Time Series is a Special Sequence: Forecasting with Sample Convolution and Interaction

Minhao Liu[1], **Ailing Zeng**[1], **Qiuxia Lai**[1], **Qiang Xu**[1]
[1]The Chinese University of Hong Kong
{mhliu, alzeng, qxlai, qxu}@cse.cuhk.edu.hk

Abstract

Time series is a special type of sequence data, a set of observations collected at even intervals of time and ordered chronologically. Existing deep learning techniques use generic sequence models (e.g., recurrent neural network, Transformer model, or temporal convolutional network) for time series analysis, which ignore some of its unique properties. For example, the downsampling of time series data often preserves most of the information in the data, while this is not true for general sequence data such as text sequence and DNA sequence.

Motivated by the above, in this paper, we propose a novel neural network architecture and apply it for the time series forecasting problem, wherein we conduct sample convolution and interaction at multiple resolutions for temporal modeling. The proposed architecture, namely **SCINet**, facilitates extracting features with enhanced predictability. Experimental results show that SCINet achieves significant prediction accuracy improvement over existing solutions across various real-world time series forecasting datasets. In particular, it can achieve high forecasting accuracy for those temporal-spatial datasets without using sophisticated spatial modeling techniques. Our codes and data are presented in the supplemental material.

1 Introduction

Time series forecasting (TSF) enables decision-making with the estimated future evolution of metrics or events and thus plays a crucial role in various scientific and engineering fields such as healthcare [1], energy management [46], traffic flow [46] and financial investment [12] to name a few. Traditional time series forecasting methods such as auto regressive integrated moving average (ARIMA) model [8] and Holt-Winters seasonal method [16] have theoretical guarantees, but they are mainly applicable for univariate forecasting problem and require time series to be stationary, significantly restricting their applications to real-world complicated time series data. With the increasing data availability and computing power in recent years, it is shown that deep learning-based TSF techniques can achieve much better prediction accuracy than conventional approaches [24].

There are mainly three kinds of deep neural networks used for sequence modeling in the literature and they are all applied for time series forecasting [24]: (i). recurrent neural networks (RNNs) and their variants such as long short-term memory (LSTM) [15] and gated recurrent units (GRUs) [10]; (ii). Transformer [41] and its variants [20, 22, 46]; (iii). temporal convolutional networks (TCNs) [40, 4]. Generally speaking, TCNs are shown to be more effective and efficient in modeling time series data among these models. Moreover, it has been combined with graph neural networks (GNNs) to solve various temporal-spatial TSF problems [23, 44, 39].

TCNs perform dilated causal convolutions, which are designed based upon two principles: the fact that the network produces an output of the same length as the input, and the fact that there can be no leakage from the future into the past [4]. For TSF problems, we argue that the above two design

Preprint. Under review.

principles are in fact unnecessary and hence should be discarded. More importantly, time series is a special type of sequence data. For example, the downsampling of time series data often preserves most of the information in the data, while this is not true for general sequence data such as text sequence and DNA sequence. Existing TSF works, however, simply apply generic sequence models without taking the unique properties of time series into consideration.

Motivated by the above, in this paper, we propose a novel neural network architecture, *sample convolution and interaction network* (*SCINet*), that is specially designed for time series forecasting. The contributions of our work are as follows:

- We discover the misconception of existing TCN design principles for the TSF problem. In particular, we show causal convolution is **NOT** necessary and better prediction accuracy can be achieved by removing such constraints.
- We propose a hierarchical TSF framework, *SCINet*, based on the unique attributes of time series data. By iteratively extracting and exchanging information at different temporal resolutions, an effective representation with enhanced predictability can be learned, as verified by its comparatively lower permutation entropy (PE) [18].
- We design the basic building block, *SCI-Block*, for constructing SCINet, which downsamples the input data/feature into two sub-sequences, and then extracts features of each sub-sequence using distinct convolutional filters to preserve the heterogeneity information. To compensate for the information loss during the downsampling procedure, we incorporate interactive learning between the two convolutional features within each SCI-Block.

Extensive experiments on various real-world TSF datasets show that our model consistently outperforms state-of-the-art methods by a considerable margin. In particular, SCINet can achieve high forecasting accuracy for those temporal-spatial datasets without using sophisticated GNN-based spatial modeling techniques.

2 Related Work and Motivation

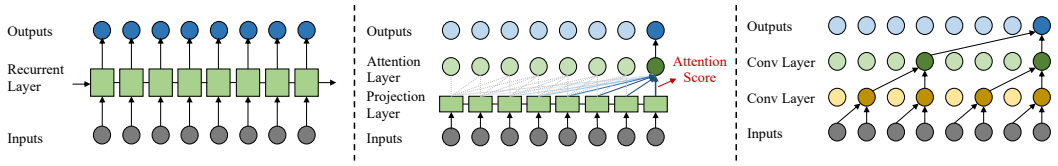
Time series forecasting is to predict the values of the time series in future time-steps from historical observations, and it can be roughly classified into single-step and multi-step forecasting [43]. Given a long time series $\mathbf{X}^*$ and a look-back window of fixed length T, at timestamp t, single-step forecasting is to predict the future value $\hat{\mathbf{X}}_{t+\tau:t+\tau} = \{\mathbf{x}_{t+\tau}\}$, while multi-step forecasting is to predict multiple future values $\hat{\mathbf{X}}_{t+1:t+\tau} = \{\mathbf{x}_{t+1}, ..., \mathbf{x}_{t+\tau}\}$ based on the past T steps $\mathbf{X}_{t-T+1:t} = \{\mathbf{x}_{t-T+1}, ..., \mathbf{x}_t\}$. Here, τ is the length of the forecast horizon, $\mathbf{x}_t \in \mathbb{R}^d$ is the value at time step t, and d is the number of time-series. For simplicity, in the following we will omit the subscripts, and use $\mathbf{X}$ and $\hat{\mathbf{X}}$ to represent the historical data and the prediction, respectively. For multi-step forecasting, we can either directly optimize the multi-step forecasting objective (direct multi-step (DMS) estimation), or learn a single-step forecaster and iteratively applies it to get multi-step predictions (iterated multi-step (IMS) estimation) [37].

In this section, we focus on deep learning-based methods for TSF due to their superior performance over traditional approaches such as ARIMA.

2.1 Deep Learning-Based Time Series Forecasting

RNN-based TSF methods [33, 35] summarize the past information compactly in the internal memory state for prediction, where the memory state is recursively updated with new inputs at each time step, as shown in Fig. 1 (a). These methods generally belong to IMS estimation methods, which suffer from error accumulation. The gradient vanishing/exploding problems and the memory constraint also greatly restrict the application scenarios.

Transformers have taken the place of RNN models in almost all sequence modeling tasks, thanks to their training efficiency and the effectiveness of self-attention mechanisms. Consequently, various Transformer-based TSF methods (see Fig. 1 (b)) are proposed in the literature [22, 25]. and they are shown to be quite effective in predicting long sequence. However, the overhead of Transformer-based models is a serious concern and lots of research efforts are dedicated to tackle this problem (e.g., [46]).



(a) RNN model (b) Transformer model (c) TCN model

Figure 1: Existing sequence modeling architectures.

Convolutional models [7, 4] have become a popular choice in various TSF problems [36, 42, 31]. This is because, the parallel convolution operation of multiple filters (Fig. 1 (c)) allows for fast data processing and efficient dependencies learning in and between the multivariate time series compared with RNN- and Transformer-based methods.

The proposed SCINet is also constructed based on temporal convolution. However, our method has several key differences compared with existing models based on *dilated causal convolution*. In Section 2.2, we highlight our insights and motivate this work.

2.2 Rethinking Dilated Causal Convolution for Time Series Forecasting

Dilated causal convolution was first used in WaveNet [40] for generating raw audio waveforms. Later, Bai et al. [4] simplify the WaveNet architecture to the so-called temporal convolutional networks (TCNs) for sequence modeling. Fig. 1 (c) shows the typical TCN architecture, which consists of a stack of causal convolutional layers with exponentially enlarged dilation factors. Over the years, TCNs have been widely used in all kinds of time series forecasting problems and achieved promising results [42, 36, 44].

