further analyze the effect of RevIN on the long prediction sequence quantitatively as shown in Table 2. We increase the model prediction length from 48 to 960 when the prediction length is fixed to 48. As a result, RevIN consistently outperforms the baselines, showing robust performance against

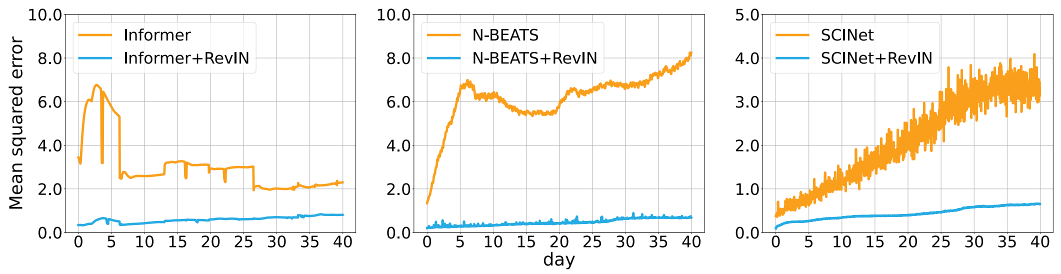

Figure 4: **Forecasting error for each time step.** We compare the three baseline models and RevIN on the ETTh$_2$ dataset when the prediction length is 960.

Table 3: **Comparison results with classical and state-of-the-art normalization methods.** The forecasting error is measured on the four datasets, using N-BEATS as the baseline for all experiments. Here, every normalization method is applied to the input data. DAIN, deep adaptive input normalization (Passalis et al., 2019); **RevBN**, the reversible batch normalization, i.e., the modified version of RevIN, where the input normalization method is replaced with batch normalization.

Dataset	ETTh$_1$		ETTh$_2$		ETTm$_1$		ECL	
Prediction length	168	960	168	960	96	1344	168	960
Min-max norm	1.074	1.224	2.987	3.308	1.035	1.320	0.374	0.387
z-score norm	0.953	1.043	3.329	3.087	1.016	1.274	0.335	0.377
Layer norm	0.871	1.303	4.092	5.822	0.502	2.488	0.343	0.379
DAIN	0.996	1.032	1.982	2.802	0.672	1.348	0.347	0.381
Batch norm	0.851	1.691	6.206	7.755	0.505	1.147	0.320	0.422
RevBN	0.717	0.779	0.729	2.148	0.601	0.991	0.327	0.414
Instance norm	0.946	1.090	3.240	3.145	1.021	1.329	0.333	0.386
RevIN (ours)	**0.515**	**0.697**	**0.419**	**0.465**	**0.388**	**0.602**	**0.220**	**0.329**

the prediction length. The difference in the forecasting error between SCINet and RevIN is relatively small when the prediction length is short (e.g., 48), but RevIN remarkably surpasses SCINet with a significant margin when the prediction length becomes long (e.g., 336, 720, and 960). These results show that our method makes the baseline models become robust to the prediction length.

To further study why RevIN shows stable performance even for the long prediction length, we analyze the forecasting error for each time step. The error at the t-th time step is computed as $MSE_t = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}(\hat{y}_{kt}^{(i)} - y_{kt}^{(i)})^2$. Fig. 4 provides the results on the ETTh$_2$ dataset comparing the baselines and RevIN. In the cases of N-BEATS and SCINet, the error extremely increases as predicting the distant future. RevIN alleviates such a significant increase in error, outperforming the baselines with a large margin. Informer also shows unstable performance for the different time steps. The error is large in the early time step and becomes relatively constant for the future compared to the other baseline models. RevIN allows the models to have consistent small errors at every time step. The results demonstrate the effectiveness of RevIN on forecasting long sequences.

4.2.2 Comparison with Other Normalization Methods

We compare RevIN with classical and state-of-the-art normalization methods, including min-max normalization, z-score normalization, layer normalization (Ba et al., 2016), batch normalization (Ioffe & Szegedy, 2015), instance normalization (Ulyanov et al., 2016), and deep adaptive input normalization (DAIN) (Passalis et al., 2019) as shown in Table 3. The normalization statistics for min-max and z-score normalization methods are computed for each input instance, not for the entire data. Additionally, we use batch normalization as the replacement of the input normalization method in RevIN, named reversible batch normalization (RevBN). Layer normalization cannot be used similarly since its standardization operation is not reversible when the input and prediction lengths are

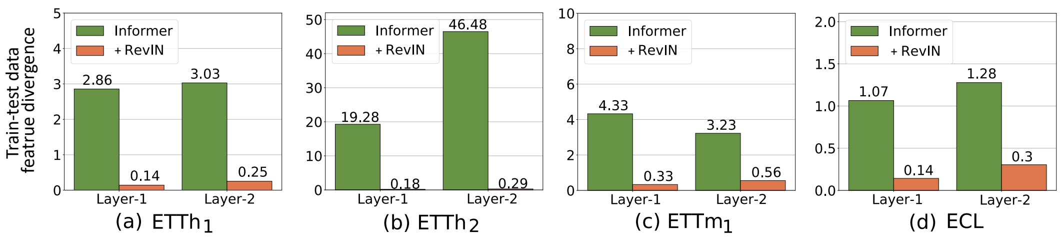

Figure 5: **Feature divergence between the train and test data at the intermediate layers of the model** The feature divergences are computed on the four datasets using the features obtained from the first (Layer-1) and the second (Layer-2) layers of the encoder in Informer.

different as our experimental settings. Here, we provide the experimental results on the four datasets, using N-BEATS as the baseline. Complete results including different prediction lengths are provided in Table 10 in Appendix A.7. RevIN shows outstanding performance compared to other normalization methods, especially on ETTh$_2$ and ETTm$_1$ datasets. Moreover, RevBN also improves the forecasting performance of batch normalization. Especially, its error is largely decreased when predicting the long sequences, e.g., 960 and 1344. It implies that the denormalization step of RevIN is essential for improving long sequence forecasting as a normalization layer. However, batch normalization applies an identical normalization to all the input sequences using the statistics obtained from the entire training data, so it can not reduce the discrepancy between the distributions of the train and test data. As a result, RevIN, which uses instance-wise normalization, surpasses RevBN with a significant margin, demonstrating that its successful reduction of the distribution discrepancy indeed effectively improves the performance. Moreover, RevIN not only improves the performance of the baselines but also has the advantage of being lightweight compared to the other method, say, DAIN that requires at least $3K^2$ additional parameters, when K is the number of variables; RevIN only requires $2K$ additional parameters, which is almost negligible costs for the baseline models.

4.2.3 Analysis on Distribution Shift In the Intermediate Layers

We can assume that RevIN can reduce the distribution shift at an intermediate feature level as well. Thus, we analyze the feature divergence between the train and test data for Informer and RevIN on the four datasets. Informer consists of two layers for the encoder and one layer for the decoder. Thus, we use the output of the first (Layer-1) and the second (Layer-2) layers of the encoder for the analysis to obtain the intermediate layer features. Following previous work (Pan et al., 2018), we compute the average feature divergence using symmetric KL divergence. The details on the calculation of the feature divergence are provided in Appendix A.10. As shown in Fig. 5, the feature divergence between the train and test data is significantly reduced as RevIN is added to the baseline model. It implies that the normalization applied to the input can resolve the distribution shift in every layer of a model. Therefore, our proposed method successfully alleviates the distribution shift problem even only added to the input and output layers in a model. This allows our method to be adopted to any network without modification on the architectures, as a model-agnostic method, even effectively improving the model performance. Note that our method can be added to the intermediate layers as well, still greatly improving the model performance, as shown in Appendix A.4.

5 Conclusion

We propose a simple yet effective normalization method tailored to time-series forecasting. RevIN consists of symmetrically structured normalization and denormalization layers, where the shifting and scaling applied on the input data is exactly reversed back on the output prediction, effectively reducing the discrepancy between train and test data distributions. Easily applied to input and output layers with a negligible number of additional parameters, RevIN significantly improves the forecasting performance of a model without any modification in the architecture of the model. By adding RevIN, we achieve the state-of-the-art performance on four large-scale real-world datasets with a significant margin. The extensive experiments demonstrate the effectiveness of RevIN for accurate time-series forecasting against the distribution shift problem.

6 Reproducibility Statement

To ensure reproducibility, we provide the source code of our method as the supplementary material, including the pretrained model weights. In the main manuscript, Section 4.1 describes how we conduct data preprocessing on the dataset used in the experiments. In the appendix, Section A.11 explains the experimental details including the seed for the experiments. Also, we provide a detailed explanation of hyperparameter settings with the reproduction details of the baseline models in Section A.12 since our method is an add-on module with no additional hyperparameters.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Stefan Bauer, Bernhard Schölkopf, and Jonas Peters. The arrow of time in multivariate time series. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 2043–2051. PMLR, 2016.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. Adarnn: Adaptive learning and forecasting of time series. *Proc. the International Conference on Information and Knowledge Management (CIKM)*, 2021.

Laura Frías-Paredes, Fermín Mallor, Martín Gastón-Romeo, and Teresa León. Assessing energy forecasting inaccuracy by simultaneously considering temporal and absolute errors. *Energy Conversion and Management*, 142:533–546, 2017.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research (JMLR)*, 17(1):2096–2030, 2016.

Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. the International Conference on Machine Learning (ICML)*, 2015.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, 2017.

Vincent Le Guen and Nicolas Thome. Shape and time distortion loss for training deep time series forecasting models. In *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, volume 4191, 2019.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2018.

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

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *Proc. the International Conference on Machine Learning (ICML)*. PMLR, 2013.

Eduardo Ogasawara, Leonardo C Martinez, Daniel De Oliveira, Geraldo Zimbrão, Gisele L Pappa, and Marta Mattoso. Adaptive normalization: A novel data normalization approach for non-stationary time series. In *The International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2010.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *Proc. the International Conference on Learning Representations (ICLR)*, 2020.

Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018.

Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Deep adaptive input normalization for time series forecasting. *IEEE transactions on neural networks and learning systems*, 31(9):3760–3765, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary 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. In *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proc. the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pp. 5956–5963, 2020.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. Visual domain adaptation with manifold embedded distribution alignment. In *Proc. the ACM international conference on Multimedia*, pp. 402–410, 2018.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*, 2021.

Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proc. the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

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 *Proc. the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

A Appendix

The supplementary material presents additional information, visualizations, and experimental results that support the main manuscript. Section A.1 shows experimental results on three real-world datasets along with qualitative analysis on the Nasdaq dataset to verify the effectiveness of our method on obvious non-stationary time-series. Section A.2 addresses the potential of RevIN on a cross-domain time-series forecasting task. Section A.3 and Section A.4 conduct the hyperparameter sensitivity analysis and the ablation study on the proposed method, respectively. Section A.5 evaluates our method using the complementary metrics that measure the similarity between the two sequences. Section A.6 and Section A.7 show the comparison results of our method and existing normalization methods. Section A.8, Section A.9, and Section A.10 provide the algorithm of RevIN, the theoretical justification of RevIN, and the calculation details of the feature divergence used in Section 4.2.3, respectively. Section A.11 and Section A.12 describe additional implementation details and the reproduction details for baselines where RevIN is applied, respectively. Section A.13 illustrates additional quantitative results of RevIN and the baseline models. Lastly, Section A.14 provides extra quantitative results, including the standard deviation values for five experiments, which can not be located in the manuscript due to the lack of space.

