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

Minhao Liu,[1] **Ailing Zeng,** [1] **Zhijian Xu,** [1] **Qiuxia Lai** [1] **Qiang Xu** [1]

[1] The Chinese University of Hong Kong
{mhliu, alzeng, zjxu, qxlai, qxu}@cse.cuhk.edu.hk

Abstract

Time series is a special type of sequence data, a set of observations collected at even time intervals 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. In particular, three components characterize time series: trend, seasonality, and irregular components, and the former two components enable us to perform forecasting with reasonable accuracy. Other types of sequence data do not have such characteristics.

Motivated by the above, in this paper, we propose a novel neural network architecture that conducts sample convolution and interaction for temporal modeling and apply it for the time series forecasting problem, namely **SCINet**. Compared to conventional dilated causal convolution architectures, the proposed downsample-convolve-interact architecture enables multi-resolution analysis besides expanding the receptive field of the convolution operation, which facilitates extracting temporal relation 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. Our codes and data are presented in the supplemental material.

1 Introduction

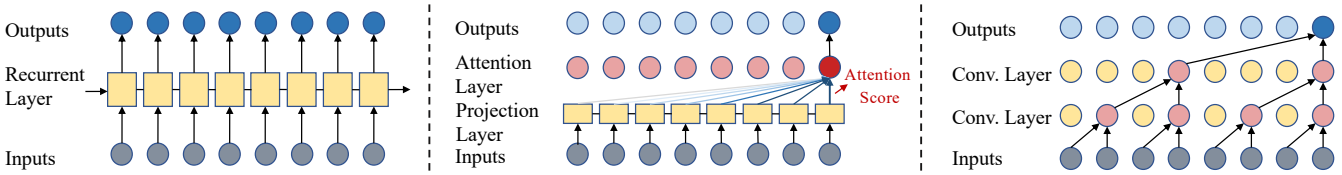
Time series forecasting (TSF) enables decision-making with the estimated future evolution of metrics or events and thus plays a crucial role in various scientific and engineering fields such as healthcare (Bahadori and Lipton 2019), energy management (Zhou et al. 2021), traffic flow (Zhou et al. 2021) and financial investment (D'Urso, Giovanni, and Massari 2019) to name a few. Traditional time series forecasting methods such as autoregressive integrated moving average (ARIMA) model (Box, Jenkins, and Macgregor 1968) and Holt-Winters seasonal method (Holt 2004) have theoretical guarantees, but they are mainly applicable for univariate forecasting problems, 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 (Lim and Zohren 2021; Oreshkin et al. 2020).

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 (Lim and Zohren 2021): (i). recurrent neural networks (RNNs) and their variants such as long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) and gated recurrent units (GRUs) (Cho et al. 2014); (ii). Transformer (Vaswani et al. 2017) and its variants (Kitaev, Kaiser, and Levskaya 2019; Li et al. 2019; Zhou et al. 2021); (iii). temporal convolutional networks (TCN) (Bai, Kolter, and Koltun 2018a). Generally speaking, TCN is 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 spatial-temporal TSF problems (Li et al. 2018; Yu, Yin, and Zhu 2018; Song et al. 2020).

While the above TSF methods have been successfully used in many real-world forecasting problems, they rely on generic sequence models and ignore the fact that 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. In fact, different from other types of sequence data, three components characterize time series: trend, seasonality, and irregular components, and the former two components enable us to perform forecasting with reasonable accuracy.

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 main contributions are as follows:

- We propose a hierarchical TSF framework, *SCINet*, based on the unique attributes of time series. 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) (Huang and Fu 2019).
- 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 compensate for the information loss during the downsampling procedure, we incorporate interactive learning between the two convolutional features within each SCI-Block.



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

Figure 1: Existing sequence modeling architectures for time series forecasting.

Extensive experiments on various real-world TSF datasets show that our model consistently outperforms existing TSF approaches by a considerable margin. In particular, for the ETT dataset in (Zhou et al. 2021), on average, SCINet achieves more than 40% relative improvement compared to state-of-the-art methods, in terms of mean squared errors of the prediction results.