TCNs are designed based upon the following two principles:

- the network produces an output of the same length as the input, which is realized by using a 1D fully-convolutional network (FCN) architecture with zero padding for each hidden layer.
- there can be no leakage from the future into the past, wherein an output i is convolved only with the i^{th} and earlier elements in the previous layer.

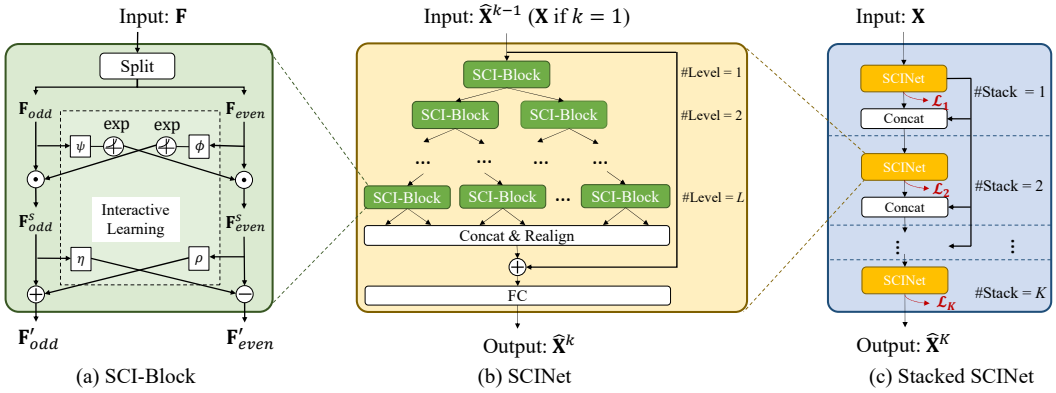
We argue that the above two design principles are in fact **unnecessary** for TSF. For the first design principle, recall that the TSF problem is to predict some future values with a given look-back window. Consequently, there is no reason to force an equivalent length between the network input and output. This is also a common practice in other sequence modeling architectures such as Transformer. For the second design principle, while we agree that causality should be considered in forecasting tasks, such information leakage problem exists only when the output and the input have temporal overlaps, e.g., in auto-regressive prediction where the previous output serves as the input for future prediction. However, when the predictions are completely based on the known inputs in the look-back window, there is no need to use causal convolution under such circumstances and we can safely apply *normal convolution* that utilizes all the historical information in the look-back window for prediction.

A simple causal convolution is only able to look-back at a timing window with size linear in the depth of the network. To alleviate this problem, TCNs use *dilated convolutions* to achieve a large receptive field with fewer convolutional layers. However, a single convolutional filter is shared across a layer in TCNs, restricting the extraction of temporal dynamics from the data/feature in the previous layer.

In view of the satisfactory performance of TCNs for various TSF problems despite the above long-standing misconceptions, we propose a novel convolutional architecture *SCINet* that lifts the unnecessary constraints of TCNs. More importantly, SCINet takes the unique properties of time series into consideration to achieve better prediction accuracy, as detailed in the following section.

3 SCINet: Sample Convolution and Interaction Networks

The proposed *Sample Convolution and Interaction Networks* (*SCINet*) is a hierarchical framework that enhances the predictability of the original time series by capturing temporal dependencies at multiple



(a) SCI-Block (b) SCINet (c) Stacked SCINet

Figure 2: The overall architecture of Sample Convolution and Interaction Network (SCINet).

temporal resolutions[1]. As shown in Fig. 2 (a), the basic building block, *SCI-Block* (Section 3.1), downsamples the input data or feature into two sub-sequences, and then process each sub-sequence with a set of distinct convolutional filters to well preserve the heterogeneity information of each part. To compensate the information loss during downsampling, we incorporate *interactive learning* between the two sub-sequences. Our *SCINet* (Section 3.2) is constructed by arranging multiple SCI-Blocks into a binary tree structure (Fig. 2 (b)). After all the downsample-convolve-interact operations, we realign and concatenate all the low-resolution components into a new sequence representation, and add it to the original time series for forecasting. To facilitate extracting complicated temporal dynamics, we could further stack multiple SCINets and apply intermediate supervision to get a *Stacked SCINet* (Section 3.3), as shown in Fig. 2 (c).

3.1 SCI-Block

The SCI-Block (Fig. 2(a)) is the basic module of the SCINet, which decomposes the input feature $\mathbf{F}$ into two sub-features $\mathbf{F}^{'}_{odd}$ and $\mathbf{F}^{'}_{even}$ through the operations of *Spliting* and *Interactive-learning*.

Splitting is to downsample the original sequence $\mathbf{F}$ into two low-resolution sub-sequences $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$ by separating the even and the odd elements. Each sub-sequence is of coarser temporal resolution compared with the original input, and only contains part of the original information. To better preserve the original information, we keep both sub-sequences for feature extraction. This is different from previous practice in multi-scale analysis where only a single sub-sequence at a certain temporal resolution is used for feature extraction. Moreover, no domain knowledge is required to manually design the downsampling rates. Thus the SCI-Block can be easily generalized to various time series data. Considering the heterogeneity information of each downsampled sub-sequence, we propose to process $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$ using two distinct sets of convolution layers.

To further compensate the information at each sub-sequence, after extracting features from $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$ using distinct convolution layers, we propose a novel *interactive-learning* strategy to allow information interchange between the two sub-features. As shown in Fig. 2(a), the interactive learning updates the two sub-sequences by learning the parameters of affine transformation from each other, which can greatly enhance the representation ability of the learned features by coupling the information of the two complementary parts. The interactive learning procedure consists of two steps. First, as shown in Eq. (1), $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$ are projected to hidden states with two different 1D convolutional modules ϕ and ψ, respectively, and transformed to the formats of $\exp$ and interact to the $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$ with the element-wise product. This can be viewed as performing scaling transformation on $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$, where the scaling factors are learned from each other using neural network modules. Here, $\odot$ is the Hadamard product or element-wise production.

$$\mathbf{F}^{s}_{odd} = \mathbf{F}_{odd} \odot \exp(\phi(\mathbf{F}_{even})), \quad \mathbf{F}^{s}_{even} = \mathbf{F}_{even} \odot \exp(\psi(\mathbf{F}_{odd})). \tag{1}$$

$$\mathbf{F}^{'}_{odd} = \mathbf{F}^{s}_{odd} \pm \rho(\mathbf{F}^{s}_{even}), \quad \mathbf{F}^{'}_{even} = \mathbf{F}^{s}_{even} \mp \eta(\mathbf{F}^{s}_{odd}). \tag{2}$$

[1] *Multi-resolution analysis* is a classic method for signal analysis [28] and has been used in time series classification tasks [11].

Second, as shown in Eq. (2), the two scaled features $\mathbf{F}^s_{even}$ and $\mathbf{F}^s_{odd}$ are further projected to another two hidden states with the other two 1D convolutional modules ρ and η, and then be added to or subtracted from $\mathbf{F}^s_{even}$ and $\mathbf{F}^s_{odd}$. The final outputs of the interactive learning module are two updated sub-features $\mathbf{F}^{'}_{even}$ and $\mathbf{F}^{'}_{odd}$. The default architectures of ϕ, ψ, ρ and η are shown in the Appendix.

