REVERSIBLE INSTANCE NORMALIZATION FOR ACCURATE TIME-SERIES FORECASTING AGAINST DISTRIBUTION SHIFT

Anonymous authors
Paper under double-blind review

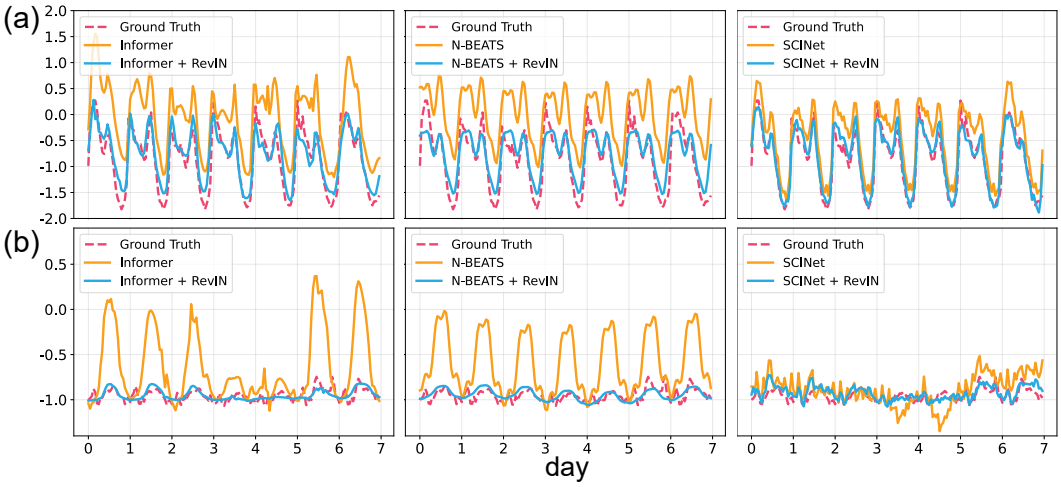

Figure 1: Multivariate time-series forecasting results of our method in comparison with the state-of-the-art baseline models, i.e., Informer (Zhou et al., 2021), N-BEATS (Oreshkin et al., 2020), and SCINet (Liu et al., 2021a). The analysis is conducted on one of the variables of the electricity counting 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 enginnering (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 im-

putation (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 provides 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 improving 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., 2021a). 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).

In summary, our contributions are as follows:

- We propose a simple and effective model-agnostic normalization method for time-series, called RevIN, which can be easily added to existing models with negligible cost and significantly improves the performance by reducing the distribution discrepancy within the train data, as well as the discrepancy between the train and test 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.

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., 2021b) 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, in-

