SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction

Anonymous Author(s)
Affiliation
Address
email

Abstract

One unique property of time series is that the temporal relations are largely preserved after downsampling into two sub-sequences. By taking advantage of this property, we propose a novel neural network architecture that conducts sample convolution and interaction for temporal modeling and forecasting, named **SCINet**. Specifically, SCINet is a recursive *downsample-convolve-interact* architecture. In each layer, we use multiple convolutional filters to extract *distinct yet valuable* temporal features from the downsampled sub-sequences or features. By combining these rich features aggregated from multiple resolutions, SCINet effectively models time series with complex temporal dynamics. Experimental results show that SCINet achieves significant forecasting accuracy improvements over both existing convolutional models and Transformer-based solutions across various real-world time series forecasting datasets. Our codes and data are available at https://anonymous.4open.science/r/SCINet-2588.

1 Introduction

Time series forecasting (TSF) enables decision-making with the estimated future evolution of metrics or events, thereby playing a crucial role in various scientific and engineering fields such as healthcare [1], energy management [37], traffic flow [37], and financial investment [9], to name a few.

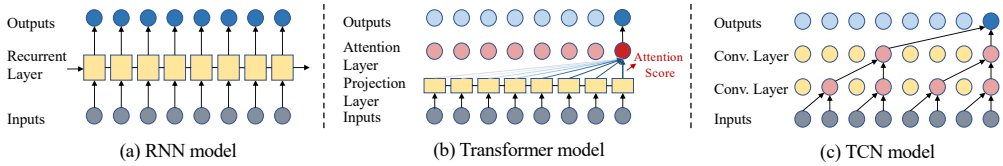
There are mainly three kinds of deep neural networks used for sequence modeling, and they are all applied for time series forecasting [21]: (i). recurrent neural networks (RNNs) [11]; (ii). Transformer-based models [33]; and (iii). temporal convolutional networks (TCN) [3].

Despite the promising results of TSF methods based on these generic models, they do not consider the specialty of time series data during modeling. For example, one unique property of time series is that the temporal relations (e.g., the trend and the seasonal components of the data) are largely preserved after downsampling into two sub-sequences. Consequently, by recursively downsampling the time series into sub-sequences, we could obtain a rich set of convolutional filters to extract dynamic temporal features at multiple resolutions.

Motivated by the above, in this paper, we propose a novel neural network architecture for time series modeling and forecasting, named *sample convolution and interaction network* (*SCINet*). The main contributions of this paper are as follows:

- We propose SCINet, a hierarchical *downsample-convolve-interact* TSF framework that effectively models time series with complex temporal dynamics. By iteratively extracting and exchanging information at multiple temporal resolutions, an effective representation with enhanced predictability can be learned, as verified by its comparatively lower permutation entropy (PE) [14].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



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

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

- 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.

Extensive experiments on various real-world TSF datasets show that our model consistently outperforms existing TSF approaches by a considerable margin. Moreover, while SCINet does not explicitly model spatial relations, it achieves competitive forecasting accuracy on spatial-temporal TSF tasks.

2 Related Work and Motivation

The time series forecasting problem is defined as: 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 forecasted data, respectively.

2.1 Related Work

Traditional time series forecasting methods such as the autoregressive integrated moving average (ARIMA) model [7] and Holt-Winters seasonal method [12] have theoretical guarantees. However, they are mainly applicable for univariate forecasting problems, restricting their applications to complex time series data. With the increasing data availability and computing power in recent years, it is shown that deep learning-based TSF techniques have the potential to achieve better forecasting accuracy than conventional approaches [21, 26].

Earlier RNN-based TSF methods [28, 29] summarize the past information compactly in the internal memory states that are recursively updated with new inputs at each time step, as shown in Fig. 1(a). The gradient vanishing/exploding problems and the inefficient training procedure greatly restrict the application of RNN-based models.