3.2 SCINet

With the SCI-Blocks presented above, we construct the SCINet by arranging multiple SCI-Blocks hierarchically and get a tree-structured framework, as shown in Fig. 2 (b). There are 2^l SCI-Blocks at the l-th level, where $l = 1, \ldots, L$ is the index of the level, and L is the total level number. For the k-th SCINet in the stacked SCINet, the input time series $\mathbf{X}$ (for $k = 1$) or feature vector $\hat{\mathbf{X}}^{k-1} = \{\hat{\mathbf{x}}^{k-1}_1, ..., \hat{\mathbf{x}}^{k-1}_\tau\}$ (for $k > 1$) is gradually down-sampled and processed by SCI-Blocks through different levels, which allows for effective feature learning of different temporal resolutions. This is because the temporal resolution of the features gradually decreases when l increases. In particular, the information from previous levels will be gradually accumulated, i.e. the features of the deeper levels would contain extra finer-scale temporal information transmitted from the shallower levels. In this way, we can capture both short-term and long-term dependencies for TSF. After going through L levels of SCI-Blocks, we rearrange the elements in all the sub-features by reversing the odd-even splitting operation, and concatenate them into a new sequence representation, which is added to the original time-series for forecasting through a residual connection [14]. Finally, a fully-connected layer is used to decode the enhanced sequence representation into $\hat{\mathbf{X}}^k = \{\hat{\mathbf{x}}^k_1, ..., \hat{\mathbf{x}}^k_\tau\}$, which represents a fused feature or the final prediction depending on the location of current SCINet in the stacked SCINet. This fully-connected layer also enables an output of a different length with the input, as opposed to the TCN where the length of the output is forced to be equal to that of the input.

Although increasing L would introduce extra computational cost, we will show that SCI-Net is still more efficient than other convolution-based methods such as TCN. As shown in Fig. 1 (c), TCN employs convolutions with exponentially enlarged dilation factors. The number of convolution layers needed for TCN to cover a look-back window of length T is $\log_2 T$, and the resulted number of convolution operations is $T * \log_2 T$. On the contrary, SCINet does not rely on dilation to enlarge the effective receptive fields of feature extraction, thanks to the interactive learning which allows global information interchange between the two downsampled parts. Thus, we do not need $\log_2 T$ levels to cover the look-back window, and the number of convolution operations of SCINet is $L * 2T$, which can be much less than that of the TCN. As experimentally verified in Section 4.3, $L \leq 5$ would satisfy the requirements of most cases.

3.3 Stacked SCINet with Intermediate Supervision

When the length of the look-back window T is comparable with the length of the prediction horizon τ, it would be difficult for a single SCINet to fully capture the temporal dependencies. To fully accumulate the historical information within the look-back window, we can further stack K SCINets end-to-end to form the *Stacked SCINet* structure, as illustrated in Fig. 2 (c).

Specially, we apply *intermediate supervision* [5] on the output of each SCINet using the ground-truth, to ease the learning of intermediate temporal features. The output of the k-th intermediate SCINet, $\hat{\mathbf{X}}^k$, is concatenated with part of the input $\mathbf{X}_{t-(T-\tau)+1:t}$ and feeded as input into the $(k+1)$-th SCINet, where $k = 1, \ldots, K-1$, and K is the total number of the SCINets in the stacked structure. The output of the K-th SCINet, $\hat{\mathbf{X}}^K$, is the final prediction. Similar to the analysis on L, increasing K would also lead to higher computational cost. However, we will show in Section 4.3 that $K \leq 3$ would be sufficient for most scenarios, as shown .

3.4 Loss Function

To train a stacked SCINet with K ($K \geq 1$) SCINets, the loss of the k-th intermediate prediction is calculated as the L1 loss between the output of the k-th SCINet and the ground-truth horizontal window to be predicted:

$$\mathcal{L}_k = \frac{1}{\tau} \sum_{i=0}^{\tau} \left\| \hat{\mathbf{x}}^k_i - \mathbf{x}_i \right\|, \quad k \neq K. \tag{3}$$

Table 1: The overall information of the 11 datasets.

Datasets	ETTh1	ETTh2	ETTm1	Traffic	Solar-Energy	Electricity	Exchange-Rate	PEMS03	PEMS04	PEMS07	PEMS08
# Variants	7	7	7	862	137	321	8	358	307	883	170
# Timesteps	17,420	17,420	69,680	17,544	52,560	26,304	7,588	26,209	16,992	28,224	17,856
Granularity	1hour	1hour	15min	1hour	10min	1hour	1day	5min	5min	5min	5min
Start time	7/1/2016	7/1/2016	7/1/2016	1/1/2015	1/1/2006	1/1/2012	1/1/1990	5/1/2012	7/1/2017	5/1/2017	3/1/2012
Task type	Multi-step	Multi-step	Multi-step	Single-step	Single-step	Single-step	Single-step	Multi-step	Multi-step	Multi-step	Multi-step
Data partition	Follow [46]			Training/Validation/Testing: 6/2/2				Training/Validation/Testing: 6/2/2			

The loss function of the last SCINet depends on the type of forecasting tasks. For multi-step forecasting, the loss $\mathcal{L}_K$ is the same as Eq. (3). For single-step forecasting, we introduce a balancing parameter $\lambda \in (0, 1)$ for the value of the last time-step[2], which will be discussed in Appendix:

$$\mathcal{L}_K = \frac{1}{\tau - 1} \sum_{i=0}^{\tau-1} \left\| \hat{\mathbf{x}}_i^K - \mathbf{x}_i \right\| + \lambda \left\| \hat{\mathbf{x}}_\tau^K - \mathbf{x}_\tau \right\|. \tag{4}$$

The total loss of the stacked SCINet can be written as:

$$\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k. \tag{5}$$

4 Experiments

In Section 4.1, we first briefly introduce the 11 datasets on which we train the model and make comparisons with other models. In Section 4.2, we show the quantitative and qualitative comparisons with the state-of-the-arts, and analyse the predictability of our method. Finally, a comprehensive ablation study is conducted in Section 4.3 to assess the effectiveness of different components. More details on *evaluation metrics*, *data pre-processing*, *experimental settings* and *network structure* & *hyper-parameter tuning* are shown in the Appendix.

4.1 Datasets

We conduct the experiments on 11 popular time-series datasets, namely *Electricity Transformer Temperature* (ETTh1, ETTh2 and ETTm1) [46], *Traffic*, *Solar-Energy*, *Electricity* and *Exchange-Rate* [21] and *PeMS* (PEMS03, PEMS04, PEMS07 and PEMS08) [9], ranging from power, energy, finance and traffic domains. A brief description of these datasets is listed in Table 1. To make a fair comparison with state-of-the-art models reported in the corresponding datasets, we categorize the datasets into 3 groups and use the same evaluation metrics as the original publications in each group. For *ETT* datasets, we use the Mean Absolute Errors (MAE) [19] and Mean Squared Errors (MSE) [27]. For *Traffic*, *Solar-Energy*, *Electricity* and *Exchange-Rate*, we use Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) following [21]. As for *PeMS*, we use the MAE, Root Mean Squared Errors (RMSE) and Mean Absolute Percentage Errors (MAPE) following [19] . Lower values are better for all the metrics except for CORR. The evaluation is calculated on the last predicted value for single-step forecasting tasks, or averaged on all the predictions for multi-step ones.

4.2 Results and Analysis

ETT datasets [46] are used for assess the performance of long-sequence TSF tasks, and the results are shown in Table 2. As can be seen, compared with RNN-based methods such as LSTMa [2] and LSTnet [21], *Transformer* based methods [20, 22, 46] are better at capturing the long-term latent patterns in the entire historical data for predicting the future, thus suffer lower prediction errors. TCN-based methods [4, 43] further outperform Transformer-based methods, because the stacked convolution layers allow for fast data processing the efficient dependencies learning in and between the multivariate time series compared with RNN and Transformer based methods. Specially, TCN† denotes a variant of TCN where the causal convolution is replaced by normal convolution, and improves the original TCN across all the ETT datasets, which well support the

[2]This is slightly different from other practice for single-step forecasting [21], because we choose to use all the available values in the prediction window as supervision signal.

Table 2: Performance comparison of different approaches on *ETT* datasets. The best results are in **bold** and the second best results are underlined and marked with *orange* color. The IMP denotes the relative improvements of SCINet over the second best model. The same for other tables.