2 Related Work and Motivation

Given a long time series $\mathbf{X}^*$ and a look-back window of fixed length T, at timestamp t, time series forecasting is to predict $\hat{\mathbf{X}}_{t+1:t+\tau} = \{\mathbf{x}_{t+1}, ..., \mathbf{x}_{t+\tau}\}$ 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 variates. 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. When $\tau > 1$, we can either directly optimize the multi-step forecasting objective (*direct multi-step (DMS)* estimation), or learn a single-step forecaster and iteratively apply it to get multi-step predictions (*iterated multi-step (IMS)* estimation) (Shi and Yeung 2018).

In this section, we focus on deep learning-based methods for TSF due to their superior performance, especially for multivariate time series ($d > 1$).

2.1 Related Work

RNN-based TSF methods (Rangapuram et al. 2018; Salinas et al. 2020) 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 (Vaswani et al. 2017) have taken the place of RNN models in almost all sequence modeling tasks, thanks to the effectiveness of the self-attention mechanisms. Consequently, various Transformer-based TSF methods (see Fig. 1 (b)) are proposed in the literature (Li et al. 2019; Lim et al. 2021), and they are shown to be quite effective in predicting long sequences. However, the overhead of Transformer-based models is a serious concern and lots of research efforts are dedicated to tackling this problem (e.g., (Zhou et al. 2021)).

The local correlation of time series data is reflected in the continuous change within a small time slot, and such local features can be effectively captured by convolutional filters. Consequently, convolutional models (Fig. 1 (c)) have become a popular choice for various TSF problems (Borovykh, Bohte, and Oosterlee 2017; Bai, Kolter, and Koltun 2018a; Sen, Yu, and Dhillon 2019; Wu et al. 2019; Nguyen and Quanz 2021). Moreover, convolutional filters can work seamlessly with GNNs to solve various spatial-temporal TSF problems.

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 the following subsection, we highlight our insights and motivate our work.

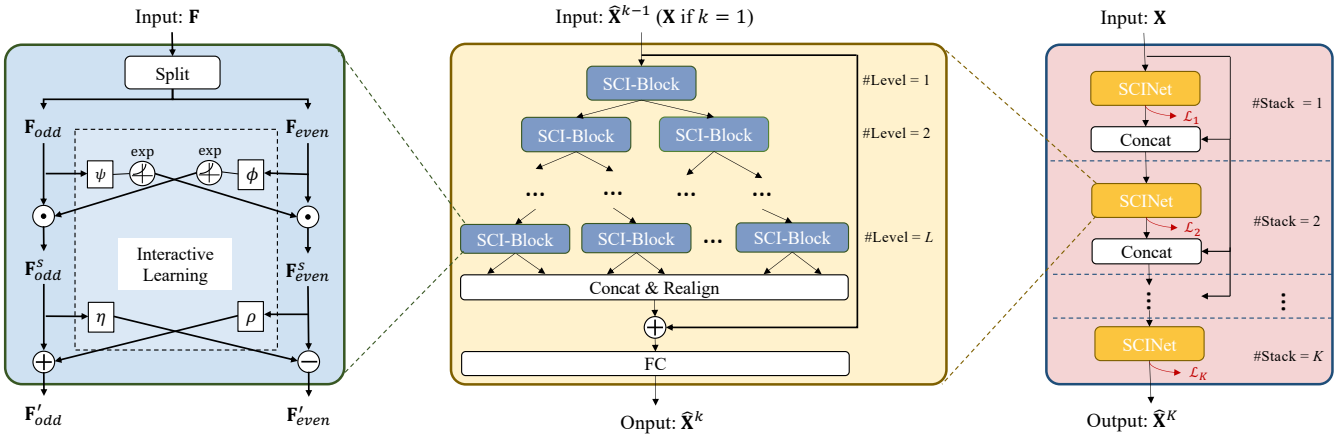
2.2 Rethinking Dilated Causal Convolution

Dilated causal convolution was first proposed for generating raw audio waveforms in WaveNet (van den Oord et al. 2016). Later, (Bai, Kolter, and Koltun 2018a) simplifies the WaveNet architecture to the so-called temporal convolutional networks 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, TCN has been widely used in all kinds of time series forecasting problems and achieve promising results (Wu et al. 2019; Sen, Yu, and Dhillon 2019).