A.1 Experimental Results on Additional Real-world Time-series Datasets

Table 4: **Forecasting performance on the Air quality, Nasdaq, and M4 competition datasets. The results on the M4 dataset (*) are multiplied by ten for readability. The average value and the standard deviation for five runs are reported.**

Methods		Informer		**+ RevIN**		N-BEATS		**+ RevIN**		SCINet		**+ RevIN**	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Air quality	24	0.802 ± 0.178	0.671 ± 0.084	**0.585 ± 0.033**	**0.539 ± 0.023**	0.698 ± 0.064	0.626 ± 0.029	**0.527 ± 0.005**	**0.498 ± 0.003**	0.512 ± 0.029	0.514 ± 0.019	**0.490 ± 0.006**	**0.474 ± 0.004**
	48	0.966 ± 0.054	0.761 ± 0.023	**0.859 ± 0.086**	**0.668 ± 0.041**	0.955 ± 0.106	0.740 ± 0.035	**0.705 ± 0.019**	**0.600 ± 0.009**	0.712 ± 0.091	0.627 ± 0.047	**0.659 ± 0.013**	**0.566 ± 0.007**
	168	1.328 ± 0.107	0.923 ± 0.040	**1.036 ± 0.056**	**0.761 ± 0.020**	1.079 ± 0.108	0.818 ± 0.046	**0.789 ± 0.008**	**0.660 ± 0.005**	0.957 ± 0.067	0.737 ± 0.031	**0.794 ± 0.025**	**0.645 ± 0.014**
	336	1.278 ± 0.074	0.901 ± 0.032	**1.145 ± 0.032**	**0.801 ± 0.010**	1.105 ± 0.052	0.835 ± 0.021	**0.860 ± 0.017**	**0.685 ± 0.006**	0.989 ± 0.111	0.760 ± 0.046	**0.854 ± 0.029**	**0.676 ± 0.010**
	720	2.028 ± 0.216	1.104 ± 0.061	**1.161 ± 0.028**	**0.810 ± 0.009**	1.538 ± 0.419	0.968 ± 0.110	**0.842 ± 0.015**	**0.686 ± 0.008**	1.228 ± 0.048	0.858 ± 0.021	**0.839 ± 0.024**	**0.680 ± 0.013**
Nasdaq	30	5.318 ± 0.052	1.093 ± 0.017	**1.273 ± 0.078**	**0.630 ± 0.009**	5.500 ± 0.647	1.254 ± 0.086	**1.023 ± 0.034**	**0.577 ± 0.007**	1.742 ± 0.111	0.739 ± 0.028	**0.985 ± 0.018**	**0.564 ± 0.005**
	60	5.525 ± 0.022	1.098 ± 0.016	**1.573 ± 0.098**	**0.666 ± 0.011**	5.226 ± 0.424	1.236 ± 0.032	**1.207 ± 0.044**	**0.617 ± 0.009**	2.304 ± 0.062	0.790 ± 0.010	**1.161 ± 0.021**	**0.601 ± 0.003**
	120	5.793 ± 0.140	1.090 ± 0.012	**2.648 ± 0.186**	**0.762 ± 0.016**	6.023 ± 0.382	1.197 ± 0.034	**1.959 ± 0.062**	**0.714 ± 0.006**	3.227 ± 0.236	0.853 ± 0.007	**1.869 ± 0.037**	**0.697 ± 0.003**
M4*	average	0.099 ± 0.002	0.258 ± 0.020	**0.008 ± 0.005**	**0.074 ± 0.005**	2.241 ± 0.037	2.065 ± 0.029	**2.082 ± 0.014**	**1.974 ± 0.006**	2.180 ± 1.943	1.943 ± 0.011	**2.079 ± 0.011**	**1.892 ± 0.004**

We evaluate our proposed method on four large-scale real-world time-series datasets, the ETTh$_1$, ETTh$_2$, ETTm$_1$, and ECL datasets. Additionally, this section provides experimental results on three real-world datasets: the two real-world datasets taken from the UCI repository, the air quality dataset and the Nasdaq dataset, and the M4 competition dataset (Makridakis et al., 2020). **Air quality** [3] data is hourly data with 13 variables of length 9537. We set the prediction length as 24, 48, 168, 336, 720 and the corresponding input length as 48, 96, 168, 360 so that their ratio becomes 2x, 2x, 1x, 0.5x, 0.5x. **Nasdaq**[4] dataset consists of 82 features, including the closing measured since 2010. It is measured daily and has 1984 data samples for each variable in total. We fix the input sequence length to 60 and prolong the prediction length as 30, 60, 120. **M4**[5] competition dataset consists of six different hourly, daily, weekly, monthly, quarterly, and yearly sets, containing 100,000 test data. We follow the original experimental protocol of the M4 competition (Makridakis et al., 2020). For evaluation, we measure the micro-averaged mean absolute error and mean squared error on the M4 competition dataset: we first compute the metric independently for each set, i.e., hourly, daily, weekly, monthly, quarterly, and yearly sets, and then calculate the weighted average of the metrics using the contributions of each set as weights.

[3]https://archive.ics.uci.edu/ml/datasets/Air+Quality

[4]https://archive.ics.uci.edu/ml/datasets/CNNpred%3A+CNN-based+stock+market+prediction+using+a+diverse+set+of+variables

[5]https://mofc.unic.ac.cy/m4/

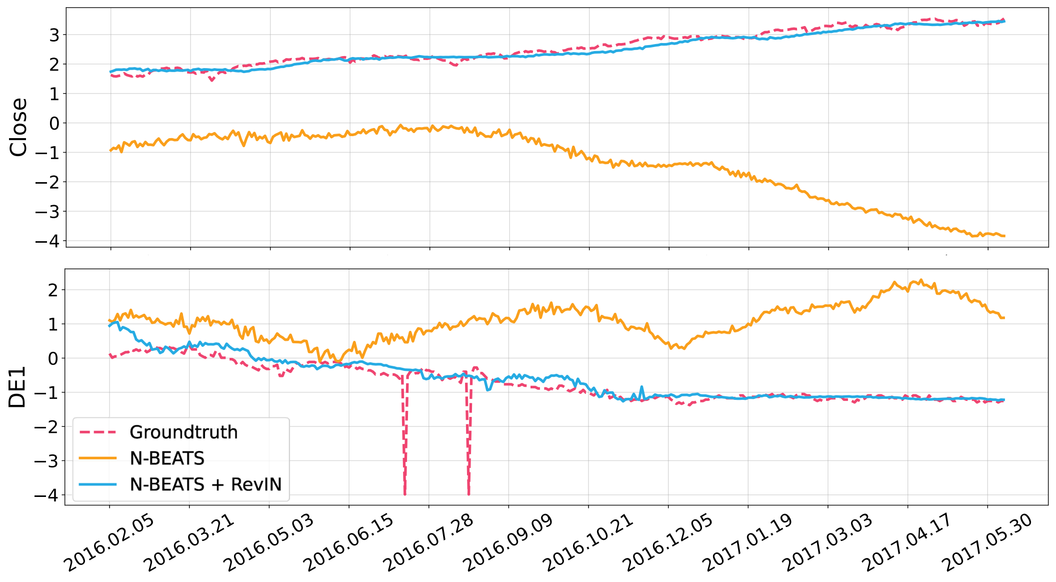

Figure 6: **Prediction results on the Nasdaq dataset.** The results on the two variables in the data, Close and DE1, are shown. The prediction length is 60 days, and the seventh value is illustrated to show the results of the entire test set. We evaluate the proposed method in comparison with the N-BEATS baseline model.

As shown in Table 4, the model adopting RevIN significantly improves the forecasting performance of the baseline on all three datasets. Notably, RevIN shows outstanding performance on the Nasdaq dataset, reducing the prediction errors by more than half compared to the baseline.

To verify the effectiveness of our method on obvious non-stationary time-series, we conduct qualitative analysis on the Nasdaq dataset. As shown in Fig. 6, the Nasdaq index (the variable 'Close') has steadily increased since 2002. Accordingly, when data are divided into the train and test data based on a specific point in time, the test data values tend to be higher than the train data values. In other words, the data severely suffers from the distribution shift problem, where the train and test data show a discrepancy in their distribution. As a result, even existing SOTA networks cannot predict the future values appropriately, as shown in Fig. 6. The baseline fails to keep up with the trend of data, whose mean value continues to increase, and thus the prediction results are shifted. However, RevIN mitigates such distribution discrepancy and remarkably increases the prediction performance of the baseline SOTA model. Similarly, RevIN shows superior performance on the decreasing data ('DE1' in Fig. 6), accurately predicting the changing mean of the data.

A.2 Cross-Domain Evaluation

Table 5: **Cross-domain evaluation results of RevIN.** The mean squared error for cross-domain evaluation is reported for ETT datasets, which have identical features but have different distributions.

Train		$ETTh_1$				$ETTh_2$				$ETTm_1$			
Test		$ETTh_2$		$ETTm_1$		$ETTh_1$		$ETTm_1$		$ETTh_1$		$ETTh_2$	
Prediction length		336	960	336	960	336	960	336	960	288	1344	288	1344
SCINet	MSE	0.471	0.741	0.471	0.765	0.614	0.671	0.608	1.668	0.659	0.654	0.518	1.813
	MAE	0.450	0.640	0.427	0.636	0.507	0.607	0.487	1.004	0.562	0.608	0.509	1.027
+ RevIN	MSE	**0.350**	**0.419**	**0.346**	**0.452**	**0.501**	**0.586**	**0.321**	**0.441**	**0.478**	**0.542**	**0.387**	**0.506**
	MAE	**0.383**	**0.451**	**0.331**	**0.449**	**0.449**	**0.542**	**0.331**	**0.445**	**0.477**	**0.531**	**0.417**	**0.503**

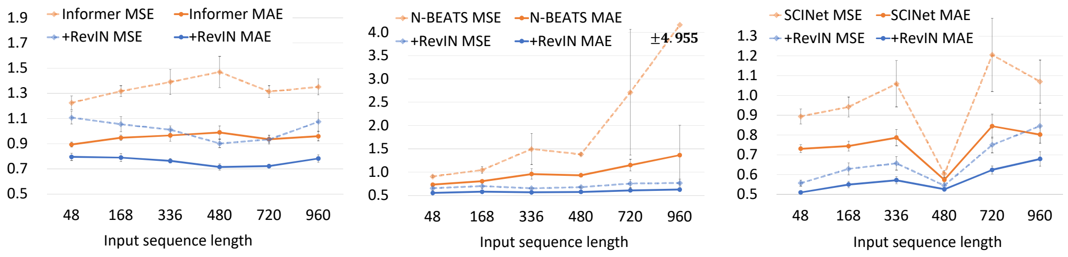

Figure 7: **Impacts of the input length on RevIN in comparison with the baseline models.** The average errors for five runs with the standard deviation values are reported. We prolong the input length from 48 (two days) to 960 (40 days) with the prediction length fixed to 960 (40 days) on the ETTh$_1$ dataset. We measure the prediction error of RevIN adopted to the three SOTA models on the MAE and MSE. In N-BEATS, the standard deviation value of the error is too large to visualize when the prediction length is 960; thus, we write the value as "± 4.955" instead.

We analyze the cross-domain evaluation results to analyze further the capability of RevIN to alleviate cross-domain distribution discrepancy. As RevIN reduces the discrepancy in the train and test distributions, it can also reduce the discrepancy in the cross-domain distributions if the input sequences from each dataset have similar local dynamics. We use ETT datasets for the evaluation since they have the same features. However, their distributions can show a significant discrepancy because ETTh$_1$ and ETTh$_2$ are measured from different locations, and ETTh and ETTm$_1$ datasets have different measurement intervals. We train the baseline and RevIN using SCINet as a baseline on ETT datasets and evaluate the trained model on the other datasets. We report the average MSE of five runs in Table 5 and variance in Table 11. Despite the difference between distributions of the datasets, RevIN shows superior performance in cross-domain evaluations. Especially, RevIN outperforms a SCINet with a large margin when the model needs to reduce the discrepancy in ETTh$_2$ and ETTm$_1$ datasets. The results demonstrate that RevIN successfully reduces the discrepancy in the input sequence distributions, leading to better generalization performance.