Methods	Metrics	ETTh1					ETTh2					ETTm1				
		Horizon					Horizon					Horizon				
		24	48	168	336	720	24	48	168	336	720	24	48	96	288	672
LogTrans [22]	MSE	0.686	0.766	1.002	1.362	1.397	0.828	1.806	4.070	3.875	3.913	0.419	0.507	0.768	1.462	1.669
	MAE	0.604	0.757	0.846	0.952	1.291	0.750	1.034	1.681	1.763	1.552	0.412	0.583	0.792	1.320	1.461
Reformer [20]	MSE	0.991	1.313	1.824	2.117	2.415	1.531	1.871	4.660	4.028	5.381	0.724	1.098	1.433	1.820	2.187
	MAE	0.754	0.906	1.138	1.280	1.520	1.613	1.735	1.846	1.688	2.015	0.607	0.777	0.945	1.094	1.232
LSTMa [2]	MSE	0.650	0.702	1.212	1.424	1.960	1.143	1.671	4.117	3.434	3.963	0.621	1.392	1.339	1.740	2.736
	MAE	0.624	0.675	0.867	0.994	1.322	0.813	1.221	1.674	1.549	1.788	0.629	0.939	0.913	1.124	1.555
LSTNet [21]	MSE	1.293	1.456	1.997	2.655	2.143	2.742	3.567	3.242	2.544	4.625	1.968	1.999	2.762	1.257	1.917
	MAE	0.901	0.960	1.214	1.369	1.380	1.457	1.687	2.513	2.591	3.709	L1700	1.215	1.542	2.076	2.941
Informer [46]	MSE	0.577	0.685	0.931	1.128	1.215	0.720	1.457	3.489	2.723	3.467	0.323	0.494	0.678	1.056	*1.192*
	MAE	0.549	0.625	0.752	0.873	0.896	0.665	1.001	1.515	1.340	1.473	0.369	0.503	0.614	0.786	*0.926*
*TCN	MSE	0.511	0.515	0.694	0.814	0.944	0.444	0.617	2.405	2.486	2.608	0.229	0.239	0.260	0.768	2.732
	MAE	0.549	0.529	0.617	0.682	0.778	0.478	0.615	1.266	1.312	1.276	0.282	0.360	0.363	0.646	1.371
*TCN†	MSE	*0.467*	*0.478*	*0.652*	*0.787*	*0.927*	*0.424*	*0.594*	*2.355*	*2.396*	*2.572*	*0.210*	*0.219*	*0.243*	*0.724*	2.136
	MAE	*0.477*	*0.492*	*0.606*	*0.675*	*0.770*	*0.465*	*0.589*	*1.168*	*1.214*	*1.241*	*0.280*	*0.316*	*0.343*	*0.633*	1.211
SCINet	MSE	**0.311**	**0.364**	**0.497**	**0.491**	**0.612**	**0.183**	**0.259**	**0.528**	**0.648**	**1.074**	**0.127**	**0.150**	**0.190**	**0.417**	**0.554**
	MAE	**0.348**	**0.388**	**0.491**	**0.494**	**0.582**	**0.271**	**0.341**	**0.509**	**0.608**	**0.761**	**0.226**	**0.261**	**0.291**	**0.462**	**0.527**
IMP	MSE	33.40%	23.85%	23.77%	37.61%	33.98%	56.84%	56.40%	77.58%	72.95%	58.24%	39.52%	31.51%	21.81%	42.40%	53.56%
	MAE	27.04%	21.14%	18.98%	26.81%	24.42%	41.72%	42.11%	56.42%	49.92%	38.68%	19.29%	17.41%	15.16%	27.01%	43.12%

∗ denotes re-implementation. † denotes the variant without causal convolution.

Table 3: Performance comparison of different approaches on *Traffic*, *Solar-Energy*, *Electricity* and *Exchange-Rate* datasets.

Methods	Metrics	Solar-Energy				Traffic				Electricity				Exchange-rate			
		Horizon				Horizon				Horizon				Horizon			
		3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
AR	RSE	0.2435	0.3790	0.5911	0.8699	0.5991	0.6218	0.6252	0.6300	0.0995	0.1035	0.1050	0.1054	0.0228	0.0279	0.0353	0.0445
	CORR	0.9710	0.9263	0.8107	0.5314	0.7752	0.7568	0.7544	0.7591	0.8845	0.8632	0.8691	0.8595	0.9734	0.9656	0.9526	0.9357
VARMLP [45]	RSE	0.1922	0.2679	0.4244	0.6841	0.5582	0.6579	0.6023	0.6146	0.1393	0.1620	0.1557	0.1274	0.0265	0.0394	0.0407	0.0578
	CORR	0.9829	0.9655	0.9058	0.7149	0.8245	0.7695	0.7929	0.7891	0.8708	0.8389	0.8192	0.8679	0.8609	0.8725	0.8280	0.7675
GP [34]	RSE	0.2259	0.3286	0.5200	0.7973	0.6082	0.6772	0.6406	05995	0.1500	0.1907	0.1621	0.1273	0.0239	0.0272	0.0394	0.0580
	CORR	0.9751	0.9448	0.8518	0.5971	0.7831	0.7406	0.7671	0.7909	0.8670	0.8334	0.8394	0.8818	0.8713	0.8193	0.8484	0.8278
RNN-GRU	RSE	0.1932	0.2628	0.4163	0.4852	0.5358	0.5522	0.5562	05633	0.1102	0.1144	0.1183	0.1295	0.0192	0.0264	0.0408	0.0626
	CORR	0.9823	0.9675	0.9150	0.8823	0.8511	0.8405	0.8345	0.8300	0.8597	0.8623	0.8472	0.8651	0.9786	0.9712	0.9531	0.9223
LSTNet [21]	RSE	0.1843	0.2559	0.3254	0.4643	0.4777	0.4893	0.4950	0.4973	0.0864	0.0931	0.1007	0.1007	0.0226	0.0280	0.0356	0.0449
	CORR	0.9843	0.9690	0.9467	0.8870	0.8721	0.8690	0.8614	0.8588	0.9283	0.9135	0.9077	0.9119	0.9735	0.9658	0.9511	0.9354
TPR-LSTM [38]	RSE	0.1803	0.2347	0.3234	0.4389	0.4487	0.4658	0.4641	0.4765	0.0823	0.0916	0.0964	0.1006	*0.0174*	*0.0241*	*0.0341*	*0.0444*
	CORR	0.9850	0.9742	0.9487	0.9081	0.8812	0.8717	0.8717	0.8629	0.9439	0.9337	0.9250	0.9133	*0.9790*	*0.9709*	*0.9564*	*0.9381*
*TCN	RSE	0.1940	0.2581	0.3512	0.4732	0.5459	0.6061	0.6367	0.6586	0.0892	0.0974	0.1053	0.1091	0.0217	0.0263	0.0393	0.0492
	CORR	0.9835	0.9602	0.9321	0.8812	0.8486	0.8205	0.8048	0.7921	0.9232	0.9121	0.9017	0.9101	0.9693	0.9633	0.9531	0.9223
*TCN†	RSE	0.1900	0.2382	0.3353	0.4676	0.5361	0.5992	0.6061	0.6456	0.0852	0.0924	0.0993	0.0989	0.0202	0.0257	0.0352	0.0487
	CORR	0.9848	0.9612	0.9432	0.8851	0.8540	0.8197	0.8205	0.7982	0.9293	0.9235	0.9173	0.9101	0.9712	0.9628	0.9501	0.9314
MTGNN	RSE	*0.1778*	*0.2348*	*0.3109*	*0.4270*	*0.4162*	*0.4754*	*0.4461*	*0.4535*	*0.0745*	*0.0878*	**0.0916**	**0.0953**	0.0194	0.0259	0.0349	0.0456
	CORR	*0.9852*	*0.9726*	*0.9509*	*0.9031*	*0.8963*	*0.8667*	*0.8794*	*0.8810*	*0.9474*	*0.9316*	*0.9278*	*0.9234*	0.9786	0.9708	0.9551	0.9372
SCINet	RSE	**0.1609**	**0.2194**	**0.2878**	**0.4032**	**0.3993**	**0.4365**	**0.4400**	**0.4406**	**0.0678**	**0.0818**	*0.0926*	*0.0957*	**0.0147**	**0.0224**	**0.0322**	**0.0426**
	CORR	**0.9934**	**0.9859**	**0.9741**	**0.9413**	**0.9038**	**0.8850**	**0.8810**	**0.8825**	**0.9559**	**0.9416**	**0.9327**	**0.9276**	**0.9868**	**0.9807**	**0.9719**	**0.9591**
IMP	RSE	9.51%	6.56%	7.43%	5.57%	4.06%	8.18%	1.37%	2.84%	8.99%	6.83%	-1.09%	-0.42%	15.52%	7.05%	5.57%	4.05%
	CORR	0.83%	1.37%	2.44%	4.23%	0.84%	2.11%	0.18%	0.17%	0.90%	1.07%	0.53%	0.45%	0.80%	1.01%	1.62%	2.24%