With *causal convolutions* in the TCN architecture, an output i is convolved only with the i^{th} and earlier elements in the previous layer. While causality should be kept in forecasting tasks, the potential "future information leakage" problem exists only when the output and the input have temporal overlaps. In other words, causal convolutions should be applied only in IMS-based forecasting wherein the previous output serves as the input for future prediction. When the predictions are completely based on the known inputs in the look-back window (e.g., DMS-based forecasting), there is no need to use causal convolutions. We can safely apply *normal convolutions* that uses 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, TCN uses *dilated convolutions* to achieve a large receptive field with fewer convolutional layers. However, a single convolutional filter is shared across each layer in TCN, restricting the extraction of temporal dynamics from the data/feature in the previous layer.

In view of the satisfactory performance of TCN in many DMS forecasting tasks despite the above limitations, we propose a novel downsample-convolve-interact architecture *SCINet* that takes the unique properties of time series into consideration to achieve better prediction accuracy, as detailed in the following section.



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

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

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 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 subsequences, and then process each sub-sequence with a set of distinct convolutional filters to extract both homogeneous and heterogeneous information from each part. To compensate for 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*.

The *Splitting* procedure downsamples the original sequence $\mathbf{F}$ into two sub-sequences $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$ by separating the even and the odd elements, which are of coarser temporal resolution but preserve most information of the original sequence.

Next, we use different convolutional kernels to extract features from $\mathbf{F}_{even}$ and $\mathbf{F}_{odd}$. As the kernels are separate, the extracted features from them would contain both homogeneous and heterogeneous information with enhanced representation capabilities. To compensate for potential information loss with downsampling, we propose a novel *interactive-learning* strategy to allow information interchange between the two sub-sequences by learning affine transformation parameters from each other. As shown in Fig. 2 (a), the interactive learning procedure consists of two steps.

First, $\mathbf{F}_{even}$ 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 (see Eq. (1)). 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.

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

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

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 added to or subtracted from[2] $\mathbf{F}^s_{even}$ 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.

Compared to the dilated convolutions used in the TCN architecture, the proposed downsample-convolve-interact architecture achieves an even larger receptive field. More importantly, unlike TCN that employs a single shared convolutional filter in each layer, significantly restricting its feature extraction capabilities, SCI-Block aggregates essential information extracted from the two downsampled sub-sequences according to the given supervision for prediction.

[1] *Multi-resolution analysis* is a classic method for signal analysis (Mallat 1989) and has been used in time series classification tasks (Cui, Chen, and Chen 2016).

[2] The selection of the operators in Eq.(2) affects the parameter initialization of our model and we show its impact in the Appendix.

Table 1: The datasets used in our experiments.

Datasets	ETTh1	ETTh2	ETTm1	PEMS03	PEMS04	PEMS07	PEMS08
# Variates	7	7	7	358	307	883	170
# Timesteps	17,420	17,420	69,680	26,209	16,992	28,224	17,856
Granularity	1hour	1hour	15min	5min	5min	5min	5min
Start time	7/1/2016	7/1/2016	7/1/2016	5/1/2012	7/1/2017	5/1/2017	3/1/2012
Task type	Multi-step	Multi-step	Multi-step	Multi-step	Multi-step	Multi-step	Multi-step
Data partition	Training/Validation/Testing: 12/4/4 months			Training/Validation/Testing: 6/2/2 split ratio			

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 number of levels. For the k-th SCINet in the stacked SCINet (Section 3.3), the input time series $\mathbf{X}$ (for $k=1$) or feature vector $\hat{\mathbf{X}}^{k-1} = \{\hat{\mathbf{x}}_1^{k-1}, ..., \hat{\mathbf{x}}_\tau^{k-1}\}$ (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. 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 (He et al. 2016). Finally, a fully-connected layer is used to decode the enhanced sequence representation into $\hat{\mathbf{X}}^k = \{\hat{\mathbf{x}}_1^k, ..., \hat{\mathbf{x}}_\tau^k\}$.