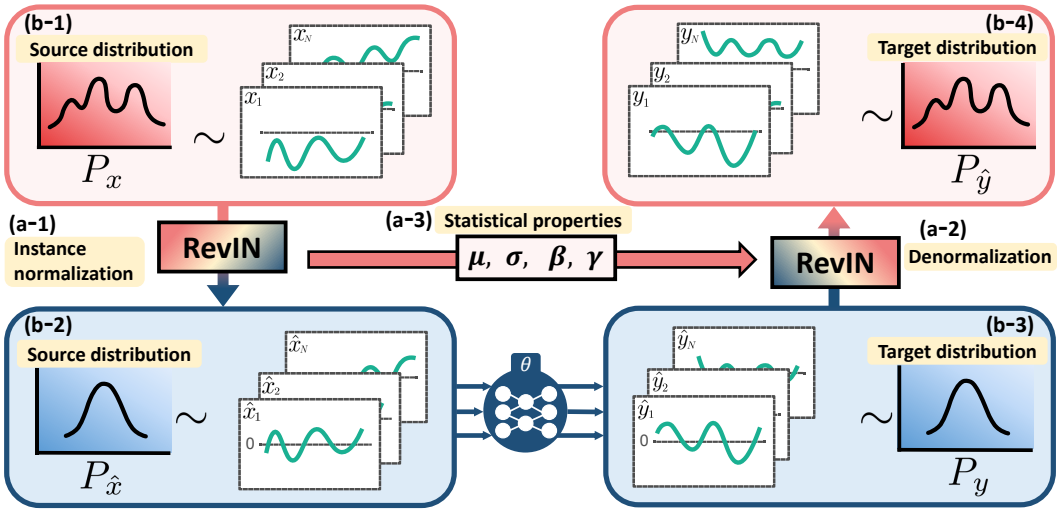

Figure 2: **Overview of the proposed method.** We present an example of an univariate case, where $x \in \mathbb{R}^{1\times T}$ (See Section 3.1). The input data x is actually multivariate. In RevIN, (a-1) instance normalization and (a-2) denormalization are symmetrically structured, applied to input 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 zero-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.

spired 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.) 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 RevIN (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. Given an input sequence $x \in \mathbb{R}^{K \times T_x}$, we aim to solve the time-series forecasting problem, which is to predict the subsequent values $y \in \mathbb{R}^{K \times T_y}$. Our proposed method, RevIN, symmetrically transforms the input data x and the prediction output y of the network, as illustrated in Fig. 2. First, we normalize the input data x with the instance-specific mean and standard deviation, which is widely accepted as instance normalization (Ulyanov et al., 2016a). The mean and standard deviation are computed for every instance $x_{k\cdot}$ of the input data (Fig. 2(a-3)) as

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

Using these statistics, we normalize the input data x as (Fig. 2(a-1))

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

where $\gamma, \beta \in \mathbb{R}^K$ are learnable affine parameter vectors. The normalized sequences have consistent statistical properties in terms of the mean and variance, where the contribution of the distribution shift to the values is reduced. As a result, the model only needs 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 $\hat{x}$ as input and forecasts their future values. However, the input data have different statistics from the original distribution, and only observing the normalized input $\hat{x}$, it is difficult to capture the original distribution of the input x. 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)). Accordingly, we denormalize the model output y by applying the reciprocal of the normalization in Eq. 2 (Fig. 2(a-2)) as

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

where the scaling and shifting variables are the exact instance statistics computed from input data x. Now, $\hat{y}$ is the final prediction of the model instead of y.

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 Fig. 4.2.3.

3.2 EFFECT OF REVERSIBLE INSTANCE NORMALIZATION ON DISTRIBUTION SHIFT PROBLEM.

Here, we verify 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 show 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

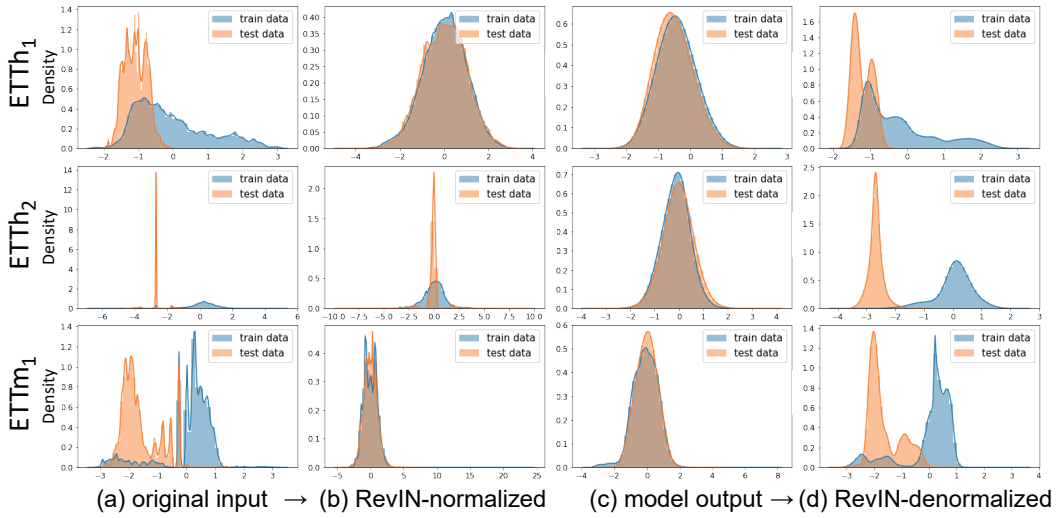

Figure 3: **Effect of RevIN on distribution discrepancy between train and test data.** From left to right columns, we compare the train and test data distributions on each step of the sequential process in RevIN: (a) the original input x, (b) the input normalized by RevIN, $\hat{x}$, (c) the model prediction output 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.