In recent years, Transformer-based models [33] have taken the place of RNN models in almost all sequence modeling tasks, thanks to the effectiveness and efficiency of the self-attention mechanisms. Various Transformer-based TSF methods (see Fig. 1(b)) are proposed in the literature [19, 20, 34, 22]. These works typically focus on the challenging long-term time series forecasting problem, taking advantage of their remarkable long sequence modeling capabilities.

Another popular type of TSF model is the so-called temporal convolutional network [6, 3, 30, 35, 24], wherein convolutional filters are used to capture local temporal features (see Fig. 1(c)). The proposed SCINet is also constructed based on temporal convolution. However, our method has several key differences compared with the TCN model based on dilated causal convolution, as discussed in the following.

2.2 Rethinking Dilated Causal Convolution for Time Series Modeling and Forecasting

The local correlation of time series data is reflected in the continuous changes within a time slot, and convolutional filters can effectively capture such local features. Consequently, convolutional neural networks are explored in the literature for time series modeling and forecasting. In particular, dilated causal convolution (DCS) is the current *de facto* method used in this respect.

DCS was first proposed for generating raw audio waveforms in WaveNet [25]. Later, [3] simplifies the WaveNet architecture to the so-called temporal convolutional networks (see Fig. 1 (c)). TCN consists of a stack of causal convolutional layers with exponentially enlarged dilation factors, which can achieve a large receptive field with just a few convolutional layers. Over the years, TCN has been widely used in all kinds of time series forecasting problems and achieve promising results [35, 30]. Moreover, convolutional filters can work seamlessly with graph neural networks (GNNs) to solve various spatial-temporal TSF problems.

With *causal convolutions* in the TCN architecture, an output i is convolved only with the ith 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 autoregressive 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, there is no need to use causal convolutions. We can safely apply *normal convolutions* on the look-back window for forecasting.

More importantly, the dilated architecture in TCN has two inherent limitations:

- A single convolutional filter is shared within each layer. Such a unified convolutional kernel tends to extract the average temporal features from the data/features in the previous layer. However, complex time series may contain substantial temporal dynamics. Hence, it is essential to extract distinct yet valuable features with a rich set of convolutional filters.
- While the final layer of the TCN model has the global view of the entire look-back window, the effective receptive fields of the intermediate layers (especially those close to the inputs) are limited, causing temporal relation loss during feature extraction.

The above limitations of the TCN architecture motivate the proposed SCINet design, as detailed in the following section.

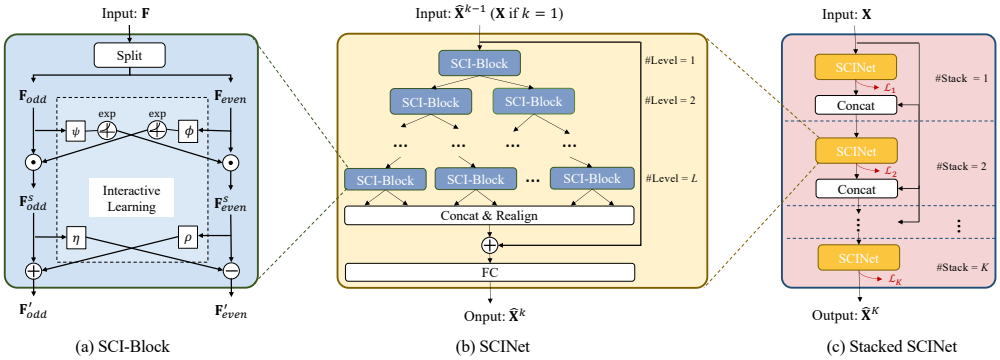


Figure 2: The overall architecture of Sample Convolution and Interaction Network (SCINet).