Generally speaking, SCINet does not require downsampling the original sequence to the coarsest level for feature extraction because it is not helpful for the forecasting task. In other words, L could be much smaller than $\log_2 T$, given the look-back window size T. Our empirical study shows that the best prediction results are achieved with $L \leq 5$ in most cases.

3.3 Stacked SCINet

When there are sufficient training samples, we could stack K layers of SCINets to achieve better prediction accuracy (see Fig. 2 (c)), at the cost of a more complex model structure.

Specially, we apply *intermediate supervision* (Bai, Kolter, and Koltun 2018b) 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, $\hat{\mathbf{X}}^k$ with length τ, is concatenated with part of the input $\mathbf{X}_{t-(T-\tau)+1:t}$ to recover the length to the original input 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. Our empirical study shows that $K \leq 3$ would be sufficient for most cases.

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:

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

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

$$\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k. \tag{4}$$

4 Experiments

In this section, we show the quantitative and qualitative comparisons with the state-of-the-art models for time series forecasting. We also present a comprehensive ablation study to evaluate the effectiveness of different components in SCINet. 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 experiments on 7 popular time-series datasets[3], namely *Electricity Transformer Temperature* (Zhou et al. 2021) and traffic datasets *PeMS* (Chen et al. 2001). A brief description of these datasets is listed in Table 1. To make a fair comparison, we use the same evaluation metrics as the original publications in each dataset. For *ETT* datasets, we use the Mean Absolute Errors (MAE) (Hyndman and Koehler 2006) and Mean Squared Errors (MSE) (Makridakis et al. 1982). For *PeMS*, we use the MAE, Root Mean Squared Errors (RMSE) and Mean Absolute Percentage Errors (MAPE) following (Hyndman and Koehler 2006) . Lower values are better for all the metrics. The evaluation is averaged on all the predictions for multi-step predictions.

4.2 Results and Analyses

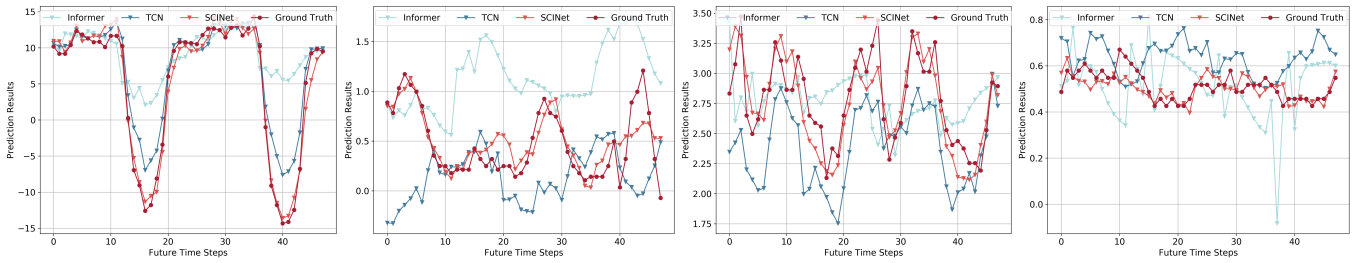
ETT datasets (Zhou et al. 2021) are used to evaluate the performance of long-sequence TSF tasks, which are conducted on two experimental settings, *Multivariate Time-series Forecasting* and *Univariate Time-series Forecasting*. For a fair comparison, we keep all input lengths T the same as Informer. The results are shown in Table 2 and Table 3, respectively.

[3] Additional results on several other datasets are presented in the Appendix, due to space limiations.