Fig. 3(a), the distributions of the train and test data hardly overlap, especially in 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 the zero-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 tests data distribution becomes 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 the 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(c)). 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 turn it back rather than directly solving the challenging problem where 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

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

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 preprocess the data to have an hourly basis following (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, 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.1.

Compared baselines RevIN is a model-agnostic method that can be easily added to any networks 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., 2021a). All the models forecast the time-series sequence non-autoregressively. Exact training details of the models are reported in Section A.2.

4.2 Results and Analysis

This section presents quantitative analyses and qualitative visualizations of RevIN when adopted to the state-of-the-art time-series forecasting model.

4.2.1 Effectiveness of Reversible Instance Normalization on Various Time-Series Forecasting Models

Table 1 demonstrates how much the proposed method improves the forecasting accuracy of the baseline models. We compare the MSE and MAE between RevIN and the baseline models, including Informer, N-BEATS, and SCINet, on ETT and ECL datasets with varying prediction lengths. RevIN and the baselines with the same hyperparameter settings, including the input and prediction lengths, are compared. We report the average value of the errors for five runs. Table 1 provides the average errors for five runs. Additionally, we report the standard deviation of all results in Appendix A.5.3 When adopted to various baselines, RevIN consistently outperforms the baselines with a large margin, achieving state-of-the-art performance. Moreover, RevIN shows significantly better results than baselines when forecasting long sequences that are difficult to predict. Specifically, for N-BEATS, the performance of the baseline on $ETTh_2$ dataset degrades significantly from 0.403 (at 24) to 6.408 (at 960) with increasing prediction length, while RevIN performs well consistently, showing the MSE of 0.192 for short sequence prediction (at 24) and 0.471 for long sequence prediction (at 960). Such tendency appears similarly on other datasets and models. The results indicate that RevIN is a model-agnostic method that can be easily applied to existing models and improve the model performance.

We further analyze the effect of RevIN on the long prediction sequence quantitatively. Table 2 shows an error of forecasting long future sequences with the same input sequence length. RevIN consistently outperforms the baselines in forecasting long sequences using relatively short input sequences with the length of 48. Although the difference in the performance of SCINet and RevIN is not large when predicting short sequence (e.g., 48), RevIN surpasses SCINet with a significant margin when predicting long sequences (e.g., 336, 720, and 960). These results show that our method works robustly to the prediction length.

To further study why RevIN is robust to the prediction length, we analyze the forecasting error for each time step [3]. Fig. 4 provides the results on $ETTh_2$ dataset for the baseline and RevIN using Informer, N-BEATS, and SCINet. For N-BEATS and SCINET, the error gradually increases for predicting the distant future. RevIN alleviates such a significant increase of error, surpassing them

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

[3] The error at t-th time step is calculated as $MSE_t = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}(\hat{y}_{kt}^{(i)} - y_{kt}^{(i)})^2$.

Table 1: **Comparison of MSE and MAE with different baselines and RevIN.** The forecasting errors of baseline and RevIN are compared on four datasets according to various baselines and prediction sequence length. We report the average value of five experiments.

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 performance of the baselines and RevIN by increasing the prediction length from 48 to 960 with the input sequence length fixed to 48. The experiment is conducted on $ETTh_1$ for five times and the average errors of five runs are reported. Adding RevIN to the baseline consistently decreases both MSE and MAE.