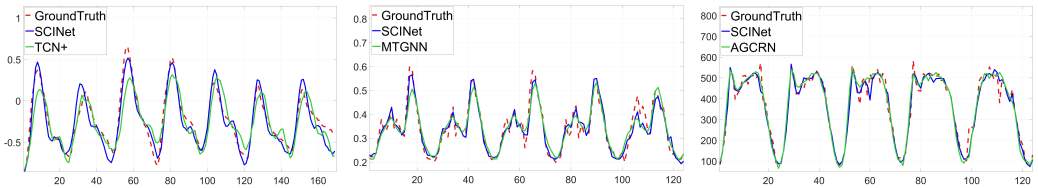
∗ denotes re-implementation. † denotes the variant without causal convolution.

statements presented in Section 2.2. The proposed SCINet outperforms all other competitors by a large margin. This is because (i) we use normal convolution instead of causal convolution to make full use of the information from the known input, and (ii) we capture temporal correlations at multiple temporal resolutions to enhance the predictability of the original time series. Thus, SCINet achieves considerably better forecasting accuracy on the newly published *ETT* datasets, which are less explored and have large room for improvement. A qualitative comparison is presented in Fig. 3(a).

Traffic, *Solar-Energy*, *Electricity* and *Exchange-Rate* are popular datasets for single-step forecasting [21]. As shown in Table. 3, our SCINet surpasses almost all other methods, including the statistical methods like VAR [45] and GP [34], and the conventional deep sequential learning models such as RNN-GRU, LSTNet, TPA-LSTM [38], TCN, and TCN†. We are slightly inferior than MTGNN [43] on *Electricity* when the length of the prediction horizon is 12 or 24. The qualitative comparison with MTGNN for horizon$=6$ is shown in Fig. 3 (b). The results clearly demonstrate the superiority of the proposed SCINet in exploring the temporal dependencies for single-step forecasting.

PeMS datasets [9] collect temporal-spatial time series from various scenarios. Most of the previous methods utilize GNNs to capture spatial correlations while modeling temporal dependencies using conventional TCN/LSTM. As shown in Table 4, the GNN-based methods [23, 44, 42, 39, 30, 3, 17] perform better than pure RNN or TCN based methods [15, 4]. However, our SCINet can achieve better performance on temporal-spatial forecasting without sophisticated spatial dependency modeling. This might be because that the strong temporal correlations among the data play a more important role than the spatial correlations for the forecasting. The visualization results in Fig. 3 (c) also show that SCINet captures more details than the second best model, AGCRN [3].

As introduced in Section 3.2, SCINet learns an enhanced sequence representation that is fed into the fully-connected layer to yield the final prediction. To figure out what benefits the enhanced



(a) ETTh1 (horizon=168) (b) Electricity (horizon=6) (c) PEMS08 (horizon=12)

Figure 3: The prediction comparison of SCINet with previous SOTAs on different datasets.

Table 4: Performance comparison of different approaches on *PeMS* datasets.

Datasets	Metrics	Methods												
		*LSTM	*TCN	*TCN†	DCRNN[23]	STGCN[44]	ASTGCN(r)[13]	GraphWaveNet[42]	STSGCN[39]	STFGNN[29]	AGCRN[3]	LSGCN[17]	SCINet	IMP
PEMS03	MAE	21.33	19.32	18.87	18.18	17.49	17.69	19.85	17.48	16.77	*15.98*	-	**15.11**	5.44%
	MAPE	21.33	19.93	18.63	18.91	17.15	19.40	19.31	16.78	16.30	*15.23*	-	**14.13**	7.22%
	RMSE	35.11	33.55	32.24	30.31	30.12	29.66	32.94	29.21	*26.28*	*28.25	-	**24.39**	7.19%
PEMS04	MAE	25.14	23.22	22.81	24.70	22.70	22.93	25.45	21.19	20.48	*19.83*	21.53	**19.02**	4.08%
	MAPE	20.33	15.59	14.31	17.12	14.59	16.56	17.29	13.90	16.77	*12.97*	13.18	**11.59**	10.64%
	RMSE	39.59	37.26	36.87	38.12	35.55	35.22	39.70	33.65	32.51	*32.30*	33.86	**31.12**	3.65%
PEMS07	MAE	29.98	32.72	30.53	28.30	25.38	28.05	26.85	24.26	23.46	*22.37*	-	**21.59**	3.49%
	MAPE	15.33	14.26	13.88	11.66	11.08	13.92	12.12	10.21	9.21	*9.12*	-	**8.79**	3.62%
	RMSE	42.84	42.23	41.02	38.58	38.78	42.57	42.78	39.03	36.60	*36.55*	-	**35.09**	3.99%
PEMS08	MAE	22.20	22.72	21.42	17.86	18.02	18.61	19.13	17.13	16.94	*15.95*	17.73	**15.62**	2.07%
	MAPE	15.32	14.03	13.09	11.45	11.40	13.08	12.68	10.96	10.60	*10.09*	11.20	**9.47**	6.14%
	RMSE	32.06	35.79	34.03	27.83	27.83	28.16	31.05	26.80	26.25	*25.22*	26.76	**24.60**	2.46%

- dash denotes that the methods do not implement on this dataset. $*$ denotes re-implementation or re-training.
† denotes the variant without causal convolution.

representation for prediction, inspired by [18, 32], we utilize *permutation entropy* (*PE*) [6] to measure the predictability [26] of the original input and the enhanced representation learned by SCINet. Time series with lower PE values are thought less complex, thus theoretically easier to predict. The PE values of the original time series and the corresponding enhanced representations are shown in Table 5. As can be observed, the enhanced representations learned by SCINet have lower PE values compared with the original inputs, which indicates that it is easier to predict the future from the enhanced representations using the same forecaster. Please note that PE is only a quantitative measurement based on complexity. It would not be proper to say that a time series with lower PE value will be easier to predict than a different one with higher PE value because the prediction accuracy also depends on many other factors, such as the available data for training, the trend and seasonality elements of the time series, as well as the prediction model.

4.3 Ablation study

To further assess the effectiveness of each main component described in Section 3, we experiment on several model variants on three representative datasets.

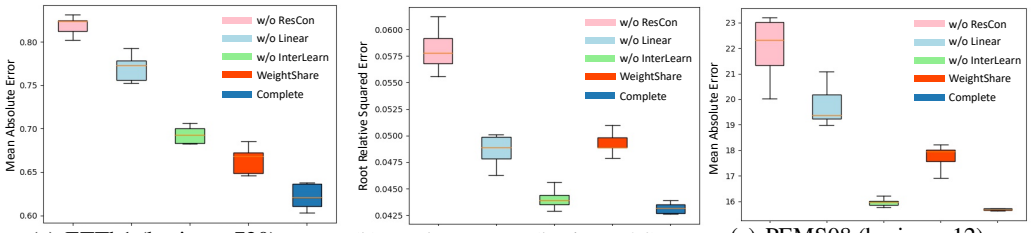
We first set the number of stacks $K = 1$ and the number of SCINet levels $L = 3$. For SCI-Block (Section 3.1), to validate the effectiveness of the interactive learning and the distinct convolution weights for processing the sub-sequences, we experiment on two variants, namely *w/o. InterLearn* and *WeightShare*. The *w/o. InterLearn* is obtained by removing the interactive-learning described in Eq. (1) and (2). In this case, the two sub-sequences would be updated using $\mathbf{F}'_{odd} = \rho(\phi(\mathbf{F}_{odd}))$ and $\mathbf{F}'_{even} = \eta(\psi(\mathbf{F}_{even}))$. For *WeightShare*, the modules ϕ, ρ, ψ, and η share the same weights. The evaluation results in Fig. 4 show that both interactive learning and the distinct weights are essential for effectively learning multi-resolution temporal patterns for precise forecasting.