Table 2: Performance comparison on the *ETT* datasets. The best results are in **bold** and the second best results are underlined with *blue* color. "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	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	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	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	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	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	**0.341**	**0.368**	**0.451**	**0.502**	**0.583**	**0.188**	**0.279**	**0.505**	**0.618**	**1.074**	**0.126**	**0.169**	**0.191**	**0.365**	**0.713**
	MAE	**0.379**	**0.395**	**0.457**	**0.497**	**0.560**	**0.288**	**0.358**	**0.504**	**0.560**	**0.761**	**0.229**	**0.274**	**0.291**	**0.415**	**0.604**
IMP	MSE	26.98%	23.01%	30.83%	36.21%	37.11%	55.66%	53.03%	78.56%	74.21%	58.24%	38.71%	22.83%	21.40%	49.59%	40.18%
	MAE	20.55%	19.72%	24.59%	26.37%	27.27%	38.06%	39.22%	56.85%	53.87%	38.68%	18.21%	13.29%	15.16%	34.44%	34.77%

∗ denotes re-implementation. † denotes the variant with normal convolutions.



(a) Sequence 1388, Variate 1 (b) Sequence 2753, Variate 4 (c) Sequence 441, Variate 5 (d) Sequence 2745, Variate 6

Figure 3: The prediction results (Horizon = 48) of SCINet, Informer (Zhou et al. 2021), and TCN on randomly-selected sequences from ETTh1 dataset.

Multivariate Time-series Forecasting on ETT: As can be seen from Table 2, compared with RNN-based methods such as LSTMa (Bahdanau, Cho, and Bengio 2015) and LSTnet (Lai et al. 2018), Transformer-based methods (Kitaev, Kaiser, and Levskaya 2019; Li et al. 2019; Zhou et al. 2021) are better at capturing the long-term latent patterns in the entire historical data for predicting the future, leading to lower prediction errors. TCN-based methods (Bai, Kolter, and Koltun 2018a; Wu et al. 2020) further outperform Transformer-based methods, because the stacked convolutional layers allow for more effective local-to-global temporal relation learning for multivariate time series compared with RNN and Transformer-based methods. Specially, TCN† denotes a variant of TCN wherein causal convolutions are replaced by normal convolutions, and improves the original TCN across all the *ETT* datasets. SCINet outperforms all the above models by a large margin. Fig. 3 presents the qualitative results on some randomly selected sequences, which clearly demonstrate the capability of SCINet in obtaining the trend and seasonality of time series for TSF.

Univariate Time-series Forecasting on ETT: In this experimental setting, we bring several strong baseline methods for univariate forecasting into comparison, including ARIMA, Prophet (Taylor and Letham 2018), DeepAR (Salinas et al. 2020) and N-Beats (Oreshkin et al. 2020). In Table 3, we can observe that N-Beats is superior to other baseline methods in most cases. In fact, N-Beats also takes the unique properties of time series into consideration and directly learns a trend and a seasonality model using a very deep stack of fully-connected layers with residuals, which is a departure from the predominant architectures, such as RNNs, CNNs and Transformers. Nevertheless, the performance of SCINet is still much better than N-Beats.

We attribute the significant performance improvements of SCINet on the ETT datasets to: (i) SCINet effectively captures temporal dependencies from multiple temporal resolutions; (ii) *ETT* datasets are publicly available recently and domain-specific solutions tuned specifically for these datasets do not exist yet.

Traffic datasets *PeMS* (Chen et al. 2001) are complicated spatial-temporal time series and they have been investigated for decades. Most recent approaches (Wu et al. 2019; Song et al. 2020; Bai et al. 2020; Huang et al. 2020; Li and Zhu 2021) use graph neural networks to capture spatial relations while modeling temporal dependencies via conventional TCN or LSTM architectures. We follow the same experimental settings as the above works. As shown in Table 4, these GNN-based methods generally perform better than pure RNN or TCN-based methods. However, SCINet still achieves better performance without sophisticated spatial relation modeling,