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**

with a large margin. For Informer, the error is relatively constant compared to the other models regardless of the prediction time step. However, the average error is still large compared to RevIN. The results show the effectiveness of RevIN when forecasting long sequence.

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., 2016b), and deep adaptive input normalization (DAIN) (Passalis et al., 2019) as shown in Table 3. The normalization parameters

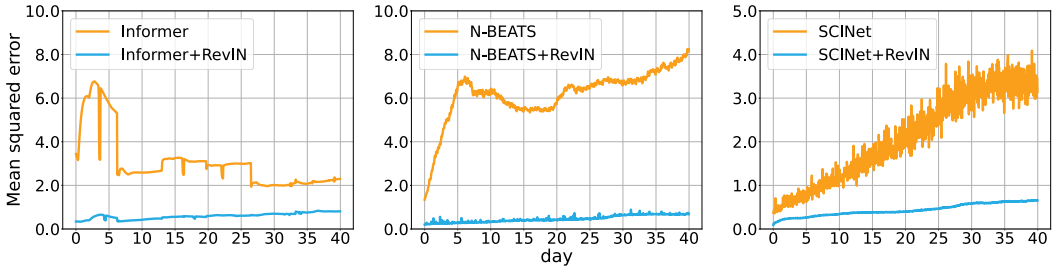

Figure 4: **Forecasting error for each time step.** The foreacsting error for each time step is compared on $ETTh_2$ dataset for the baseline and RevIN using Informer, N-BEATS, and SCINet. We report the cases with the prediction length of 960. Our method shows a lower error than the baselines.

Table 3: **Comparison with classical and state-of-the-art normalization methods.** The MSE is measured on the four datasets using N-BEATS as the baseline for all experiments. Here, min-max and z-score normalization methods are applied to each input instance. **RevBN** is the modified version of RevIN, where the input normalization is replaced with the batch normalization. DAIN, deep adaptive input normalization (Passalis et al., 2019); **RevBN**, the reversible 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**

for min-max and z-score normalization methods are calculated for each input instance. Additionally, we the batch normalization with the denormalization step of RevIN (reversible batch normalization, RevBN). In the case of layer normalization, its standardization operation cannot be reversed, since the input and prediction sequence length differ in our experimental settings. The MSE is measured on the four datasets using N-BEATS as the baseline for all experiments. Additional results with other prediction lengths are included in Table 5. RevIN shows outstanding performance compared to other normalization methods, especially for $ETTh_2$ and $ETTm_1$ datasets. Moreover, RevBN shows improved performance especially when predicting long sequence (960, 1344), compared to the Batch normalization. It describes that the denormalization step of RevIN is essential for the input normalization techniques tailored to time-series forecasting. However, since batch normalization applies an identical normalization to all the input sequences using train data statistics, it can not reduce the discrepancy between the datasets. As a result, RevIN surpasses RevBN with a large margin, which implies that our method successfully reduces the distribution discrepancy within data. Additionally, RevIN not only improves the performance of baselines but also has the advantage of being lightweight, only requiring $2K$ additional parameters, which is almost negligible compared to the number of parameters of baseline models.

4.2.3 Analysis on Distribution Shift In the Intermediate Layers

To study whether RevIN also can reduce the distribution shift at an intermediate feature level, we analyze the feature divergence between train and test datasets for baseline and RevIN on ETT and ECL datasets, using Informer as the baseline. Following (Pan et al., 2018), we calculate the average feature divergence of each feature between train and test datasets using symmetric KL divergence.