For the design of SCINet (Section 3.2), we experiment on two variants. The first variant *w/o. ResConn* is obtained by removing the residual connection from the complete SCINet. The other variant *w/o. Linear* removes the final fully-connected layer from the complete model. As can be observed in Fig. 4, removing the residual connection leads to a significant performance drop, indicating the

Table 5: Permutation entropy comparison before and after SCINet.

Permutation Entropy		Datasets										
		ETTh1	ETTh2	ETTm1	Traffic	Solar-Energy	Electricity	Exc-rate	PEMS03	PEMS04	PEMS07	PEMS08
Parameters	m (τ = 1)*	6	6	7	6	7	6	6	6	6	6	6
Value	Original Input	0.8878	0.9009	0.8455	0.9371	0.4739	0.9489	0.8260	0.9649	0.9203	0.9148	0.9390
	Enhanced Representation	0.7096	0.7715	0.6571	0.8832	0.3537	0.8901	0.7836	0.8377	0.8749	0.8330	0.8831

* m (embedding dimension) and τ (time-lag) are two parameters used for calculating PE, and the values are selected following [32, 18].



(a) ETTh1 (horizon=720) (b) Exchange-rate (horizon=24) (c) PEMS08 (horizon=12)

Figure 4: Component analysis of SCINet on three datasets. Smaller values are better. See Section 4.3.

importance of the residual learning to ease the prediction. The fully-connected layer is also important for forecasting, as it can be regarded as a decoder to predict the future values from the sum of the original input and the learned residual component.

We then conduct experiments on ETTh1 to see the impact of different K and L under various length of the look-back window T. The length of the prediction horizon is fixed to be 24. As can be observed from Table. 6 (The best result for K = 1 or $L = 3$ with a certain T is inbold), when T is much larger than the length of the prediction horizon, e.g. $T = 96$ or 128, larger L would give better performance. However, purely increasing the number of levels L is unnecessary when T is too large. For example, for $T = 192$, $L = 4$ performs better than $L = 5$. This is because the interactive learning in the SCI-Block (Section 3.1) allows the two sub-sequences to capture 'global'-view information from each other, which is not restricted by the kernel size or the receptive fields of the convolution operations. Thus $L = 3$ or 4 is enough for effective information extraction for various length of the look-back window T.

As for the number of stacks K, if T is comparable with the length of the prediction horizon, we need to increase K to achieve better prediction performance. This is becasue, the information extracted from a single SCINet is typically not enough for a short look-back window (e.g. $T = 24$ or 36). Thus, stacking more SCINets facilitates more effective learning of the temporal correlations between the look-back window and the prediction. However, when T is large, it is not necessary to use a large K, because stacking more SCINets would enlarge the negative impact of the noise in the look-back window on the relatively short prediction horizon. To seek a trade-off between the computational cost and prediction accuracy, based on the experimental results above, we conclude that $L \leq 5$ and $K \leq 3$ would satisfy most of the scenarios.

Table 6: The impact of different L and K.

Number of Levels & Stacks	Horizon T	24	36	96	128	192
Level L (K =1)	2	0.3932	0.3837	0.3792	0.3736	0.3877
	3	**0.3712**	**0.3679**	0.3662	0.3593	0.3536
	4	-	0.3583	**0.3479**	0.3477	**0.3511**
	5	-	-	0.3493	**0.3463**	0.3675
Stack K (L = 3)	1	0.3712	0.3679	0.3662	0.3593	**0.3536**
	2	0.3618	0.3536	**0.3588**	**0.3501**	0.3652
	3	**0.3489**	**0.3476**	0.3601	0.3632	0.3821
	4	0.3492	0.3488	0.3679	0.3721	0.3812

- Dash denotes the input cannot be further split.

5 Conclusion

In this paper, we propose a novel neural network architecture, *sample convolution and interaction network* (*SCINet*) for time series forecasting, which is motivated by the misconception of existing TCN design principles for the TSF problem and the unique properties of time series data when compared to generic sequence data. The proposed SCINet is designed as a hierarchical structure, which iteratively extracts and exchanges information at different temporal resolutions and learns an effective representation with enhanced predictability. The basic building block, *SCI-Block*, downsamples the input data/feature into two sub-sequences and performs feature extraction and interaction to preserve the heterogeneity information of each sub-part and compensate for the information loss during the downsampling procedure. Extensive experiments on 11 real-world TSF datasets demonstrate the superiority of our model over state-of-the-art methods. In particular, SCINet also performs well on those temporal-spatial datasets, although we do not incorporate specific spatial modeling techniques such as GNN into our solution.

References

[1] M. T. Bahadori and Z. C. Lipton. Temporal-clustering invariance in irregular healthcare time series. *ArXiv*, abs/1904.12206, 2019.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[3] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *ArXiv*, abs/2007.02842, 2020.

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

[5] S. Bai, J. Z. Kolter, and V. Koltun. Trellis networks for sequence modeling. *ArXiv*, abs/1810.06682, 2019.

[6] C. Bandt and B. Pompe. Permutation entropy: a natural complexity measure for time series. *Physical review letters*, 88 17:174102, 2002.

[7] A. Borovykh, S. Bohte, and C. W. Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[8] G. Box, G. Jenkins, and J. Macgregor. Some recent advances in forecasting and control. *Journal of The Royal Statistical Society Series C-applied Statistics*, 17:158–179, 1968.

[9] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia. Freeway performance measurement system: Mining loop detector data. *Transportation Research Record*, 1748:102 – 96, 2001.

[10] K. Cho, B. V. Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv*, abs/1409.1259, 2014.

[11] Z. Cui, W. Chen, and Y. Chen. Multi-scale convolutional neural networks for time series classification. *ArXiv*, abs/1603.06995, 2016.

[12] P. D'Urso, L. Giovanni, and R. Massari. Trimmed fuzzy clustering of financial time series based on dynamic time warping. *Annals of Operations Research*, 299:1379–1395, 2019.

[13] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

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

[17] R. Huang, C. Huang, Y. Liu, G. Dai, and W. Kong. Lsgcn: Long short-term traffic prediction with graph convolutional networks. In *IJCAI*, 2020.

[18] Y. Huang and Z. Fu. Enhanced time series predictability with well-defined structures. *Theoretical and Applied Climatology*, pages 1–13, 2019.

[19] R. Hyndman and A. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688, 2006.

[20] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.

[21] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long- and short-term temporal patterns with deep neural networks. *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.

[22] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *NeurIPS*, 32:5243–5253, 2019.

[23] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

[24] B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379, 2021.

[25] B. Lim, S. O. Arik, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *arXiv preprint arXiv:1912.09363*, 2019.

[26] E. N. Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1, 1996.

[27] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1:111–153, 1982.

[28] S. Mallat. A theory for multi-resolution approximation: the wavelet approximation. *IEEE Trans. PAMI*, 11:674–693, 1989.

[29] L. Mengzhang and Z. Zhanxing. Spatial-temporal fusion graph neural networks for traffic flow forecasting. 2020.

[30] L. Mengzhang and Z. Zhanxing. Spatial-temporal fusion graph neural networks for traffic flow forecasting. *arXiv preprint arXiv:2012.09641*, 2020.