A.3 Hyperparameter Sensitivity Analysis

We analyze the hyperparameter sensitivity of the proposed method RevIN in comparison with the baseline SOTA models. Input sequence length can be a crucial hyperparameter to RevIN since it computes the mean and variance across each input sequence and thus can determine the prediction accuracy and stability of the training process. In Fig. 7, we increase the model input sequence length from 48 (two days) to 960 (40 days) to see its impact on RevIN when the prediction length is fixed to 960 (40 days). First, the results demonstrate that the models adopting RevIN consistently outperform the baselines for various input sequence lengths in terms of the MAE and MAE. More importantly, RevIN makes the baseline models more robust to the input length. In other words, RevIN shows stable performance for various input lengths in contrast to the baseline models, which shows a high variance in their performance according to the input length. Notably, in N-BEATS, the average error as well as the standard deviation of the error significantly increase as prolonging the input length. This is because the trend of data is expressed as a linear function in N-BEATS. As the input length becomes prolonged, there is a chance that the variance (or non-stationarity) in the time-series values becomes higher, decreasing the accuracy of the linear trend to fit the data unless the data is monotonically increasing or decreasing. Also, the mispredicted trends linearly increase the error in the future values. However, when N-BEATS adopts the RevIN layer, it removes the non-stationary statistics from the input, and thus, the model shows robust performance against the input length. Removing the variability, i.e., normalizing its mean and standard deviation, before feeding it to the model and returning it to the output would make the model learning stable.

A.4 Ablation Study

We ablate the affine transformation from RevIN to see its impact on the performance. We conduct the experiments on the ETTh$_1$, ETTh$_2$, ETTm$_1$, ECL, Nasdaq, and air quality datasets using N-BEATS

Table 6: **Ablation study results.** We ablate the affine transformation (affine.) from RevIN and evaluate the performance on the six datasets. N-BEATS is used as the baseline for all experiments. We report the average and the standard deviation values for the five runs.

Methods		+ **RevIN** w/o affine.		+ **RevIN** w/ affine. (ours)	
Metric		MSE	MAE	MSE	MAE
$ETTh_1$	48	0.370 ± 0.006	0.393 ± 0.003	**0.363 ± 0.005**	**0.389 ± 0.003**
	960	0.675 ± 0.038	0.576 ± 0.014	**0.638 ± 0.035**	**0.559 ± 0.017**
$ETTh_2$	48	0.257 ± 0.003	0.322 ± 0.002	**0.255 ± 0.008**	**0.321 ± 0.005**
	960	0.483 ± 0.012	0.487 ± 0.007	**0.471 ± 0.015**	**0.481 ± 0.008**
$ETTm_1$	96	0.384 ± 0.013	0.408 ± 0.009	**0.378 ± 0.011**	**0.406 ± 0.007**
	1344	0.664 ± 0.085	0.567 ± 0.039	**0.631 ± 0.061**	**0.556 ± 0.020**
ECL	48	0.197 ± 0.002	0.302 ± 0.002	**0.195 ± 0.002**	**0.301 ± 0.001**
	960	0.347 ± 0.034	0.415 ± 0.026	**0.325 ± 0.019**	**0.398 ± 0.015**
Air quality	48	0.707 ± 0.009	0.601 ± 0.002	**0.705 ± 0.019**	**0.600 ± 0.009**
	720	0.852 ± 0.025	0.689 ± 0.013	**0.842 ± 0.015**	**0.686 ± 0.008**
Nasdaq	30	0.983 ± 0.017	0.565 ± 0.005	**0.981 ± 0.017**	**0.564 ± 0.005**
	60	1.163 ± 0.010	0.610 ± 0.002	**1.155 ± 0.020**	**0.608 ± 0.005**

as the baseline. The results in Table 6 show that the affine transformation contributes to performance improvement consistently on a variety of datasets.

As mentioned earlier, a model can add RevIN in an intermediate layer, even to several layers. While existing approaches are a preprocessing-and-postprocessing method applied outside of the main prediction model, RevIN is an end-to-end trainable layer that can be added to any layer in the model as batch normalization and instance normalization, which are the recently proposed deep learning-based normalization layers. Thus, we verify that adopting RevIN to the intermediate layers instead of the input and output layers can improve the forecasting performance as well. We add RevIN at the first stack of N-BEATS and SCINet and evaluate their performance on the six datasets.

The results in Table 7 demonstrate that even when added to the intermediate layers, RevIN improves the performance of the baselines as a learnable normalization layer. We mainly focus on adding RevIN to the input and output of a model since it shows robust performance on average. Nevertheless, the model adopting RevIN in the intermediate layers also shows the best performance frequently, consistently outperforming the baseline without RevIN. This performance is even better than the dynamic normalization methods, LSTNet* and ES-RNN*, when comparing them where N-BEATS is used as the baseline (Table 9). For example, when the prediction length is 960, the mean squared errors of LSTNet*, ES-RNN*, RevIN (inter.), and RevIN (i/o) are 5.627, 1.338, 0.523, and 0.471, on average.

A.5 Evaluation on Similarity Metrics

We additionally evaluate our method using the metrics that measure similarity between the two sequences, dynamic time warping (DTW) and temporal distortion index (TDI) (Le Guen & Thome, 2019; Frías-Paredes et al., 2017). As shown in Table 8, the results show that our approach significantly improves the baseline models across all datasets in terms of the DTW and TDI. Notably,

Table 7: **Effectiveness of RevIN when added to the intermediate layers in the model.** We add RevIN at the first stack of N-BEATS and SCINet and evaluate their performance on the six datasets. We report the average value and standard deviation of five experiments. RevIN (inter.) denotes that RevIN is added to the intermediate layers of the baseline network. RevIN (i/o) denotes that RevIN is added to the input and output layer of the baseline network.