Table 3: Univariate time-series forecasting results on the *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
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
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
N-Beats	MSE	*0.042*	*0.065*	*0.106*	*0.127*	*0.269*	*0.078*	*0.123*	0.244	0.270	0.281	0.031	*0.056*	*0.095*	*0.157*	*0.207*
	MAE	*0.156*	*0.200*	*0.255*	*0.284*	*0.422*	*0.210*	*0.271*	0.393	0.418	0.432	*0.117*	*0.168*	*0.234*	*0.311*	*0.370*
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
SCINet	MSE	**0.031**	**0.051**	**0.081**	**0.094**	**0.176**	**0.070**	**0.102**	**0.157**	**0.177**	**0.253**	**0.019**	**0.045**	**0.072**	**0.117**	**0.180**
	MAE	**0.132**	**0.173**	**0.222**	**0.242**	**0.343**	**0.194**	**0.242**	**0.311**	**0.340**	**0.403**	**0.088**	**0.143**	**0.198**	**0.266**	**0.328**
IMP	MSE	26.19%	21.54%	23.58%	25.98%	18.72%	10.26%	17.07%	32.76%	32.95 %	10.11%	38.71%	19.64%	24.21%	25.48%	13.04%
	MAE	15.38%	13.50%	12.94%	14.79%	24.94%	7.62%	10.70%	19.79%	20.09%	7.42%	24.79%	14.88%	15.38%	14.47%	11.35%

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

Datasets	Metrics	Methods												
		*LSTM	*TCN	*TCN†	DCRNN	STGCN	ASTGCN(r)	GraphWaveNet	STSGCN	STFGNN	AGCRN	LSGCN	SCINet	**IMP**
PEMS03	MAE	21.33	19.32	18.87	18.18	17.49	17.69	19.85	17.48	16.77	*15.98*	-	**15.00**	6.13%
	MAPE	21.33	19.93	18.63	18.91	17.15	19.40	19.31	16.78	16.30	*15.23*	-	**14.29**	6.17%
	RMSE	35.11	33.55	32.24	30.31	30.12	29.66	32.94	29.21	*26.28*	*28.25	-	**24.31**	7.49%
PEMS04	MAE	25.14	23.22	22.81	24.70	22.70	22.93	25.45	21.19	20.48	*19.83*	21.53	**18.95**	4.44%
	MAPE	20.33	15.59	14.31	17.12	14.59	16.56	17.29	13.90	16.77	*12.97*	13.18	**11.86**	8.56%
	RMSE	39.59	37.26	36.87	38.12	35.55	35.22	39.70	33.65	32.51	*32.30*	33.86	**30.89**	4.40%
PEMS07	MAE	29.98	32.72	30.53	28.30	25.38	28.05	26.85	24.26	23.46	*22.37*	-	**21.19**	5.27%
	MAPE	15.33	14.26	13.88	11.66	11.08	13.92	12.12	10.21	9.21	*9.12*	-	**8.83**	3.18%
	RMSE	42.84	42.23	41.02	38.58	38.78	42.57	42.78	39.03	36.60	*36.55*	-	**34.03**	6.89%
PEMS08	MAE	22.20	22.72	21.42	17.86	18.02	18.61	19.13	17.13	16.94	*15.95*	17.73	**15.72**	1.44%
	MAPE	15.32	14.03	13.09	11.45	11.40	13.08	12.68	10.96	10.60	*10.09*	11.20	**9.80**	2.87%
	RMSE	32.06	35.79	34.03	27.83	27.83	28.16	31.05	26.80	26.25	*25.22*	26.76	**24.76**	1.82%

- dash denotes that the methods do not implement on this dataset. $*$ denotes re-implementation or re-training. † denotes the variant with normal convolutions.

which further proves the superb temporal modeling capabilities of SCINet.

Inspired by (Huang and Fu 2019; Pennekamp et al. 2019), we use *permutation entropy* (*PE*) (Bandt and Pompe 2002) to measure the predictability of the original input and the enhanced representation learnt by SCINet. Time series with lower PE values are regarded as less complex, thus theoretically easier to predict[4]. 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 learnt by SCINet indeed 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.

4.3 Ablation study

To further assess the effectiveness of each main component used in SCINet, we experiment on several model variants on two datasets: *ETTh1* and *PEMS08*.