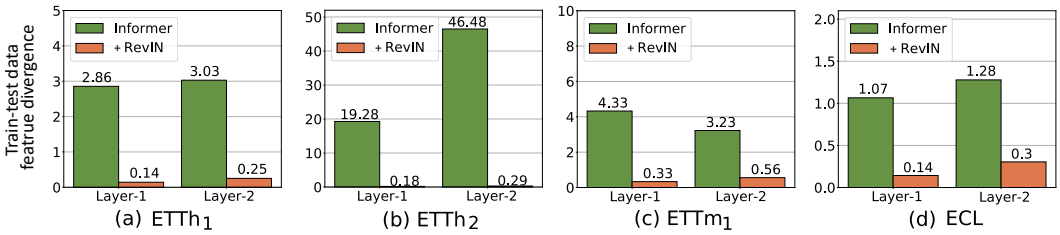

Figure 5: **Feature divergence at intermediate layers of the model, where the y-axis represents the feature divergence between train and test datasets.** The feature divergences are calculated on ETT and ECL datasets from the output of encoder layers of Informer. Encoder-1 and Encoder-2 indicate the first and second encoder layers, respectively.

The detailed calculation of symmetric KL divergence is described in Appendix A.4. The analysis results are shown in Fig. 5. Since Informer consists of two layers for the encoder and one layer for the decoder, we calculate the feature divergence of the intermediate outputs, the first encoder layer (Layer-1) output and the second encoder layer (Layer-2) output. As shown in the Fig. 5, the feature divergence between the train and test data significantly is reduced with RevIN. It implies that the normalization applied to the input can resolve the distribution shift in every layer in a model. Therefore, as a model-agnostic method, our method successfully alleviates the distribution shift problem via normalization and denormalization of the input and output without modifying the model architectures.

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 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 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.1 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.2 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 *International Conference on Machine Learning*, 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.

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.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proc. 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*, 2021a.

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*, 2021b.

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

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *Proc. 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. 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5956–5963, 2020.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proc. 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*, 2016a.

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

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 *Thirty-first AAAI conference on artificial intelligence*, 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 Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

A Appendix

The supplementary material presents additional information, visualizations, and experimental results that support the main manuscript. Section A.1 describes additional implementation details with the hyperparameter settings. Section A.2 explains the reproduction details for baselines where RevIN is applied. Section A.3 and Section A.4 provide the algorithm of RevIN and the calculation details of the feature divergence used in Section 4.2.3. Lastly, Section A.5 provides additional quantitative and qualitative results that can not be located in the manuscript due to the lack of space.

A.1 Experimental Details

We train N-BEATS with the weight decay with the factor of 0.001 to stabilize training. In the case of Information and SCINet, the training details provided in their official code are used. 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.

A.2 Reproduction Details for Baseline Models

This section describes the implementation details of baselines. Note that we compare baseline and RevIN using the same hyperparameters except for the presence of RevIN.

Informer. We use the official open-source code of Informer [4]. We follow the provided hyperparameters for the experiments they studied, and we tune only the input length to obtain optimal performances. The input length is chosen from $\{48, 96, 168, 336, 360, 480, 720\}$ for the ETTh_1, ETTh_2, and ECL datasets and from $\{48, 96, 336, 672\}$ for the ETTm_1 dataset. We use our hyperparameter settings for additional experiments that the authors did not provide hyperparameters: ETTh_1 with prediction length of 960, ETTh_2 with prediction length of 960, ETTm_1 with prediction length of 1344, and all experiments on the ECL dataset.

N-BEATS. N-BEATS is reproduced using the PyTorch framework (Paszke et al., 2019). We follow the hyperparameter settings of N-BEATS-I. We train N-BEATS to minimize the mean squared error between the predicted results and ground truth values. Only the results for a single model without an ensemble method are reported. Since N-BEATS is the model tailored to univariate time-series forecasting, we flatten the multivariate input sequence to have a single dimension before feed it to the model.

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

SCINet. For most of the experiments, we follow the experimental settings provided in the SCINet official code [5]. Similar to Informer experiments, we use our hyperparameters for experimental settings that SCINet did not study. More details of the hyperparameter settings for SCINet are shown in supplementary code.

A.3 ALGORITHM OF REVERSIBLE INSTANCE NORMALIZATION

Algorithm 10 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 10 (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.

Algorithm 1: RevIN, applied to model input x and output y.

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

Output: γ, β, θ.