Methods		N-BEATS		+ **RevIN** (inter.)		+ **RevIN** (i/o)		SCINet		+ **RevIN** (inter.)		+ **RevIN** (i/o)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	0.478 ± 0.022	0.505 ± 0.012	0.347 ± 0.006	0.389 ± 0.004	**0.330** ± **0.006**	**0.373** ± **0.004**	0.338 ± 0.012	0.373 ± 0.009	**0.306** ± **0.004**	0.347 ± 0.004	0.308 ± 0.003	**0.347** ± **0.002**
	48	0.536 ± 0.060	0.542 ± 0.041	0.375 ± 0.008	0.407 ± 0.005	**0.372** ± **0.001**	**0.400** ± **0.002**	0.436 ± 0.025	0.459 ± 0.021	**0.363** ± **0.004**	0.394 ± 0.004	0.365 ± 0.005	**0.389** ± **0.003**
	168	1.005 ± 0.146	0.782 ± 0.064	0.495 ± 0.086	0.481 ± 0.062	**0.466** ± **0.030**	**0.452** ± **0.014**	0.459 ± 0.015	0.461 ± 0.013	0.415 ± 0.001	0.424 ± 0.002	**0.406** ± **0.003**	**0.416** ± **0.003**
	336	0.932 ± 0.079	0.743 ± 0.042	0.538 ± 0.043	0.508 ± 0.027	**0.515** ± **0.013**	**0.483** ± **0.008**	0.527 ± 0.010	0.513 ± 0.006	0.552 ± 0.002	0.516 ± 0.001	**0.467** ± **0.005**	**0.471** ± **0.003**
	720	1.389 ± 0.230	0.926 ± 0.066	0.608 ± 0.016	0.572 ± 0.009	**0.576** ± **0.035**	**0.534** ± **0.018**	0.596 ± 0.015	0.571 ± 0.013	0.560 ± 0.007	0.550 ± 0.005	**0.507** ± **0.006**	**0.505** ± **0.004**
	960	1.383 ± 0.380	0.932 ± 0.120	**0.664** ± **0.033**	0.604 ± 0.015	0.678 ± 0.019	**0.575** ± **0.009**	0.604 ± 0.017	0.574 ± 0.014	0.619 ± 0.005	0.582 ± 0.002	**0.545** ± **0.010**	**0.526** ± **0.005**
$ETTh_2$	24	0.403 ± 0.185	0.472 ± 0.101	0.199 ± 0.002	0.291 ± 0.002	**0.192** ± **0.003**	**0.276** ± **0.002**	0.199 ± 0.026	0.295 ± 0.027	0.186 ± 0.003	0.272 ± 0.001	**0.180** ± **0.004**	**0.263** ± **0.002**
	48	1.330 ± 0.240	0.918 ± 0.073	0.263 ± 0.007	0.335 ± 0.005	**0.254** ± **0.011**	**0.320** ± **0.008**	0.350 ± 0.025	0.422 ± 0.027	0.313 ± 0.045	0.373 ± 0.032	**0.231** ± **0.006**	**0.302** ± **0.006**
	168	7.174 ± 0.449	2.329 ± 0.049	0.425 ± 0.015	0.434 ± 0.010	**0.410** ± **0.010**	**0.418** ± **0.005**	0.559 ± 0.044	0.518 ± 0.025	0.338 ± 0.003	0.380 ± 0.001	**0.337** ± **0.007**	**0.378** ± **0.003**
	336	4.859 ± 0.268	1.863 ± 0.043	**0.446** ± **0.007**	0.456 ± 0.005	0.449 ± 0.011	**0.447** ± **0.006**	0.664 ± 0.073	0.583 ± 0.030	0.422 ± 0.001	0.443 ± 0.001	**0.357** ± **0.003**	**0.403** ± **0.002**
	720	5.656 ± 1.053	2.012 ± 0.186	0.505 ± 0.022	0.501 ± 0.013	**0.496** ± **0.008**	**0.482** ± **0.002**	1.546 ± 0.378	0.944 ± 0.141	0.634 ± 0.010	0.564 ± 0.005	**0.411** ± **0.003**	**0.445** ± **0.002**
	960	6.408 ± 2.039	2.077 ± 0.242	0.523 ± 0.040	0.522 ± 0.025	**0.471** ± **0.015**	**0.481** ± **0.008**	1.862 ± 0.153	1.066 ± 0.055	0.734 ± 0.014	0.603 ± 0.005	**0.438** ± **0.007**	**0.462** ± **0.004**
$ETTm_1$	24	0.443 ± 0.043	0.437 ± 0.035	**0.387** ± **0.018**	**0.391** ± **0.012**	0.403 ± 0.006	0.392 ± 0.005	0.130 ± 0.003	0.231 ± 0.003	0.108 ± 0.002	0.203 ± 0.004	**0.106** ± **0.002**	**0.196** ± **0.001**
	48	0.453 ± 0.034	0.472 ± 0.018	0.341 ± 0.008	0.388 ± 0.007	**0.328** ± **0.010**	**0.371** ± **0.007**	0.155 ± 0.004	0.262 ± 0.004	0.142 ± 0.011	0.241 ± 0.013	**0.135** ± **0.003**	**0.222** ± **0.002**
	96	0.603 ± 0.051	0.581 ± 0.027	0.401 ± 0.007	0.428 ± 0.004	**0.379** ± **0.011**	**0.406** ± **0.007**	0.195 ± 0.012	0.291 ± 0.013	0.192 ± 0.016	0.285 ± 0.017	**0.162** ± **0.001**	**0.247** ± **0.001**
	288	0.849 ± 0.095	0.702 ± 0.051	0.502 ± 0.032	0.483 ± 0.018	**0.451** ± **0.016**	**0.445** ± **0.008**	0.361 ± 0.008	0.419 ± 0.004	**0.264** ± **0.002**	0.323 ± 0.001	0.265 ± 0.003	**0.321** ± **0.002**
	672	0.860 ± 0.057	0.726 ± 0.026	**0.553** ± **0.020**	0.512 ± 0.009	0.555 ± 0.011	**0.511** ± **0.008**	1.020 ± 0.040	0.756 ± 0.025	0.663 ± 0.081	0.583 ± 0.033	**0.357** ± **0.004**	**0.380** ± **0.002**
	1344	14.613 ± 26.108	1.948 ± 1.655	0.722 ± 0.064	0.594 ± 0.027	**0.631** ± **0.061**	**0.556** ± **0.020**	1.841 ± 0.242	1.044 ± 0.100	0.989 ± 0.211	0.717 ± 0.081	**0.412** ± **0.008**	**0.422** ± **0.003**
ECL	24	0.279 ± 0.007	0.372 ± 0.003	0.182 ± 0.001	0.300 ± 0.001	**0.176** ± **0.002**	**0.285** ± **0.001**	0.138 ± 0.004	0.246 ± 0.005	**0.111** ± **0.000**	**0.207** ± **0.001**	0.112 ± 0.001	**0.207** ± **0.001**
	48	0.309 ± 0.007	0.388 ± 0.004	0.207 ± 0.003	0.318 ± 0.002	**0.194** ± **0.001**	**0.301** ± **0.001**	0.163 ± 0.007	0.265 ± 0.007	**0.124** ± **0.001**	**0.221** ± **0.001**	0.126 ± 0.001	0.222 ± 0.001
	168	0.333 ± 0.016	0.410 ± 0.012	0.237 ± 0.007	0.340 ± 0.004	**0.218** ± **0.002**	**0.320** ± **0.001**	0.177 ± 0.003	0.281 ± 0.005	0.154 ± 0.002	**0.248** ± **0.001**	**0.153** ± **0.003**	0.249 ± 0.002
	336	0.326 ± 0.004	0.406 ± 0.001	0.245 ± 0.011	0.348 ± 0.007	**0.241** ± **0.005**	**0.337** ± **0.002**	0.202 ± 0.004	0.308 ± 0.004	**0.161** ± **0.002**	**0.261** ± **0.002**	0.162 ± 0.001	0.262 ± 0.001
	720	0.420 ± 0.094	0.467 ± 0.058	0.308 ± 0.019	0.393 ± 0.016	**0.303** ± **0.012**	**0.383** ± **0.011**	0.234 ± 0.006	0.333 ± 0.004	0.184 ± 0.003	0.283 ± 0.003	**0.183** ± **0.003**	**0.281** ± **0.002**
	960	0.399 ± 0.022	0.455 ± 0.017	0.335 ± 0.018	0.413 ± 0.015	**0.325** ± **0.019**	**0.398** ± **0.015**	0.235 ± 0.011	0.330 ± 0.008	**0.196** ± **0.005**	0.295 ± 0.005	0.200 ± 0.003	**0.292** ± **0.002**
Air quality	24	0.698 ± 0.064	0.626 ± 0.029	0.558 ± 0.011	0.537 ± 0.008	**0.527** ± **0.005**	**0.498** ± **0.003**	0.512 ± 0.029	0.514 ± 0.019	**0.488** ± **0.006**	0.486 ± 0.009	0.490 ± 0.006	**0.474** ± **0.004**
	48	0.955 ± 0.106	0.740 ± 0.035	0.722 ± 0.013	0.629 ± 0.005	**0.705** ± **0.019**	**0.600** ± **0.009**	0.712 ± 0.091	0.627 ± 0.047	**0.651** ± **0.032**	0.578 ± 0.023	0.659 ± 0.013	**0.566** ± **0.007**
	168	1.079 ± 0.108	0.818 ± 0.046	0.819 ± 0.007	0.691 ± 0.004	**0.789** ± **0.008**	**0.660** ± **0.005**	0.957 ± 0.067	0.737 ± 0.031	**0.787** ± **0.020**	0.648 ± 0.012	0.794 ± 0.025	**0.645** ± **0.014**
	336	1.105 ± 0.052	0.835 ± 0.021	0.902 ± 0.018	0.721 ± 0.010	**0.860** ± **0.017**	**0.685** ± **0.006**	0.989 ± 0.111	0.760 ± 0.046	0.870 ± 0.022	0.695 ± 0.011	**0.854** ± **0.029**	**0.676** ± **0.010**
	720	1.538 ± 0.419	0.968 ± 0.110	0.945 ± 0.030	0.757 ± 0.012	**0.842** ± **0.015**	**0.686** ± **0.008**	1.228 ± 0.048	0.858 ± 0.021	0.939 ± 0.064	0.730 ± 0.025	**0.839** ± **0.024**	**0.680** ± **0.013**
Nasdaq	30	5.500 ± 0.647	1.254 ± 0.086	**0.940** ± **0.055**	0.581 ± 0.037	1.023 ± **0.034**	0.577 ± **0.007**	1.742 ± 0.111	0.739 ± 0.028	1.111 ± 0.095	0.599 ± 0.020	**0.985** ± **0.018**	**0.564** ± **0.005**
	60	5.226 ± 0.424	1.236 ± 0.032	**0.989** ± **0.025**	**0.578** ± **0.008**	1.207 ± 0.044	0.617 ± 0.009	2.304 ± 0.062	0.790 ± 0.010	1.280 ± 0.023	0.630 ± 0.004	**1.161** ± **0.021**	**0.601** ± **0.003**
	120	6.023 ± 0.382	1.197 ± 0.034	**1.166** ± **0.014**	**0.615** ± **0.003**	1.959 ± 0.062	0.714 ± 0.006	3.227 ± 0.236	0.853 ± 0.007	2.585 ± 0.374	0.776 ± 0.031	**1.869** ± **0.037**	**0.697** ± **0.003**

RevIN shows a significant margin compared to the baseline for long prediction length. For example, when RevIN is added, the average DTW decreases from 38.348 to 15.240 for Informer, from 53.148 to 12.766 for N-BEATS, and from 20.498 to 11.080 for SCINet when the prediction length is 960. There are few cases where the proposed method predicts a less similar sequence than the base-

Table 8: **Comparison of RevIN and the baseline on similarity metrics.** We measure the similarity metrics, DTW and TDI, on the four datasets. We report the average value and standard deviation of five experiments.

Methods		Informer		**+ RevIN**		N-BEATS		**+ RevIN**		SCINet		**+ RevIN**	
Metric		TDI	DTW	TDI	DTW	TDI	DTW	TDI	DTW	TDI	DTW	TDI	DTW
$ETTh_1$	24	1.602 ± 0.093	2.515 ± 0.121	**1.207** ± **0.037**	**2.206** ± **0.097**	1.309 ± 0.147	2.361 ± 0.048	**1.031** ± **0.021**	**1.830** ± **0.012**	1.136 ± 0.113	1.784 ± 0.037	**0.969** ± **0.017**	**1.672** ± **0.007**
	48	3.932 ± 0.796	4.178 ± 0.333	**2.762** ± **0.190**	**3.401** ± **0.063**	2.430 ± 0.386	3.564 ± 0.230	**1.592** ± **0.055**	**2.745** ± **0.012**	2.418 ± 0.297	2.873 ± 0.117	**1.456** ± **0.045**	**2.508** ± **0.012**
	168	31.569 ± 4.652	10.384 ± 0.420	**9.459** ± **1.173**	**6.532** ± **0.227**	19.369 ± 3.679	8.937 ± 0.490	**6.190** ± **1.033**	**5.756** ± **0.161**	4.913 ± 1.085	5.344 ± 0.170	**3.791** ± **0.075**	**5.017** ± **0.008**
	336	72.959 ± 6.769	14.850 ± 0.514	**45.215** ± **17.098**	**11.605** ± **1.321**	47.961 ± 8.500	11.782 ± 0.531	**14.302** ± **2.127**	**8.532** ± **0.162**	**6.120** ± **0.329**	7.723 ± 0.087	6.973 ± 0.292	**7.319** ± **0.025**
	720	167.254 ± 13.008	23.295 ± 0.244	**98.648** ± **18.571**	**18.483** ± **0.346**	143.832 ± 28.984	20.494 ± 0.952	**26.265** ± **8.439**	**12.383** ± **0.458**	20.351 ± 3.588	11.550 ± 0.246	**11.337** ± **0.296**	**10.511** ± **0.056**
	960	182.008 ± 16.076	28.848 ± 1.217	**128.152** ± **9.174**	**22.018** ± **0.204**	148.671 ± 48.669	23.810 ± 2.418	**40.774** ± **13.985**	**15.107** ± **0.744**	24.067 ± 3.383	13.551 ± 0.270	**14.148** ± **0.461**	**12.508** ± **0.063**
$ETTh_2$	24	2.016 ± 0.152	2.481 ± 0.353	**1.528** ± **0.142**	**1.601** ± **0.029**	1.476 ± 0.230	2.307 ± 0.488	**1.128** ± **0.084**	**1.409** ± **0.007**	1.210 ± 0.128	1.350 ± 0.095	**1.088** ± **0.022**	**1.250** ± **0.015**
	48	**4.462** ± **0.567**	8.194 ± 0.397	4.802 ± 0.342	**2.806** ± **0.081**	4.244 ± 0.470	6.131 ± 0.542	**2.529** ± **0.165**	**2.207** ± **0.010**	3.368 ± 0.380	2.362 ± 0.183	**2.374** ± **0.058**	**1.906** ± **0.019**
	168	**24.016** ± **3.355**	32.496 ± 1.563	24.516 ± 1.705	**6.950** ± **0.313**	42.270 ± 1.708	28.487 ± 0.642	**14.759** ± **1.149**	**5.242** ± **0.104**	12.307 ± 1.637	4.899 ± 0.229	**10.205** ± **0.533**	**4.225** ± **0.083**
	336	74.168 ± 9.015	29.549 ± 2.453	**55.817** ± **3.180**	**9.572** ± **0.270**	95.645 ± 6.432	30.837 ± 1.115	**42.654** ± **3.450**	**8.063** ± **0.155**	28.823 ± 2.681	7.447 ± 0.350	**15.831** ± **0.141**	**5.975** ± **0.022**
	720	129.440 ± 22.426	35.908 ± 2.230	**111.842** ± **18.634**	**13.109** ± **0.391**	185.321 ± 17.061	47.461 ± 6.122	**92.521** ± **33.126**	**11.953** ± **0.977**	142.118 ± 36.524	16.428 ± 2.748	**26.577** ± **0.987**	**8.989** ± **0.035**
	960	177.336 ± 18.126	38.348 ± 1.665	**170.218** ± **14.187**	**15.240** ± **0.246**	281.720 ± 75.629	53.148 ± 9.408	**100.142** ± **29.742**	**12.766** ± **0.734**	218.477 ± 27.365	20.498 ± 1.979	**43.028** ± **2.007**	**11.080** ± **0.108**
$ETTm_1$	24	**2.343** ± **0.130**	1.730 ± 0.098	2.478 ± 0.094	**1.590** ± **0.040**	**3.109** ± **0.135**	2.062 ± 0.178	3.368 ± 0.067	**1.882** ± **0.026**	2.618 ± 0.096	1.194 ± 0.023	**2.134** ± **0.034**	**0.995** ± **0.007**
	48	4.818 ± 0.122	3.133 ± 0.097	**4.116** ± **0.093**	**2.457** ± **0.038**	5.223 ± 0.340	3.036 ± 0.083	**4.400** ± **0.084**	**2.426** ± **0.061**	4.176 ± 0.355	1.661 ± 0.077	**3.317** ± **0.045**	**1.470** ± **0.015**
	96	8.550 ± 0.671	4.813 ± 0.228	**5.837** ± **0.098**	**3.586** ± **0.085**	9.686 ± 1.050	5.220 ± 0.162	**6.803** ± **0.271**	**3.803** ± **0.090**	5.479 ± 0.349	2.409 ± 0.093	**4.782** ± **0.045**	**2.180** ± **0.009**
	288	32.918 ± 2.223	11.054 ± 0.341	**17.231** ± **0.759**	**7.403** ± **0.122**	42.809 ± 3.710	10.706 ± 0.513	**15.942** ± **0.531**	**7.285** ± **0.064**	22.650 ± 2.713	5.365 ± 0.072	**15.457** ± **0.273**	**4.436** ± **0.033**
	672	80.546 ± 17.739	16.365 ± 0.749	**38.265** ± **6.209**	**12.030** ± **0.455**	111.531 ± 13.075	16.601 ± 0.649	**45.064** ± **5.369**	**12.482** ± **0.311**	149.483 ± 10.180	13.980 ± 0.350	**44.721** ± **1.867**	**8.040** ± **0.126**
	1344	161.539 ± 16.808	23.822 ± 1.203	**77.844** ± **7.652**	**17.293** ± **0.606**	444.199 ± 59.765	69.125 ± 63.858	**161.725** ± **99.382**	**19.490** ± **3.115**	397.100 ± 39.675	27.319 ± 2.824	**98.890** ± **4.495**	**12.779** ± **0.156**
ECL	24	0.517 ± 0.007	1.610 ± 0.015	**0.339** ± **0.005**	**1.234** ± **0.006**	0.506 ± 0.004	1.702 ± 0.017	**0.384** ± **0.002**	**1.368** ± **0.010**	0.368 ± 0.017	1.171 ± 0.016	**0.281** ± **0.001**	**1.058** ± **0.053**
	48	0.677 ± 0.029	2.375 ± 0.038	**0.383** ± **0.006**	**1.806** ± **0.012**	0.676 ± 0.025	2.444 ± 0.028	**0.446** ± **0.008**	**1.972** ± **0.011**	0.466 ± 0.034	1.752 ± 0.038	**0.318** ± **0.006**	**1.527** ± **0.004**
	168	1.338 ± 0.017	4.364 ± 0.071	**0.785** ± **0.024**	**3.709** ± **0.054**	1.738 ± 0.166	4.576 ± 0.087	**0.839** ± **0.013**	**3.828** ± **0.016**	0.955 ± 0.077	3.379 ± 0.033	**0.730** ± **0.029**	**3.122** ± **0.026**
	336	2.276 ± 0.046	6.358 ± 0.148	**1.495** ± **0.044**	**5.640** ± **0.126**	3.555 ± 0.562	6.448 ± 0.049	**1.661** ± **0.083**	**5.702** ± **0.038**	2.336 ± 0.209	5.103 ± 0.033	**1.194** ± **0.034**	**4.591** ± **0.012**
	720	23.481 ± 15.963	**16.561** ± **5.353**	**22.117** ± **12.615**	17.227 ± 3.790	12.379 ± 4.196	10.841 ± 1.204	**5.583** ± **1.060**	**9.239** ± **0.230**	4.576 ± 0.806	7.913 ± 0.102	**2.830** ± **0.092**	**7.162** ± **0.056**
	960	32.548 ± 18.405	**21.065** ± **3.649**	**28.328** ± **8.781**	21.717 ± 3.002	15.866 ± 3.309	12.128 ± 0.377	**9.198** ± **1.400**	**11.140** ± **0.347**	6.972 ± 0.678	9.255 ± 0.108	**4.502** ± **0.233**	**8.560** ± **0.064**