We first set the number of stacks $K = 1$ and the number of SCINet levels $L = 3$. For the SCI-Block design, 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 procedure 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 weight.

The evaluation results in Fig. 4 show that both interactive learning and distinct weights are essential, as they improve the prediction accuracies of both datasets at various prediction horizons. At the same time, comparing Fig. 4(a) with Fig. 4(b), we can observe that interactive learning is more effective for cases with longer look-back window sizes. This is because, intuitively, we can extract more effective features by exchanging information between the downsampled subsequences when there are longer look-back windows for such interactions.

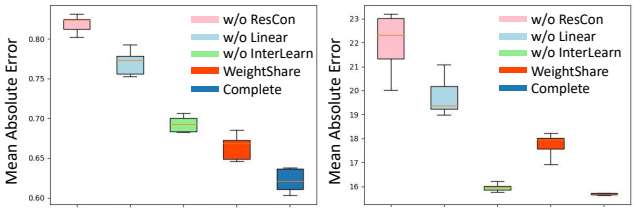
For the design of SCINet with multiple levels of SCI-Blocks, we also experiment on two variants. The first variant *w/o. ResConn* is obtained by removing the residual connection from the complete SCINet. The other variant *w/o. Linear* removes the decoder (i.e., the fully-connected layer) from the complete model. As can be observed in Fig. 4, removing the residual connection leads to a significant performance drop. Besides the general benefit in facilitating the model training, more importantly, the predictability of the original time series is enhanced with the help of residuals. The fully-connected layer is also critical for prediction accuracy, indicating the effectiveness of the decoder in extracting and

[4]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 always easier to predict than a different type of time series with a 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 data, as well as the predictive model.

Table 5: Permutation entropy comparison before and after SCINet.

Permutation Entropy		Datasets						
		ETTh1	ETTh2	ETTm1	PEMS03	PEMS04	PEMS07	PEMS08
Parameters	m (τ = 1)*	6	6	7	6	6	6	6
Value	Original Input	0.8878	0.9009	0.8455	0.9649	0.9203	0.9148	0.9390
	Enhanced Representation	0.7096	0.7715	0.6571	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 (Pennekamp et al. 2019; Huang and Fu 2019).



(a) ETTh1(H = 720) (b) PEMS08(H = 12)

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

fusing the most relevant temporal information according to the given supervision for prediction.

Finally, we conduct experiments on ETTh1 dataset (with the multivariate experimental setting) to evaluate the impact of K (number of stacks) and L (number of levels), under various look-back window sizes T. The prediction horizon is fixed to be 24.

As can be observed from Table 6, when fixing $K = 1$, larger L leads to better prediction accuracy for the cases with larger T ($T = 128$ or 192). This is because we could further extract essential information from coarser temporal resolutions with deeper levels in the SCINet when T is large. As for the number of stacks K, when fixing $L = 3$, if T is small (e.g. $T = 24$ or 48), we find that increasing K would improve prediction accuracy. This is because, under such circumstances, the information extracted from a single SCINet is insufficient. By stacking more SCINets, we effectively increase the representation learning capability of the model, which facilitates extracting more robust temporal relations for the forecasting task. However, when T is large (e.g., 192), a shallow stack can already well capture the temporal dependencies for the time series. Under such circumstances, using deeper stacks may suffer from overfitting issues with the increase of parameters, which degrades the performance in the inference stage.

From Table 6, we can observe a clear trade-off between L and K. Moreover, the performance variation under different T also indicates the importance of the look-back window selection for forecasting tasks. While T is typically pre-determined based on domain knowledge about the time series data, based on our empirical study, $L \leq 5$ and $K \leq 3$ are usually sufficient and tuning these hyperparameters does not incur much effort.

5 Limitations 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 time series might contain noisy data, missing data or collected at irregular time intervals, 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 learning procedure, but it might be affected by the missing data if the ratio exceeds a certain threshold, wherein the downsampling-based multi-resolution sequence representation in SCINet may introduce biases, leading to poor prediction performance. The proposed downsampling mechanism may also have difficulty handling data collected at irregular intervals. We plan to take the above issues into consideration in the future development of SCINet.