1 **for** $m = 0, 1, 2 \ldots, e$ **do**

2 *Compute* $\mu_{\mathcal{T}} \leftarrow \frac{1}{T_x} \sum_{j=1}^{T_x} x_{kj}^{(i)}$ ▷ instance mean

3 *Compute* $\sigma_{\mathcal{T}}^2 \leftarrow \frac{1}{T_x} \sum_{j=1}^{T_x} (x_{kj}^{(i)} - \mu_{\mathcal{T}})^2$ ▷ instance variance

4 *Normalize* $\hat{x}_{kt}^{(i)} \leftarrow \frac{x_{kt}^{(i)} - \mu_{\mathcal{T}}}{\sqrt{\sigma_{\mathcal{T}}^2 + \epsilon}}$ ▷ normalization

5 *Transform* $\hat{x}_{kt}^{(i)} \leftarrow \gamma_{ik} \cdot \hat{x}_{kt}^{(i)} + \beta_{ik} \equiv \mathbf{RevIN}_{\gamma,\beta}^{\mathrm{n}}(x_{kt}^{(i)})$ ▷ scale and shift

6 *Predict* $y \leftarrow g_\theta(\hat{x})$ ▷ model prediction

7 *Retransform* $\hat{y}_{kt}^{(i)} \leftarrow \frac{y_{kt}^{(i)} - \beta_{ik}}{\gamma_{ik}}$ ▷ reverse scale and shift

8 *Denormalize* $\hat{y}_{kt}^{(i)} \leftarrow \mu_{\mathcal{T}} + \hat{y}_{kt}^{(i)} \sqrt{\sigma_{\mathcal{T}}^2 + \epsilon} \equiv \mathbf{RevIN}_{\gamma,\beta}^{\mathrm{dn}}(y_{kt}^{(i)})$ ▷ denormalization

9 *Update γ, β, θ based on the final prediction $\hat{y}$*

10 **end**

A.4 CALCULATION DETAILS ON FEATURE DIVERGENCE

This section explains how the feature divergence in Section 4.2.3 is calculated. Following (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

$$D(f_k^{\text{train}} || f_k^{\text{test}}) = KL(f_k^{\text{train}} || f_k^{\text{test}}) + KL(f_k^{\text{test}} || f_k^{\text{train}}), \tag{4}$$

$$\text{where} \quad KL(f_k^A || f_k^B) = \log \frac{\sigma_k^B}{\sigma_k^A} + \frac{{\sigma_k^A}^2 + (\mu_k^A - \mu_k^B)^2}{2{\sigma_k^B}^2} - \frac{1}{2}. \tag{5}$$

A.5 ADDITIONAL RESULTS

A.5.1 CROSS-DATASET EVALUATION

We analyze the cross-dataset evaluation results to analyze further the capability of RevIN to alleviate cross-dataset distribution discrepancy. As RevIN reduces the discrepancy in the train and test distributions, it can also reduce the discrepancy in the cross-dataset distributions if the input

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

Table 4: **Cross-dataset evaluation results of RevIN.** The mean squared error for cross-dataset 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**

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 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 4 and variance in Table 6. Despite the difference between distributions of the datasets, RevIN shows superior performance in cross-dataset 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.5.2 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.5.3 Additional Quantitative Results

Table 5 shows the results with additional prediction lengths of comparison with other normalization methods. Table 6 and Table 7 show the standard deviation for five independent experiments of cross-dataset evaluation and for comparison on long sequence forecasting, respectively. Table 8 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.