line, but the margin is minimal in terms of either DTW or TDI compared to the significant margin found when our method outperforms the baseline. These results demonstrate that the model adopting RevIN can generate a sequence more similar to the groundtruth than the baseline, especially showing a better prediction accuracy on the long sequence.

A.6 Comparison with Existing Dynamic Normalization Methods

We compare RevIN with the normalization methods proposed in LSTNet (Lai et al., 2017) and ES-RNN (Smyl, 2020), as shown in Table 9. Similarly to adding RevIN to the baseline model, we add the autoregressive linear bypass module of LSTNet and the modified Holt-Winters exponential smoothing of ES-RNN to the baseline model, respectively. RevIN consistently shows the best performance among the normalization methods adopted on N-BEATS showing a significant margin. The autoregressive linear bypass module of LSTNet (LSTNet*) also consistently reduces the prediction error compared to the baseline, but the performance improvement is smaller than our method. For example, when the prediction length is 960 on the $ETTh_2$ dataset, N-BEATS shows an average error of 6.408 and LSTNet* reduces the error to 5.627, but which is still much higher than RevIN that reduces the error to 0.471. Similarly, when the prediction length is 1344 on the $ETTm_1$ dataset, the baseline shows an average error of 14.613 and LSTNet* largely decreases the error to 5.592, but RevIN more significantly decreases the error to 0.631. LSTNet utilizes a linear autoregressive

Table 9: **Forecasting performance of RevIN in comparison with existing dynamic normalization methods.** We compare RevIN with LSTNet* that adds the autoregressive linear bypass module of LSTNet to the baseline and ES-RNN* that adds the exponential smoothing of ES-RNN to the baseline. The experiments are conducted on the ETTh$_1$, ETTh$_2$, ETTm$_1$, ECL, and the M4 dataset using N-BEATS as the baseline model. The missing performances are where the model fails to converge.

Methods		N-BEATS		+ LSTNet*		+ ESRNN*		**+ RevIN**	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh$_1$	24	0.478 ± 0.022	0.505 ± 0.012	0.462 ± 0.047	0.497 ± 0.035	0.547 ± 0.031	0.515 ± 0.021	**0.330 ± 0.006**	**0.373 ± 0.004**
	48	0.536 ± 0.060	0.542 ± 0.041	0.587 ± 0.063	0.576 ± 0.043	0.662 ± 0.027	0.567 ± 0.012	**0.372 ± 0.001**	**0.400 ± 0.002**
	168	1.005 ± 0.146	0.782 ± 0.064	1.031 ± 0.099	0.795 ± 0.054	0.698 ± 0.044	0.600 ± 0.018	**0.466 ± 0.030**	**0.452 ± 0.014**
	336	0.932 ± 0.079	0.743 ± 0.042	0.964 ± 0.047	0.760 ± 0.023	0.768 ± 0.041	0.640 ± 0.022	**0.515 ± 0.013**	**0.483 ± 0.008**
	720	1.389 ± 0.230	0.926 ± 0.066	1.549 ± 0.061	0.994 ± 0.022	0.966 ± 0.084	0.742 ± 0.022	**0.576 ± 0.035**	**0.534 ± 0.018**
	960	1.383 ± 0.380	0.932 ± 0.120	1.293 ± 0.059	0.897 ± 0.026	-	-	**0.678 ± 0.019**	**0.575 ± 0.009**
ETTh$_2$	24	0.403 ± 0.185	0.472 ± 0.101	0.394 ± 0.099	0.485 ± 0.068	0.614 ± 0.010	0.522 ± 0.004	**0.192 ± 0.003**	**0.276 ± 0.002**
	48	1.330 ± 0.240	0.918 ± 0.073	1.261 ± 0.214	0.907 ± 0.075	0.654 ± 0.009	0.543 ± 0.007	**0.254 ± 0.011**	**0.320 ± 0.008**
	168	7.174 ± 0.449	2.329 ± 0.049	7.053 ± 0.428	2.290 ± 0.095	0.962 ± 0.129	0.696 ± 0.054	**0.410 ± 0.010**	**0.418 ± 0.005**
	336	4.859 ± 0.268	1.863 ± 0.043	5.070 ± 0.336	1.914 ± 0.083	1.204 ± 0.158	0.789 ± 0.050	**0.449 ± 0.011**	**0.447 ± 0.006**
	720	5.656 ± 1.053	2.012 ± 0.186	6.311 ± 2.057	2.049 ± 0.225	1.284 ± 0.145	0.810 ± 0.033	**0.496 ± 0.008**	**0.482 ± 0.002**
	960	6.408 ± 2.039	2.077 ± 0.242	5.627 ± 1.670	1.965 ± 0.314	1.338 ± 0.535	0.809 ± 0.127	**0.471 ± 0.015**	**0.481 ± 0.008**
ETTm$_1$	24	0.443 ± 0.043	0.437 ± 0.035	0.412 ± 0.026	0.426 ± 0.024	0.564 ± 0.015	0.477 ± 0.009	**0.403 ± 0.006**	**0.392 ± 0.005**
	48	0.453 ± 0.034	0.472 ± 0.018	0.420 ± 0.028	0.455 ± 0.018	0.615 ± 0.093	0.531 ± 0.049	**0.328 ± 0.010**	**0.371 ± 0.007**
	96	0.603 ± 0.051	0.581 ± 0.027	0.572 ± 0.039	0.553 ± 0.030	0.668 ± 0.031	0.555 ± 0.015	**0.379 ± 0.011**	**0.406 ± 0.007**
	288	0.849 ± 0.095	0.702 ± 0.051	0.789 ± 0.069	0.677 ± 0.039	0.795 ± 0.070	0.623 ± 0.032	**0.451 ± 0.016**	**0.445 ± 0.008**
	672	0.860 ± 0.057	0.726 ± 0.026	0.958 ± 0.183	0.758 ± 0.076	1.657 ± 1.116	0.890 ± 0.290	**0.555 ± 0.011**	**0.511 ± 0.008**
	1344	14.613 ± 26.108	1.948 ± 1.655	5.592 ± 7.032	1.497 ± 0.671	-	-	**0.631 ± 0.061**	**0.556 ± 0.020**
ECL	24	0.279 ± 0.007	0.372 ± 0.003	0.198 ± 0.005	0.310 ± 0.003	0.242 ± 0.005	0.332 ± 0.006	**0.176 ± 0.002**	**0.285 ± 0.001**
	48	0.309 ± 0.007	0.388 ± 0.004	0.245 ± 0.009	0.343 ± 0.007	0.275 ± 0.007	0.352 ± 0.006	**0.194 ± 0.001**	**0.301 ± 0.001**
	168	0.333 ± 0.016	0.410 ± 0.012	0.285 ± 0.006	0.375 ± 0.004	-	-	**0.218 ± 0.002**	**0.320 ± 0.001**
	336	0.326 ± 0.004	0.406 ± 0.001	0.304 ± 0.019	0.393 ± 0.013	-	-	**0.241 ± 0.005**	**0.337 ± 0.002**
	720	0.420 ± 0.094	0.467 ± 0.058	0.378 ± 0.083	0.443 ± 0.056	-	-	**0.303 ± 0.012**	**0.383 ± 0.011**
	960	0.399 ± 0.022	0.455 ± 0.017	0.360 ± 0.037	0.433 ± 0.027	-	-	**0.325 ± 0.019**	**0.398 ± 0.015**
M4	average	0.224 ± 0.004	0.207 ± 0.003	0.223 ± 0.004	0.206 ± 0.003	0.223 ± 0.001	0.204 ± 0.001	**0.208 ± 0.001**	**0.197 ± 0.001**