3 SCINet: Sample Convolution and Interaction Network

SCINet adopts an encoder-decoder architecture. The encoder is a hierarchical convolutional network that captures dynamic temporal dependencies at multiple resolutions with a rich set of convolutional filters. 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 processes each sub-sequence with a set of convolutional filters to extract distinct yet valuable temporal features 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)). A distinctive advantage of such design is that each SCI-Block has both local and global views of the entire time series, thereby facilitating the extraction of useful temporal features. After all the downsample-convolve-interact operations, we realign the extracted features into a new sequence representation and add it to the original time series for forecasting with a fully-connected network as the decoder. To facilitate extracting complicated temporal patterns, 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 distinct yet valuable temporal relations 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[1] $\mathbf{F}^s_{even}$ 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 supplementary materials.

Compared to the dilated convolutions used in the TCN architecture, the proposed downsample-convolve-interact architecture achieves an even larger receptive field at each convolutional layer. More importantly, unlike TCN that employs a single shared convolutional filter at each layer, significantly restricting its feature extraction capabilities, SCI-Block aggregates essential information extracted from the two downsampled sub-sequences that have both local and global views of the entire time series.

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. Within the k-th SCINet of the stacked SCINet (Section 3.3), the input time series $\mathbf{X}$ (for $k=1$) or feature vector $\hat{\mathbf{X}}^{k-1} = \{\hat{\mathbf{x}}^{k-1}_1, ..., \hat{\mathbf{x}}^{k-1}_\tau\}$ (for $k>1$) is gradually downsampled 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 temporal dependencies in the time series.

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. It is then added to the original time series through a residual connection [10] to generate a new sequence with enhanced predictability. Finally, a simple fully-connected network is used to decode the enhanced sequence representation into $\hat{\mathbf{X}}^k = \{\hat{\mathbf{x}}^k_1, ..., \hat{\mathbf{x}}^k_\tau\}$.

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

3.3 Stacked SCINet

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

Specifically, we apply *intermediate supervision* [4] on the output of each SCINet using the ground-truth values, to ease the learning of the 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 forecasting results.

3.4 Loss Function

To train a stacked SCINet with K ($K \geq 1$) SCINets, the loss of the k-th prediction results 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:

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

3.5 Complexity Analysis

Thanks to the downsampling procedure, the neurons at each convolutional layer of SCINet have a larger receptive field than those of TCN. More importantly, the set of rich convolutional filters in SCINet enable flexible extraction of temporal features from multiple resolutions. Consequently, SCINet usually does not require downsampling the original sequence to the coarsest level for effective forecasting. Given the look-back window size T, TCN generally requires $\lceil \log_2 T \rceil$ layers when the dilation factor is 2, while the number of layers L in SCINet could be much smaller than $\log_2 T$. Our empirical study shows that the best forecasting accuracy is achieved with $L \leq 5$ in most cases even with large T (e.g., 168). As for the number of stacks K, our empirical study also shows that $K \leq 3$ would be sufficient.

Consequently, the computational cost of SCINet is usually on par with that of the TCN architecture. The worst-case time complexity is $\mathcal{O}(T \log T)$, much less than that of vanilla Transformer-based solutions: $\mathcal{O}(T^2)$.

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*, *network structures and their hyper-parameters* are shown in the supplementary materials.

4.1 Datasets

We conduct experiments on 9 popular time series datasets: (1) *Electricity Transformer Temperature* [37] (ETTh1) (2) *Traffic* (3) *Solar-Energy* (4) *Electricity* (5) *Exchange-Rate* (6) *PeMS* (*PEMS03, PEMS04, PEMS07 and PEMS08*). A brief description of these datasets is listed in Table 1. All the experiments on these datasets in this section are conducted under multi-variate TSF setting.

To make a fair comparison, we follow existing experimental settings, and use the same evaluation metrics as the original publications [15, 23, 36, 17] in each dataset.

Table 1: The overall information of the 9 datasets.