Moreover, this work focuses on the deterministic time series forecasting problem. Many application scenarios require probabilistic forecasts, and we plan to revise SCINet to generate such prediction results.

Finally, while SCINet generates promising results for spatial-temporal time series without modeling spatial relations, we believe prediction accuracy can be further improved by incorporating such models. We plan to investigate such solutions in our future work.

6 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 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. Extensive

Table 6: The impact of L and K on MSE.

Number of Levels & Stacks	Horizon T	24	48	96	128	192
Level L (K =1)	2	0.411	0.348	0.347	0.334	0.384
	3	**0.405**	**0.346**	**0.316**	0.418	0.330
	4	-	0.360	0.340	0.331	**0.325**
	5	-	-	0.354	**0.323**	0.356
Stack K (L = 3)	1	0.405	0.346	**0.316**	0.418	**0.330**
	2	0.423	0.344	0.344	**0.339**	0.375
	3	**0.374**	**0.341**	0.345	0.353	0.363
	4	0.390	0.342	0.335	0.356	0.388

- Dash denotes the input cannot be further splitted.

experiments on various real-world TSF datasets demonstrate the superiority of our model over state-of-the-art methods.

References

Bahadori, M. T.; and Lipton, Z. C. 2019. Temporal-Clustering Invariance in Irregular Healthcare Time Series. *ArXiv*, abs/1904.12206.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bai, L.; Yao, L.; Li, C.; Wang, X.; and Wang, C. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *NeurIPS*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018a. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv*, abs/1803.01271.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018b. Trellis Networks for Sequence Modeling. In *ICLR*.

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

Borovykh, A.; Bohte, S.; and Oosterlee, C. W. 2017. Conditional Time Series Forecasting with Convolutional Neural Networks. *stat*, 1050: 16.

Box, G.; Jenkins, G.; and Macgregor, J. 1968. Some Recent Advances in Forecasting and Control. *Journal of The Royal Statistical Society Series C-applied Statistics*, 17: 158–179.

Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway Performance Measurement System: Mining Loop Detector Data. *Transportation Research Record*, 1748: 102 – 96.

Cho, K.; van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *SSST-8*.

Cui, Z.; Chen, W.; and Chen, Y. 2016. Multi-Scale Convolutional Neural Networks for Time Series Classification. *ArXiv*, abs/1603.06995.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9: 1735–1780.

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

Huang, R.; Huang, C.; Liu, Y.; Dai, G.; and Kong, W. 2020. LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks. In *IJCAI*.

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

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

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2019. Reformer: The Efficient Transformer. In *ICLR*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long- and short-term temporal patterns with deep neural networks. In *SIGIR*.

Li, M.; and Zhu, Z. 2021. Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. In *AAAI*.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

Lim, B.; Arık, S. Ö.; Loeff, N.; and Pfister, T. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*.

Lim, B.; and Zohren, S. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Trans. of the Royal Society A*.

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

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

Nguyen, N.; and Quanz, B. 2021. Temporal Latent Auto-Encoder: A Method for Probabilistic Multivariate Time Series Forecasting. In *AAAI*.

Oreshkin, B.; Carpo, D.; Chapados, N.; and Bengio, Y. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR*.

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

Rangapuram, S. S.; Seeger, M. W.; Gasthaus, J.; Stella, L.; Wang, Y.; and Januschowski, T. 2018. Deep state space models for time series forecasting. *NeurIPS*.

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

Sen, R.; Yu, H.-F.; and Dhillon, I. 2019. Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting. In *NeurIPS*.

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

Song, C.; Lin, Y.; Guo, S.; and Wan, H. 2020. Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting. In *AAAI*.

Taylor, S. J.; and Letham, B. 2018. Forecasting at Scale. *The American Statistician*, 72: 37 – 45.

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. WaveNet: A Generative Model for Raw Audio. In *The 9th ISCA Speech Synthesis Workshop*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All You Need. In *NeurIPS*.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In *KDD*.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*.

Yu, T.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*.

Zhou, H.-Y.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI*.