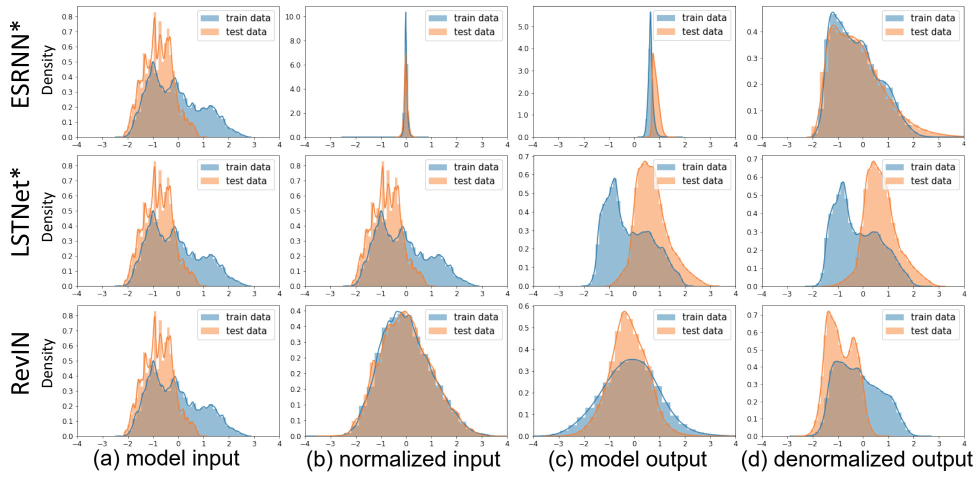

Figure 8: **Effect of RevIN on distribution discrepancy between train and test data compared to existing dynamic normalization methods.** We compare RevIN with LSTNet[*] that adds the autoregressive linear bypass module of LSTNet to the baseline and ES-RNN[*] that adds the exponential smoothing of ES-RNN to the baseline. The analysis is conducted on the ETTh_2 dataset with a prediction length of 960 using N-BEATS as the baseline model. From left to right columns, we compare the train and test data distributions on each step of the sequential process in each method.

model to make the non-linear neural network robust to scale change. However, it does not change the statistics of the input sequence. Thus, the discrepancy between the train and test datasets still exists for the non-linear neural network. In the case of ES-RNN[*], the training of the model is unstable and it often degrades the baseline performance, e.g., when the prediction length is 24 and 48 on the ETTh_1 and ETTm_1 datasets and the prediction length is 672 on the ETTm_1 dataset. However, it significantly reduces the error for the long prediction length much better than LSTNet[*], but still worse than RevIN, e.g., when the prediction length is 960 on the ETTh_2 dataset.

Additionally, we further analyze the data distributions of the methods on the ETTh_2 dataset with the prediction length of 960 as shown in Fig. 8. From left to right columns, we compare the train and test data distributions on each step of the sequential process in each method.

ES-RNN[*] shows the distributions of (a) the original input, (b) the normalized input where the level and seasonality are removed by its proposed method, (c) the model prediction output, (d) the denormalized output where the level and seasonality is multiplied back to the original distribution.

LSTNet[*] shows the distributions of (a) the original input, (b) the same original input since the method does not transform the input data before feeding it to the prediction model, the model prediction output before (c) and after (d) adding the output of the autoregressive model.

RevIN shows the distributions of (a) the original input, (b) the input normalized by RevIN, (c) the model prediction output, and (d) the denormalized output.

In Fig. 8(a), the train and test data shows discrepancy in their distributions. Also, they have several peaks, not being centered to the mean. This implies that time-series in the data would have different means from each other. In Fig. 8(b), both RevIN and ES-RNN[*] transform data distributions into mean-centered distributions. Particularly, ES-RNN[*] makes the distribution extremely concentrated on the mean with a small variance. The two methods make sequences have similar statistics, thereby alleviating the distribution shift problem in the input data. This would lead to competitive performance on the long prediction sequence in contrast to LSTNet[*].

LSTNet[*] cannot resolve the distribution discrepancy because it has no module that can change the input statistics. Also, the distributions of the model output (Fig. 8(c)) completely differ from the input (Fig. 8(b)). The methods cannot make the input and output distribution to be consistent. In addition, its autoregressive model barely affects the model output distribution as shown in Fig. 8(c,d). There is almost no difference between the model output and the final output in LSTNet[*].

Also, the distributions of the final output (Fig. 8(d)) significantly differ from the original data (Fig. 8(a)) in LSTNet*, which implies that they fail to learn the appropriate data distribution, and this would be the main reason for higher prediction error than RevIN. Similarly, although ES-RNN* alleviates the distribution discrepancy in the input data, it fails to return the model output (Fig. 8(d)) back to the original distribution (Fig. 8(a)), especially for the test data.

On the other hand, in RevIN, the distributions of the final output (Fig. 8(d)) return well to the original distributions (Fig. 8(a)). Also, RevIN makes the input and output of the model to maintain the consistent distributions while the train and test data are overlapped. Thus, RevIN can show robust performance against the input and output lengths.

A.7 Additional Results on Comparison with Other Normalization Methods

This section provides the additional results on comparison with other normalization methods, which can not be located in the main manuscript due to the lack of space. The forecasting error of RevIN and existing normalization methods are evaluated on the ETTh$_1$, ETTh$_2$, ETTm$_1$, and ECL datasets in Table 10. RevIN consistently outperforms the other normalization methods across all datasets. Interestingly, when the denormalization step is added to batch normalization (RevBN) as RevIN, the model better forecasts the long sequence, e.g., when the prediction length is 960, than batch normalization.

In the case of DAIN, it learns to transform the mean and variance values of the current input and use the new values to normalize the data. As a result, the input sequences can have different means and variances from each other, which indicates that the distribution shift can still exist after the normalization. As shown in our paper, the denormalization step shows a critical role in improving the model performance by recovering the prediction to the original value. However, a denormalization step cannot be added to DAIN since it has the Hadamard multiplication operation at the last step, which is not reversible when the input and prediction sequence lengths are different. These differences would be the reason for its inferior performance to RevIN, even requiring more computational costs and a larger amount of model parameters (at least three $D \times D$ matrices, where D is the number of variables).

A.8 Algorithm of Reversible Instance Normalization

Algorithm 7 summarizes the RevIN procedure. Reversible instance normalization is the symmetric structure of the normalization (line 4-5) and denormalization operation (line 7-8). RevIN simply scales and shifts the input (line 2-5) and output (line 7-8) of a model using identical statistics. Indeed, g_θ in Algorithm 7 (line 6) can be any network. RevIN is a model-agnostic normalization method easily adopted to any model with negligible cost but effectively improves forecasting performance.

Table 10: **Comparison with classical and state-of-the-art normalization methods.** The mean absolute error is measured on the ETTh$_1$, ETTh$_2$, ETTm$_1$, and ECL datasets. T_y indicates the prediction length. **RevBN** is the modified version of RevIN, where the input normalization is replaced with the batch normalization.

Dataset	T_y	Min-max norm	z-score norm	Layer norm	DAIN	Batch norm	**RevBN**	Instance norm	**RevIN (Ours)**
ETTh$_1$	24	0.885	0.959	0.472	0.652	0.451	0.574	0.989	**0.322**
	48	1.010	0.898	0.741	1.389	0.557	0.649	0.999	**0.373**
	168	1.074	0.953	0.871	0.996	0.851	0.717	0.946	**0.515**
	336	1.083	0.969	0.827	0.979	0.828	0.775	1.078	**0.509**
	720	1.226	0.978	1.184	1.014	0.916	0.705	0.986	**0.567**
	960	1.224	1.043	1.303	1.032	1.691	0.779	1.090	**0.697**
ETTh$_2$	24	2.659	3.152	0.478	1.437	0.336	0.550	2.976	**0.192**
	48	2.772	3.232	1.335	1.476	1.018	1.058	3.175	**0.244**
	168	2.987	3.329	4.092	1.982	6.206	0.729	3.240	**0.419**
	336	2.914	3.288	4.207	2.631	5.422	0.546	3.186	**0.452**
	720	3.092	3.031	5.822	2.954	7.062	1.552	3.079	**0.492**
	960	3.308	3.087	5.204	2.802	7.755	2.148	3.145	**0.465**
ETTm$_1$	24	0.981	0.930	0.515	0.431	0.477	0.680	0.926	**0.395**
	48	0.998	1.005	0.555	0.747	0.489	0.531	1.005	**0.337**
	96	1.035	1.016	0.502	0.672	0.505	0.601	1.021	**0.388**
	288	0.974	0.988	0.773	0.877	0.677	0.656	1.056	**0.444**
	672	1.157	1.029	0.795	1.043	0.620	0.670	1.157	**0.549**
	1344	1.320	1.274	2.488	1.348	1.147	0.991	1.329	**0.602**
ECL	24	0.370	0.313	0.294	0.348	0.301	0.304	0.307	**0.174**
	48	0.334	0.326	0.310	0.387	0.319	0.331	0.313	**0.194**
	168	0.374	0.335	0.343	0.347	0.320	0.327	0.333	**0.220**
	336	0.378	0.338	0.358	0.357	0.337	0.369	0.335	**0.244**
	720	0.746	0.417	0.371	0.366	0.374	0.440	0.378	**0.294**
	960	0.387	0.377	0.379	0.381	0.422	0.414	0.386	**0.329**

Algorithm 1: RevIN, applied to input x and output y of a module in the model.

Input : T_x, input sequence length; $x_{kt}^{(i)}$, the k-th feature at time step t of the i-th item in a mini-batch; $\gamma, \beta \in \mathbb{R}^K$, parameters to be learned for RevIN; g_θ, a module in the model.
Output: γ, β, θ.

```
1 Compute $\mu_\mathcal{T} \leftarrow \frac{1}{T_x}\sum_{j=1}^{T_x} x_{kj}^{(i)}$    ▷ instance mean
2 Compute $\sigma_\mathcal{T}^2 \leftarrow \frac{1}{T_x}\sum_{j=1}^{T_x} (x_{kj}^{(i)} - \mu_\mathcal{T})^2$    ▷ instance variance
3 ┌Normalize $\hat{x}_{kt}^{(i)} \leftarrow \frac{x_{kt}^{(i)} - \mu_\mathcal{T}}{\sqrt{\sigma_\mathcal{T}^2 + \epsilon}}$    ▷ normalization
4 │ ┌Transform $\hat{x}_{kt}^{(i)} \leftarrow \gamma_k \cdot \hat{x}_{kt}^{(i)} + \beta_k \equiv \textbf{RevIN}^{\text{n}}_{\gamma,\beta}(x_{kt}^{(i)})$    ▷ scale and shift
5 │ │ Predict $\tilde{y} \leftarrow g_\theta(\hat{x})$    ▷ model prediction
6 │ └Retransform $\hat{y}_{kt}^{(i)} \leftarrow \frac{\tilde{y}_{kt}^{(i)} - \beta_k}{\gamma_k}$    ▷ reverse scale and shift
7 └Denormalize $\hat{y}_{kt}^{(i)} \leftarrow \mu_\mathcal{T} + \hat{y}_{kt}^{(i)} \sqrt{\sigma_\mathcal{T}^2 + \epsilon} \equiv \textbf{RevIN}^{\text{dn}}_{\gamma,\beta}(\tilde{y}_{kt}^{(i)})$    ▷ denormalization
```