Datasets	ETTh1	Traffic	Solar-Energy	Electricity	Exchange-Rate	PEMS03	PEMS04	PEMS07	PEMS08
Variants	7	862	137	321	8	358	307	883	170
Timesteps	17,420	17,544	52,560	26,304	7,588	26,209	16,992	28,224	17,856
Granularity	1hour	1hour	10min	1hour	1day	5min	5min	5min	5min
Start time	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	Single-step	Single-step	Single-step	Single-step	Multi-step	Multi-step	Multi-step	Multi-step
Data partition	Follow [37]	Training/Validation/Testing: 6/2/2				Training/Validation/Testing: 6/2/2			

4.2 Results and Analyses

Table 2, Table 3 and Table 4 provide the main experimental results of SCINet. We observe that SCINet shows superior performance than other TSF models on various tasks, including short-term, long-term and spatial-temporal time series forecasting.

Short-term Time Series Forecasting: we evaluate the performance of the SCINet in short-term TSF tasks with other baseline methods on *Traffic*, *Solar-Energy*, *Electricity* and *Exchange-Rate* datasets. The experimental setting is the same as [17], which uses the input length of 168 to forecast different future horizons$\{3, 6, 12, 24\}$.

As can be seen in Table 2, the proposed SCINet outperforms existing RNN/TCN-based (LSTNet [17], TPA-LSTM [31], TCN [3], TCN†) and Transformer-based [34, 37, 33] TSF solutions in most cases, especially for the Solar-Energy and Exchange-Rate datasets. Note that, TCN† denotes a variant of TCN wherein causal convolutions are replaced by normal convolutions, and improves the original TCN across all the datasets, which supports our claim in Sec. 2.2. Moreover, we can also observe that the Transformer-based methods have poor performance in this task. For short-term forecasting, the recent data points are typically more important for accurate forecasting. However, the permutation-invariant self-attention mechanisms used in Transformer-based methods do not pay much attention to such critical information. In contrast, the general sequential models (RNN/TCN) can formulate it easily, showing quite competitive results in short-term forecasting.

Table 2: Short-term forecasting performance comparison on the four datasets. The best results are shown in bold and second best results are highlighted with underlined blue font. The red percentage shows the RSE improvement over the second best model.

Model		**SCINet**		Autoformer		Informer		Transformer		*TCN		*TCN†		LSTNet		TPR-LSTM		
Metric	τ	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	RSE	CORR	**IMP**
Solar-Energy	3	**0.1775**	**0.9853**	N/A	N/A	N/A	N/A	N/A	N/A	0.1940	0.9835	0.1900	0.9848	0.1843	0.9843	0.1803	0.9850	1.55%
	6	**0.2301**	**0.9739**	N/A	N/A	N/A	N/A	N/A	N/A	0.2581	0.9602	0.2382	0.9612	0.2559	0.9690	0.2347	0.9742	1.96%
	12	**0.2997**	**0.9550**	N/A	N/A	N/A	N/A	N/A	N/A	0.3512	0.9321	0.3353	0.9432	0.3254	0.9467	0.3234	0.9487	7.33%
	24	**0.4081**	**0.9112**	N/A	N/A	N/A	N/A	N/A	N/A	0.4732	0.8812	0.4676	0.8851	0.4643	0.8870	0.4389	0.9081	7.02%
Traffic	3	**0.4216**	**0.8920**	0.593	0.294	0.563	0.306	0.552	0.306	0.5459	0.8486	0.5361	0.8540	0.4777	0.8721	0.4487	0.8812	6.04%
	6	**0.4414**	**0.8809**	0.612	0.289	0.569	0.306	0.572	0.304	0.6061	0.8205	0.5992	0.8197	0.4893	0.8690	0.4658	0.8717	5.24%
	12	**0.4495**	**0.8772**	0.670	0.269	0.574	0.302	0.570	0.308	0.6367	0.8048	0.6061	0.8205	0.4950	0.8614	0.4641	0.8717	3.15%
	24	**0.4453**	**0.8825**	0.585	0.293	0.589	0.306	0.573	0.302	0.6586	0.7921	0.6456	0.7982	0.4973	0.8588	0.4765	0.8629	6.55%
Electricity	3	**0.0740**	**0.9494**	0.170	0.468	0.140	0.470	0.130	0.467	0.0892	0.9232	0.0852	0.9293	0.0864	0.9283	0.0823	0.9439	10.09%
	6	**0.0845**	**0.9387**	0.163	0.463	0.158	0.468	0.133	0.467	0.0974	0.9121	0.0924	0.9235	0.0931	0.9135	0.0916	0.9337	7.75%
	12	**0.0929**	**0.9305**	0.173	0.462	0.178	0.469	0.139	0.467	0.1053	0.9017	0.0993	0.9173	0.1007	0.9077	0.0964	0.9250	3.63%
	24	**0.0967**	**0.9270**	0.170	0.462	0.150	0.466	0.144	0.466	0.1091	0.9101	0.0989	0.9101	0.1007	0.9119	0.1006	0.9133	3.88%
Exchange Rate	3	**0.0171**	**0.9787**	0.066	0.260	0.148	0.267	0.153	0.268	0.0217	0.9693	0.0202	0.9712	0.0226	0.9735	0.0174	0.979	1.72%
	6	**0.0240**	0.9704	0.068	0.254	0.168	0.262	0.167	0.268	0.0263	0.9633	0.0257	0.9628	0.0280	0.9658	0.0241	**0.9709**	0.41%
	12	**0.0331**	0.9553	0.083	0.248	0.151	0.259	0.173	0.266	0.0393	0.9531	0.0352	0.9501	0.0356	0.9511	0.0341	**0.9564**	2.93%
	24	**0.0436**	**0.9396**	0.090	0.252	0.197	0.256	0.187	0.263	0.0492	0.9223	0.0487	0.9314	0.0449	0.9354	0.0444	0.9381	1.80%