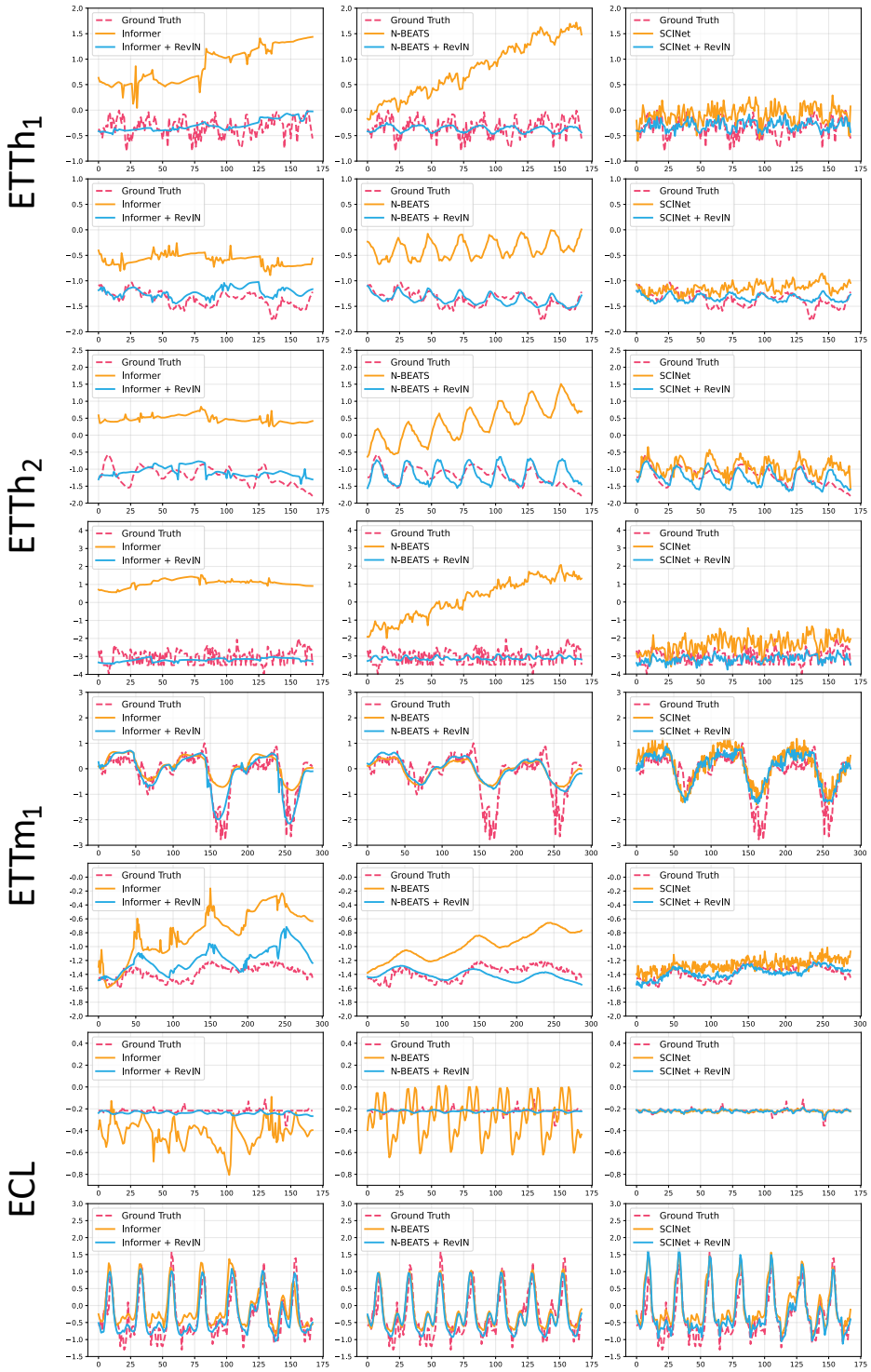

Figure 6: **Additional results on prediction comparison of RevIN and the existing SOTA models.** The models are evaluated on $ETTh_1$, $ETTh_2$ and ECL datsets, with the prediction length of 168. For $ETTm_1$ dataset, we evalute the model with prediction length of 288. Our method consistently improves the baseline models in terms of the scale and shift of the predicted sequence.

Table 5: **The additional comparison results with classical and state-of-the-art normalization methods.** The MSE is reported. L indicates the prediction length.

Dataset	L	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**

Table 6: **The stdandard deviation of the five runs for cross-dataset 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 7: **The stdandard 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 8: **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**