[31] N. H. Nguyen and B. Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. *ArXiv*, abs/2101.10460, 2021.

[32] F. Pennekamp, A. C. Iles, J. Garland, G. Brennan, U. Brose, U. Gaedke, U. Jacob, P. Kratina, B. Matthews, S. Munch, M. Novak, G. M. Palamara, B. C. Rall, B. Rosenbaum, A. Tabi, C. Ward, R. Williams, H. Ye, and O. Petchey. The intrinsic predictability of ecological time series and its potential to guide forecasting. *Ecological Monographs*, 89, 2019.

[33] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. *NeurIPS*, 31:7785–7794, 2018.

[34] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371, 2013.

[35] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[36] R. Sen, H.-F. Yu, and I. Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *NeurIPS*, 2019.

[37] X. Shi and D.-Y. Yeung. Machine learning for spatiotemporal sequence forecasting: A survey. *arXiv preprint arXiv:1808.06865*, 2018.

[38] S.-Y. Shih, F.-K. Sun, and H. yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, pages 1–21, 2019.

[39] C. Song, Y. Lin, S. Guo, and H. Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *AAAI*, 2020.

[40] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop*, 2016.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[42] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, 2019.

[43] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[44] T. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

[45] G. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50: 159–175, 2003.

[46] H.-Y. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *ArXiv*, abs/2012.07436, 2020.

A Datasets and Evaluation Metrics

We conduct the experiments on 11 popular time-series datasets, namely *Electricity Transformer Temperature* (ETTh1, ETTh2 and ETTm1) [46], *Traffic*, *Solar-Energy*, *Electricity* and *Exchange-Rate* [21] and *PeMS* (PEMS03, PEMS04, PEMS07 and PEMS08) [9], ranging from power, energy, finance and traffic domains. Here, we present the details of each dataset and the metrics used for assessing the performance of time series forecasting (TSF) models on different datasets.

A.1 Electricity Transformer Temperature (ETT)

ETT[3]consists of 2 year electric power data collected from two separated counties of China, including hourly subsets $\{ETTh1, ETTh2\}$ and quarter-hourly subsets $\{ETTm1\}$. Each data point includes an "oil temperature" value and 6 power load features. The train, validation and test sets contain 12, 4 and 4 months data, respectively.

For data pre-processing, we perform zero-mean normalization, i.e., $X^{'} = (X - mean(X))/std(X)$, where $mean(X)$ and $std(X)$ are the mean and the standard deviation of historical time series, respectively. We use Mean Absolute Errors (MAE) [19] and Mean Squared Errors (MSE) [27] for model comparison.

$$MAE = \frac{1}{\tau}\sum_{i=0}^{\tau}|\hat{x}_i - x_i| \tag{6}$$

$$MSE = \frac{1}{\tau}\sum_{i=0}^{\tau}(\hat{x}_i - x_i)^2 \tag{7}$$

where $\hat{x}_i$ is the model's prediction, and x_i is the ground-truth. τ is the length of the prediction horizon.

A.2 Traffic, Solar-Energy, Electricity and Exchange-Rate

Traffic[4] contains the hourly data describing the road occupancy rates (ranging from 0 to 1) that are recorded by the sensors on San Francisco Bay area freeways from 2015 to 2016 (48 months in total). *Solar-Energy*[5] records the solar power production from 137 PV plants in Alabama State, which are sampled every 10 minutes in 2016. *Electricity*[6] includes the hourly electricity consumption (kWh) records of 321 clients from 2012 to 2014. *Exchange-Rate*[7] collects the daily exchange rates of 8 foreign countries from 1990 to 2016.

In our experiments, the length of the look-back window T for the above datasets is 168, and we trained independent models for different length of future horizon (i.e., $\tau = 3, 6, 12, 24$). We use Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) to evaluate the performance of the TSF models on these datasets following [21], which are calculated as follows:

$$RSE = \frac{\sqrt{\sum_{i=0}^{\tau}(\hat{x}_i - x_i)^2}}{\sqrt{\sum_{i=0}^{\tau}(x_i - mean(X))^2}}, \tag{8}$$

$$CORR = \frac{1}{d}\sum_{j=0}^{d}\frac{\sum_{i=0}^{\tau}(x_{i,j} - mean(X_j))(\hat{x}_{i,j} - mean(\hat{X}_j))}{\sum_{i=0}^{\tau}(x_{i,j} - mean(X_j))^2(\hat{x}_{i,j} - mean(\hat{X}_j))^2}, \tag{9}$$

where X and $\hat{X}$ are the ground-truth and model's prediction, respectively. d is the number of time-series.

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

[4] http://pems.dot.ca.gov

[5] http://www.nrel.gov/grid/solar-power-data.html

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

[7] https://github.com/laiguokun/multivariate-time-series-data

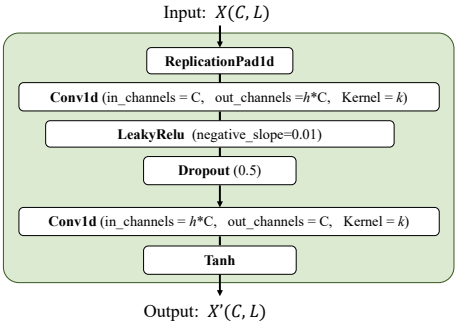


Figure 5: The structure of ϕ, ρ, ψ, and η.

Table 7: The hyperparameters in ETT datasets

Model configurations		ETTh1					ETTh2					ETTm1				
Hyperparameter	Horizon	24	48	168	336	720	24	48	168	336	720	24	48	96	288	672
	Look-back window	48	96	336	336	720	48	96	336	336	720	48	96	384	672	672
	Batch size	4	16	32	512	128	16	4	8	128	32	16	16	32	32	32
	Learning rate	9e-3	9e-3	5e-4	1e-4	1e-5	7e-3	7e-3	5e-5	5e-5	1e-5	1e-3	1e-3	5e-5	1e-5	1e-5
SCI Block	h	4	4	4	1	1	8	4	0.5	1	32	4	4	0.5	4	4
	k	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
SCINet	L (level)	3	3	3	4	5	3	4	4	4	5	3	4	4	5	5
Stacked SCINet	K (stack)	1	1	1	1	1	1	1	1	1	1	2	2	2	1	1

A.3 PeMS

PeMS[8] contains four public traffic network datasets (*PEMS03, PEMS04, PEMS07 and PEMS08*) which are respectively constructed from Caltrans Performance Measurement System (PeMS) of four districts in California. The data is aggregated into 5-minutes windows, resulting in 12 points per hour and 288 points per day. We use traffic flow data from the past hour to predict the flow for the next hour. The data is pre-processed using zero-mean normalization as ETT.

Following [19], we use Root Mean Squared Errors (RMSE) and Mean Absolute Percentage Errors (MAPE) as evaluation metrics on this dataset.

$$RMSE = \sqrt{\frac{1}{\tau}\sum_{i=0}^{\tau}(\hat{x}_i - x_i)^2}, \tag{10}$$

$$MAPE = \sqrt{\frac{1}{\tau}\sum_{i=0}^{\tau}|(\hat{x}_i - x_i)/x_i|}. \tag{11}$$

[8] `https://pems.dot.ca.gov`

Table 8: The hyperparameters in Traffic, Solar-energy, Electricity and Exchange-rate datasets

Model configurations		Traffic				Solar				Electricity				Exc-Rate			
Hyperparameter	Horizon	3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
	Look-back window	168															
	Batch size	16				1024				32				4			
	Learning rate	5e-4				1e-4				9e-3				5e-3			
SCI Block	h	2				2				8				0.125			
	k	5				5				5				5			
SCINet	L (level)	3				4				3				3			
Stacked SCINet	K (stack)	1				1				1				1			
	Loss weight (λ)	$\times$				0.5				$\times$				0.5			