A.9 Theoretical Justification of RevIN Against Distribution Shift Problem

Let $\{x^{(i)}\}_{i=1}^{N}$ be our data, where $x^{(i)} \in \mathbb{R}^{K \times T_x}$ denotes a time-series with K variables of length T_x. Consider a simple univariate case where $K = 1$ without the loss of generality. Then, $x^{(i)} \in \mathbb{R}^{T_x}$ denotes the i-th time-series in the data. Consider two such data, the training and test data, whose distributions are denoted as P_{tra} and P_{tst}, respectively. The distribution shift appears when distribution of the data sequences are different (Du et al., 2021). That is,

$$P_{tra}(x) \neq P_{tst}(x). \tag{4}$$

In our work, we consider the distribution shift problem in terms of the mean and the variance. Assume that the given data suffer from the distribution shift problem in terms of the mean and variance, which can be expressed as

$$\mathbb{E}[x_{tra}] \neq \mathbb{E}[x_{tst}] \;\text{ or }\; \mathrm{Var}[x_{tra}] \neq \mathrm{Var}[x_{tst}], \;\text{ for }\; x_{tra} \sim P_{tra},\; x_{tst} \sim P_{tst}. \tag{5}$$

In order to solve this problem, the normalization step of RevIN can be applied to the input training sample $x^{(i)} \sim P_{tra}$. Then, the input sample can be transformed as

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mathbb{E}[x^{(i)}]}{\sqrt{\mathrm{Var}[x^{(i)}]}} \cdot \gamma + \beta. \tag{6}$$

By the laws of expectation and variance,

$$\mathbb{E}[\hat{x}^{(i)}] = \beta \;\text{ and }\; \mathrm{Var}[\hat{x}^{(i)}] = \gamma^2, \;\text{ for all }\; i \tag{7}$$

By symmetry, this also holds for $x^{(i)} \sim P_{tst}$, for all i. Therefore, the mean and variance of the training and test data distributions become identical. Thus, by definition, the distribution shift problem is solved for the input training and test data by the normalization step. Given the normalized time-series data, the forecasting model $f : \mathbb{R}^{T_x} \to \mathbb{R}^{T_y}$, parameterized by θ, predicts the corresponding future values as $\tilde{y} = f_\theta(\hat{x})$. Then, the denormalization step of RevIN returns the dynamics of the original data, $\mathbb{E}[x^{(i)}]$ and $\mathrm{Var}[x^{(i)}]$ to the prediction output so that model does not have to reconstruct them from the normalized input. The model prediction $\tilde{y}^{(i)}$ can be denormalized as

$$\hat{y}^{(i)} = \frac{\tilde{y}^{(i)} - \beta}{\gamma} \cdot \sqrt{\mathrm{Var}[x^{(i)}]} + \mathbb{E}[x^{(i)}] \tag{8}$$

By the laws of expectation and variance, the mean and variance of $\tilde{y}^{(i)}$ can be expressed as

$$\mathbb{E}[\hat{y}^{(i)}] = \Delta + \mathbb{E}[x^{(i)}] \;\text{ and }\; \mathrm{Var}[\hat{y}^{(i)}] = \lambda \cdot \mathrm{Var}[x^{(i)}]. \tag{9}$$

The denormalization step allows the mean and variance of the final prediction values to be expressed as the difference from the input statistics. Here, since the input data $x^{(i)}$ and the groundtruth future values are consecutive sequences, one can assume that their difference in the mean and variance can be expressed as Eq. (9) as well. Under the assumption, when adopting the denormalization step of RevIN, the model only needs to capture the difference from the input statistics, Δ and λ, to accurately predict the statistics of the future values. The denormalization step allows the model to focus on capturing the difference in the distribution between input and output by removing all common non-stationary statistics between the input and output to the model.

A.10 Calculation Details on Feature Divergence

This section explains how the feature divergence in Section 4.2.3 is calculated. Following previous work (Pan et al., 2018), we calculate the average feature divergence of each feature between train and test datasets using symmetric KL divergence, assuming that the output features follow a Gaussian distribution with mean μ and variance σ^2. Then, the equation for feature divergence of the k-th feature f_k is

$$D(f_k^{\text{train}} || f_k^{\text{test}}) = KL(f_k^{\text{train}} || f_k^{\text{test}}) + KL(f_k^{\text{test}} || f_k^{\text{train}}), \tag{10}$$

$$\text{where} \quad KL(f_k^A || f_k^B) = \log \frac{\sigma_k^B}{\sigma_k^A} + \frac{{\sigma_k^A}^2 + (\mu_k^A - \mu_k^B)^2}{2{\sigma_k^B}^2} - \frac{1}{2}. \tag{11}$$

A.11 Experimental Details

We exactly follow the experimental settings of the baseline models by using their official code to conduct experiments except for N-BEATS that have no officially released code. We reproduce the model and set hyperparameters as stated in the original N-BEATS paper (Oreshkin et al., 2020). Additionally, we conduct a grid search for the learning rate of N-BEATS with the range of [1e-5, 1e-3] and train the model with the weight decay with the factor of 0.001 to stabilize training. We train and test the models with the following seeds: 12, 22, 32, 42, and 52. Experiments using N-BEATS and Informer as backbone are performed on NVIDIA TITAN RTX, and experiments using SCINet are conducted on NVIDIA TITAN Xp. Following the multivariate time-series forecasting settings of prior work (Zhou et al., 2021; Liu et al., 2021), we select the input window lengths from 2d, 4d, 7d, 14d, 15d, 20d for the hourly datasets, the $ETTh_1$, $ETTh_2$, and ECL datasets, and from 12h, 1d, 4d, 7d for the $ETTm_1$ dataset. Particularly, we set the ratio of input length to prediction length to be from 2.0 to 0.35 as the prediction length becomes longer. In the case of SCINet, we follow the setting of the original paper for the input length since the method requires the input sequence length to meet a specific condition.

A.12 Reproduction Details for Baseline Models

This section describes the implementation details of baselines, Informer, N-BEATS, and SCINet. Note that we compare each baseline model and RevIN using the same hyperparameters except for the presence of RevIN.

Informer. We use the official open-source code of Informer [6]. We follow the provided hyperparameters for the experiments they studied. In the case of the ECL dataset, we use the same hyperparameter settings for the $ETTh_1$ dataset since it is not officially provided.

N-BEATS. N-BEATS is reproduced using the PyTorch framework (Paszke et al., 2019). We follow the hyperparameter settings of the N-BEATS-I model. We train N-BEATS to minimize the mean squared error between the predicted results and groundtruth values. To compare with other methods, we report the prediction results of a single model instead of the ensemble method that N-BEATS used in the original paper. Since N-BEATS is the model tailored to univariate time-series forecasting, we flatten the multivariate input sequence to have a single dimension before feeding it to the model.

SCINet. We follow the experimental settings provided in the official code [7] of SCINet. Different from other baseline models, when the prediction length is 720, the input sequence length is set as 736 since SCINet requires the input length to be the multiple of 32 due to its hierarchical architecture (Liu et al., 2021).

A.13 Additional Qualitative Results

Informer fails to capture the repeated pattern of increase and decrease on $ETTh_2$ dataset. Moreover, the prediction results of Informer are inaccurately scaled and shifted. With RevIN, Informer predicts reasonable results. Without any modification to the network architecture, RevIN can improve the forecasting performance. Although the pattern is accurately captured for N-BEATS, the prediction results are inaccurately scaled, resulting in low performance. RevIN improves the performance of N-BEATS in that the model predicts accurately. For SCINet, RevIN improves the baseline to predict more precise results.

A.14 Extra Quantitative Results

Table 11 and Table 12 show the standard deviation for five independent experiments of cross-domain evaluation and for comparison on long sequence forecasting, respectively. Table 13 describes the full results of comparison with baselines and RevIN, including the standard deviation for five runs and comparison with the reported performance in the original papers. It can be seen as RevIN improves the performance as statistically significant than previous state-of-the-art models.

[6]https://github.com/zhouhaoyi/Informer2020

[7]https://github.com/cure-lab/SCINet

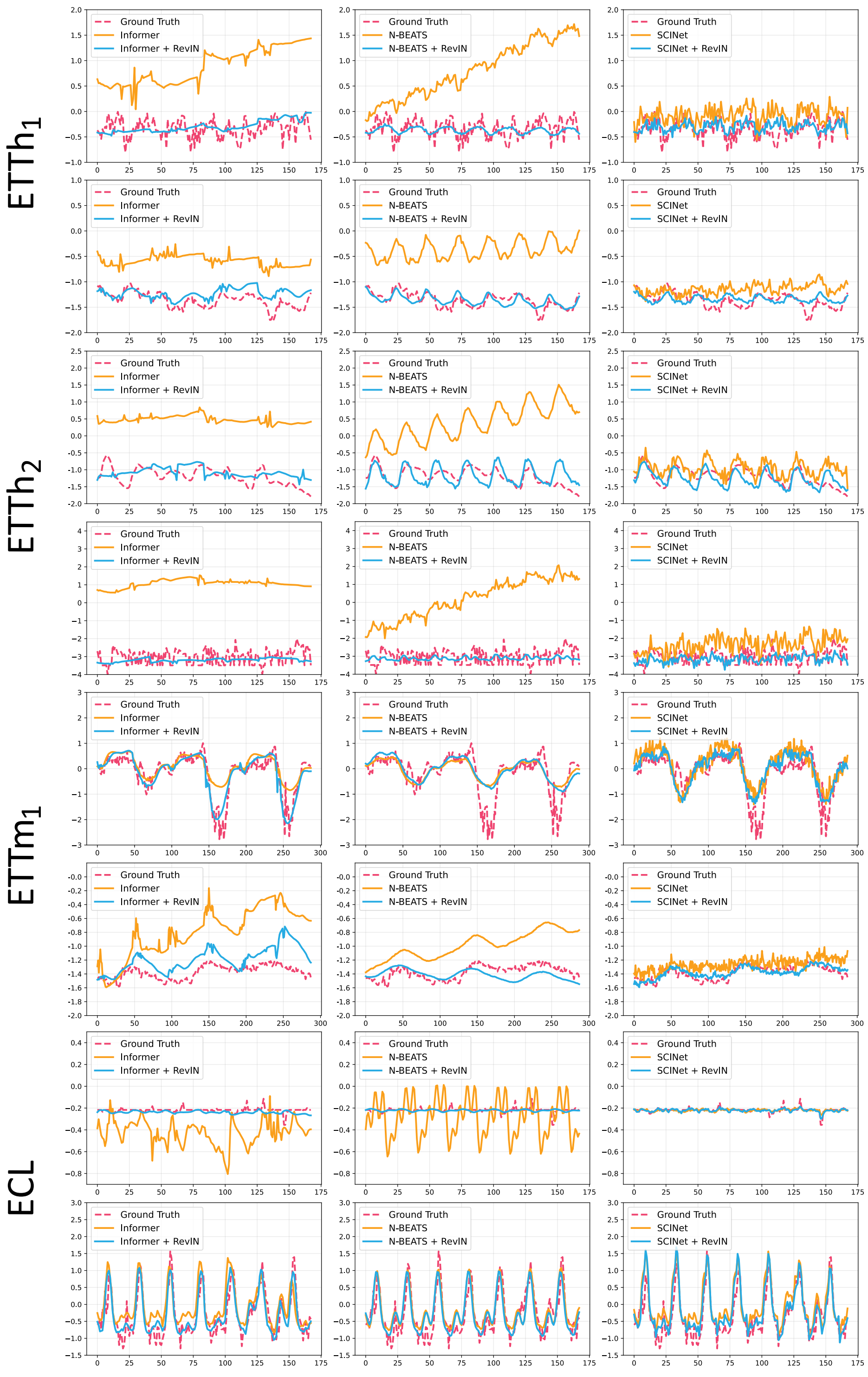

Figure 9: **Additional results on prediction comparison of RevIN and the existing SOTA models.** The models are evaluated on ETTh$_1$, ETTh$_2$, and ECL datasets, with a prediction length of 168. For the ETTm$_1$ dataset, we evaluate the model with a prediction length of 288. Our method consistently improves the baseline models in terms of the scale and shift of the predicted sequence.