Table 9: Univariate time-series forecasting results on ETT datasets

Methods	Metrics	ETTh1					ETTh2					ETTm1				
		Horizon					Horizon					Horizon				
		24	48	168	336	720	24	48	168	336	720	24	48	96	288	672
LogTrans	MSE	0.103	0.167	0.207	0.230	0.273	0.102	0.169	0.246	0.267	0.303	0.065	0.078	0.199	0.411	0.598
	MAE	0.259	0.328	0.375	0.398	0.463	0.255	0.348	0.422	0.437	0.493	0.202	0.220	0.386	0.572	0.702
Reformer	MSE	0.222	0.284	1.522	1.860	2.112	0.263	0.458	1.029	1.668	2.030	0.095	0.249	0.920	1.108	1.793
	MAE	0.389	0.445	1.191	1.124	1.436	0.437	0.545	0.879	1.228	1.721	0.228	0.390	0.767	1.245	1.528
LSTMa	MSE	0.114	0.193	0.236	0.590	0.683	0.155	0.190	0.385	0.558	0.640	0.121	0.305	0.287	0.524	1.064
	MAE	0.272	0.358	0.392	0.698	0.768	0.307	0.348	0.514	0.606	0.681	0.233	0.411	0.420	0.584	0.873
DeepAR	MSE	0.107	0.162	0.239	0.445	0.658	0.098	0.163	0.255	0.604	0.429	0.091	0.219	0.364	0.948	2.437
	MAE	0.280	0.327	0.422	0.552	0.707	0.263	0.341	0.414	0.607	0.580	0.243	0.362	0.496	0.795	1.352
ARIMA	MSE	0.108	0.175	0.396	0.468	0.659	3.554	3.190	2.800	2.753	2.878	0.090	0.179	0.272	0.462	0.639
	MAE	0.284	0.424	0.504	0.593	0.766	0.445	0.474	0.595	0.738	1.044	0.206	0.306	0.399	0.558	0.697
Prophet	MSE	0.115	0.168	1.224	1.549	2.735	0.199	0.304	2.145	2.096	3.355	0.120	0.133	0.194	0.452	2.747
	MAE	0.275	0.330	0.763	1.820	3.253	0.381	0.462	1.068	2.543	4.664	0.290	0.305	0.396	0.574	1.174
Informer	MSE	0.098	*0.158*	*0.183*	0.222	0.269	*0.093*	*0.155*	*0.232*	0.263	*0.277*	*0.030*	0.069	0.194	*0.401*	*0.512*
	MAE	0.247	*0.319*	*0.346*	0.387	0.435	*0.240*	*0.314*	*0.389*	0.417	*0.431*	*0.137*	0.203	0.372	0.554	*0.644*
Informer†	MSE	*0.092*	0.161	0.187	*0.215*	*0.257*	0.099	0.159	0.235	*0.258*	0.285	0.034	*0.066*	*0.187*	0.409	0.519
	MAE	*0.246*	0.322	0.355	*0.369*	*0.421*	0.241	0.317	0.390	*0.423*	0.442	0.160	*0.194*	*0.384*	*0.548*	0.665
SCINet	MSE	**0.028**	**0.049**	**0.075**	**0.087**	**0.156**	**0.068**	**0.089**	**0.156**	**0.173**	**0.249**	**0.018**	**0.043**	**0.066**	**0.114**	**0.154**
	MAE	**0.128**	**0.171**	**0.215**	**0.231**	**0.316**	**0.189**	**0.227**	**0.312**	**0.338**	**0.399**	**0.085**	**0.139**	**0.186**	**0.261**	**0.308**
IMP	MSE	69.57%	68.99%	59.02%	59.53%	39.30%	26.88%	42.58%	32.76%	32.95 %	10.11%	40.00%	34.85%	64.71%	71.57%	69.92%
	MAE	47.97%	46.39%	37.86%	37.40%	24.94%	21.25%	27.71%	19.79%	20.09%	7.42%	37.96%	28.35%	51.56%	52.37%	52.17%

B Reproducibility

Our code is implemented with the open-source PyTorch. All the experiments are conducted on Nvidia Tesla V100 SXM2 GPUs (32GB memory), which is sufficient for all the baselines. The detailed hyperparameters will be introduced in the following.

Structure of the network modules ϕ, ρ, ψ, and η in SCI-Block: As shown in Fig. 5, ϕ, ρ, ψ, and η share the same network architecture. First, the replication padding is used to alleviate the border effects caused by the convolution operation. Then, a 1d convolutional layer with kernel size k is applied to extend the input channel C to h*C and followed with LeakyRelu and Dropout. Next, the second 1d convolutional layer with kernel size k is to recover the channel h*C to the input channel C. The stride of all the convolutions is set to be 1. We use a LeakyRelu activation after the first convolutional layer because of its sparsity properties and a reduced likelihood of vanishing gradient. We apply a Tanh activation after the second convolutional layer because we do not want to discard negative values in this stage.

Training details: For *ETT*, we follow the same training strategy as [46] to preserve 10% validation data for each dataset, and all the experiments are conducted over five random train/val shifting selection along time. The results are averaged over the five runs. For all other datasets, we fix the seed to be 4321, and train the model for 100 epochs. The reported results on the test set are based on the model that achieves the best performance on the validation set.

Hyper-parameter tuning: We conduct a grid search over all the tunable hyper-parameters on the held-out validation set of the datasets. The detailed hyper-parameter configurations of *ETT* are shown in Table. 7, and those for *Traffic, Solar-Energy, Electricity and Exchange-rate* are shown in Table. 8. We only apply the weighted loss to the *Solar* and *Exchange-rate* data since they show less auto-correlation [21], which indicates the temporal correlation of the distant time-stamp cannot be well modelled by general L1 loss. Besides, the parameters of the four datasets in *PeMS* are the same and can be summarized as follows: *batch size*: 8, *learning rate*: 1e-3, *h*: 0.03125, *k*: 3, *L*: 3, *K*: 2. For LSTM in *PeMS* dataset, the hidden state is chosen from $\{32, 64, 128, 256\}$. For TCN, in order to cover different length of the look-back window, the number of layers is chosen from $\{3, 4, 5\}$ and the dimension of the hidden state is chosen from $\{32, 64, 128, 256\}$. Besides, to build a non-causal TCN[9], we only need to remove the *chomps* in the code and make the padding equal to the dilation.

C Extra Experimental Results

Zhou et al [46] conduct two tasks on *ETT* datasets, namely *multivariate forecasting* and *univariate forecasting*. Due to space limitation, we only report the results of multivariate forecasting in the paper. Here, we show the results of the univariate forecasting in Table 9. As can be seen, the proposed SCINet also shows significant superiority in univariate forecasting over other competitors.

[9]https://github.com/locuslab/TCN/issues/45

D Limitation and Future Work

In this paper, we mainly focus on TSF problem for the *regular time series* collected at even intervals of time and ordered chronologically. However, in real-world applications, the collected time series usually have missing data or noise data, which is referred to as *irregular time series*. The proposed SCINet is relatively robust to the noisy data thanks to the progressive downsampling and interactive leaning, but it might be affected by the missing data if the ratio exceeds some threshold, say 50%, where the downsampling based multi-resolution sequence representation in SCINet may introduce bias, leading to poor prediction performance. To extend the application scenarios of SCINet, e.g. for handling irregular time series, in future, we plan to combine it with other data imputation techniques to better model the temporal dependencies of the missing data.

E Broader Impact

Time-series forecasting has significant societal implications as well. It can potentially promote the research progress in addressing many world-level economic and societal issues, such as predictions of viral transmission, economic growth, and climate change. For example, the outbreak of the COVID-19 was almost a devastating blow to the world, especially for the underdeveloped area. Therefore, an accurate prediction of the chain of virus transmission can help the government design suitable strategies to break the transmission. Moreover, we also notice that there should be a negative impact of this technique on protecting personal privacy and security. For example, the prediction of the customers' behaviour may be utilized unscrupulously in various business. Hackers may also use the predicted data to attack the bank's security system for financial crime. Therefore, we call on researchers to make rational use of artificial intelligence and cultivate a responsible AI-ready culture in technology development.