Table 11: **The standard deviation of the five runs for cross-domain evaluation results.**

Train		ETTh$_1$				ETTh$_2$				ETTm$_1$			
Test		ETTh$_2$		ETTm$_1$		ETTh$_1$		ETTm$_1$		ETTh$_1$		ETTh$_2$	
Prediction length		336	960	336	960	336	960	336	960	288	1344	288	1344
SCINet	MSE	0.034	0.056	0.043	0.029	0.020	0.061	0.049	0.110	0.032	0.026	0.009	0.166
	MAE	0.024	0.027	0.032	0.013	0.014	0.035	0.031	0.046	0.015	0.016	0.004	0.078
+ RevIN	MSE	0.010	0.007	0.008	0.004	0.006	0.015	0.009	0.001	0.003	0.013	0.005	0.010
	MAE	0.006	0.004	0.005	0.001	0.004	0.007	0.004	0.001	0.002	0.006	0.002	0.006

Table 12: **The standard deviation of the five runs for comparison on long sequence forecasting.**

Prediction length	48		168		336		720		960	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Informer	0.056	0.035	0.052	0.024	0.085	0.031	0.037	0.024	0.034	0.022
+ RevIN	0.030	0.008	0.051	0.030	0.073	0.026	0.067	0.033	0.041	0.022
N-BEATS	0.042	0.030	0.056	0.027	0.081	0.037	0.072	0.029	0.067	0.030
+ RevIN	0.006	0.003	0.014	0.007	0.011	0.006	0.041	0.020	0.027	0.012
SCINet	0.008	0.007	0.063	0.042	0.115	0.064	0.033	0.022	0.049	0.029
+ RevIN	0.002	0.001	0.013	0.007	0.022	0.010	0.028	0.017	0.015	0.008

Table 13: **Time-series forecasting results.** The mean and standard deviation of the five independent experiments are recorded. † denotes that the results are obtained from the original paper.

Methods		Informer†		Informer		Informer +**RevIN**		N-BEATS		N-BEATS+**RevIN**		SCINet†		SCINet		SCINet+**RevIN**	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	0.577	0.549	0.550 ± 0.041	0.536 ± 0.025	**0.504 ± 0.049**	**0.472 ± 0.025**	0.478 ± 0.022	0.505 ± 0.012	**0.330 ± 0.006**	**0.373 ± 0.004**	0.311	0.348	0.338 ± 0.012	0.373 ± 0.009	**0.308 ± 0.003**	**0.347 ± 0.002**
	48	0.685	0.625	0.772 ± 0.122	0.668 ± 0.055	**0.646 ± 0.039**	**0.547 ± 0.015**	0.536 ± 0.060	0.542 ± 0.041	**0.372 ± 0.001**	**0.400 ± 0.002**	0.364	0.388	0.436 ± 0.025	0.459 ± 0.021	**0.365 ± 0.005**	**0.389 ± 0.003**
	168	0.931	0.752	1.138 ± 0.096	0.853 ± 0.045	**0.655 ± 0.055**	**0.561 ± 0.024**	1.005 ± 0.146	0.782 ± 0.064	**0.466 ± 0.030**	**0.452 ± 0.014**	0.497	0.491	0.459 ± 0.015	0.461 ± 0.013	**0.406 ± 0.003**	**0.416 ± 0.003**
	336	1.128	0.873	1.278 ± 0.129	0.909 ± 0.058	**1.058 ± 0.119**	**0.758 ± 0.059**	0.932 ± 0.079	0.743 ± 0.042	**0.515 ± 0.013**	**0.483 ± 0.008**	0.491	0.494	0.527 ± 0.010	0.513 ± 0.006	**0.467 ± 0.005**	**0.471 ± 0.003**
	720	1.215	0.896	1.357 ± 0.056	0.945 ± 0.009	**0.926 ± 0.057**	**0.717 ± 0.036**	1.389 ± 0.230	0.926 ± 0.066	**0.576 ± 0.035**	**0.534 ± 0.018**	0.612	0.582	0.596 ± 0.015	0.571 ± 0.013	**0.507 ± 0.006**	**0.505 ± 0.004**
	960	.	.	1.470 ± 0.124	0.990 ± 0.052	**0.902 ± 0.033**	**0.715 ± 0.025**	1.383 ± 0.380	0.932 ± 0.120	**0.678 ± 0.019**	**0.575 ± 0.009**	.	.	0.604 ± 0.017	0.574 ± 0.014	**0.545 ± 0.010**	**0.526 ± 0.005**
$ETTh_2$	24	0.720	0.665	0.450 ± 0.099	0.520 ± 0.071	**0.238 ± 0.010**	**0.325 ± 0.006**	0.403 ± 0.185	0.472 ± 0.101	**0.192 ± 0.003**	**0.276 ± 0.002**	0.183	0.271	0.199 ± 0.026	0.295 ± 0.027	**0.180 ± 0.004**	**0.263 ± 0.002**
	48	1.457	1.001	2.171 ± 0.094	1.200 ± 0.048	**0.361 ± 0.023**	**0.404 ± 0.014**	1.330 ± 0.240	0.918 ± 0.073	**0.254 ± 0.011**	**0.320 ± 0.008**	0.259	0.341	0.350 ± 0.025	0.422 ± 0.027	**0.231 ± 0.006**	**0.302 ± 0.006**
	168	3.489	1.515	8.157 ± 0.631	2.558 ± 0.113	**0.859 ± 0.072**	**0.649 ± 0.026**	7.174 ± 0.449	2.329 ± 0.049	**0.410 ± 0.010**	**0.418 ± 0.005**	0.528	0.509	0.559 ± 0.044	0.518 ± 0.025	**0.337 ± 0.007**	**0.378 ± 0.003**
	336	2.723	1.340	4.746 ± 0.455	1.844 ± 0.102	**0.890 ± 0.057**	**0.673 ± 0.023**	4.859 ± 0.268	1.863 ± 0.043	**0.449 ± 0.011**	**0.447 ± 0.006**	0.648	0.608	0.664 ± 0.073	0.583 ± 0.030	**0.357 ± 0.003**	**0.403 ± 0.002**
	720	3.467	1.473	3.190 ± 0.326	1.529 ± 0.085	**0.576 ± 0.044**	**0.546 ± 0.025**	5.656 ± 1.053	2.012 ± 0.186	**0.496 ± 0.008**	**0.482 ± 0.002**	1.074	0.761	1.546 ± 0.378	0.944 ± 0.141	**0.411 ± 0.003**	**0.445 ± 0.002**
	960	.	.	2.972 ± 0.183	1.441 ± 0.035	**0.600 ± 0.033**	**0.570 ± 0.018**	6.408 ± 2.039	2.077 ± 0.242	**0.471 ± 0.015**	**0.481 ± 0.008**	.	.	1.862 ± 0.153	1.066 ± 0.055	**0.438 ± 0.007**	**0.462 ± 0.004**
$ETTm_1$	24	0.323	0.369	0.330 ± 0.021	0.382 ± 0.017	**0.309 ± 0.020**	**0.352 ± 0.010**	0.443 ± 0.043	0.437 ± 0.035	**0.403 ± 0.006**	**0.392 ± 0.005**	0.127	0.226	0.130 ± 0.003	0.231 ± 0.003	**0.106 ± 0.002**	**0.196 ± 0.001**
	48	0.494	0.503	0.499 ± 0.024	0.486 ± 0.012	**0.390 ± 0.008**	**0.391 ± 0.006**	0.453 ± 0.034	0.472 ± 0.018	**0.328 ± 0.010**	**0.371 ± 0.007**	0.150	0.261	0.155 ± 0.004	0.262 ± 0.004	**0.135 ± 0.003**	**0.222 ± 0.002**
	96	0.678	0.614	0.605 ± 0.033	0.554 ± 0.027	**0.405 ± 0.013**	**0.411 ± 0.006**	0.603 ± 0.051	0.581 ± 0.027	**0.379 ± 0.011**	**0.406 ± 0.007**	0.190	0.291	0.195 ± 0.012	0.291 ± 0.013	**0.162 ± 0.001**	**0.247 ± 0.001**
	288	1.056	0.786	0.906 ± 0.039	0.738 ± 0.028	**0.563 ± 0.024**	**0.502 ± 0.015**	0.849 ± 0.095	0.702 ± 0.051	**0.451 ± 0.016**	**0.445 ± 0.008**	0.417	0.462	0.361 ± 0.008	0.419 ± 0.004	**0.265 ± 0.003**	**0.321 ± 0.002**
	672	1.192	0.926	0.943 ± 0.062	0.760 ± 0.034	**0.663 ± 0.082**	**0.550 ± 0.031**	0.860 ± 0.057	0.726 ± 0.026	**0.555 ± 0.011**	**0.511 ± 0.008**	0.554	0.527	1.020 ± 0.040	0.756 ± 0.025	**0.357 ± 0.004**	**0.380 ± 0.002**
	1344	.	.	1.095 ± 0.065	0.823 ± 0.040	**0.824 ± 0.039**	**0.632 ± 0.019**	14.613 ± 26.108	1.948 ± 1.655	**0.631 ± 0.061**	**0.556 ± 0.020**	.	.	1.841 ± 0.242	1.044 ± 0.100	**0.412 ± 0.008**	**0.422 ± 0.003**
ECL	24	.	.	0.250 ± 0.005	0.358 ± 0.005	**0.148 ± 0.001**	**0.257 ± 0.001**	0.279 ± 0.007	0.372 ± 0.003	**0.176 ± 0.002**	**0.285 ± 0.001**	.	.	0.138 ± 0.004	0.246 ± 0.005	**0.112 ± 0.001**	**0.207 ± 0.001**
	48	0.344	0.393	0.300 ± 0.010	0.386 ± 0.005	**0.171 ± 0.004**	**0.279 ± 0.003**	0.309 ± 0.007	0.388 ± 0.004	**0.194 ± 0.001**	**0.301 ± 0.001**	.	.	0.163 ± 0.007	0.265 ± 0.007	**0.126 ± 0.001**	**0.222 ± 0.001**
	168	0.368	0.424	0.345 ± 0.012	0.423 ± 0.010	**0.261 ± 0.010**	**0.354 ± 0.007**	0.333 ± 0.016	0.410 ± 0.012	**0.218 ± 0.002**	**0.320 ± 0.001**	.	.	0.177 ± 0.003	0.281 ± 0.005	**0.153 ± 0.003**	**0.249 ± 0.002**
	336	0.381	0.431	0.429 ± 0.041	0.473 ± 0.023	**0.356 ± 0.026**	**0.414 ± 0.015**	0.326 ± 0.004	0.406 ± 0.001	**0.241 ± 0.005**	**0.337 ± 0.002**	.	.	0.202 ± 0.004	0.308 ± 0.004	**0.162 ± 0.001**	**0.262 ± 0.001**
	720	0.406	0.443	0.851 ± 0.088	0.719 ± 0.072	**0.834 ± 0.122**	**0.700 ± 0.076**	0.420 ± 0.094	0.467 ± 0.058	**0.303 ± 0.012**	**0.383 ± 0.011**	.	.	0.234 ± 0.006	0.333 ± 0.004	**0.183 ± 0.003**	**0.281 ± 0.002**
	960	0.460	0.548	0.930 ± 0.075	0.750 ± 0.042	**0.894 ± 0.047**	**0.741 ± 0.031**	0.399 ± 0.022	0.455 ± 0.017	**0.325 ± 0.019**	**0.398 ± 0.015**	.	.	0.235 ± 0.011	0.330 ± 0.008	**0.200 ± 0.003**	**0.292 ± 0.002**