- Autoformer, Informer and Transformer achieved by Autoformer [36] requires pre-prossessed datasets for training.
- N/A denotes no pre-prossessed dataset for training.
- $*$ denotes re-implementation. † denotes the variant with normal convolutions.

Long-term Time Series Forecasting: many real-world applications also require to predict long-term events. Therefore, we conduct the experiments on *ETT*, *Exchange Rate, Electricity and Traffic* datasets to evaluate the performance of SCINet on long-term TSF tasks. In this experiment, we only compare SCINet with Transformer-based methods [34, 16, 19, 37, 33], since they are more popular in recent long-term TSF research.

As can be seen from Table 3, the SCINet achieves state-of-the-art performances in most benchmarks and prediction length settings (except ETTh1 (720)). Overall, SCINet yields 21% average improvements on MSE among the above settings. In particular, for Exchange-Rate, compared to previous state-of-the-art results, SCINet gives average 65% improvements on MSE. We attribute it to that the proposed SCINet can better capture both short (*local temporal dynamics*)- and long (*trend, seasonality*)-term temporal dependencies to make an accurate prediction in long-term TSF. Besides, compared with the vanilla Transformer-based methods [16, 19, 37], the newly-proposed Transformer-based forecasting model Autoformer [34] achieves the second best performance in all experimental settings

and also surpasses the SCINet in ETTh1(720). This is because, Autoformer incorporates more prior knowledge about the time series data. It focuses on modeling seasonal patterns and conducts self-attention at the sub-series level (instead of the raw data), which is much better in extracting long-term temporal patterns than vanilla Transformer-based methods.

Table 3: Long-term forecasting performance comparison with Transformer-based models on the four datasets. The best results are shown in bold and second best results are highlighted with underlined blue font. The red percentage shows the MSE improvement over the second best model.

Model		**SCINet**		Autoformer		Informer		Transformer		LogTrans		Reformer		
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	**IMP**
ETTh1	24	**0.341**	**0.379**	0.384	0.425	0.577	0.549	0.607	0.581	0.686	0.604	0.991	0.754	11.20%
	48	**0.368**	**0.395**	0.392	0.419	0.685	0.625	0.850	0.722	0.766	0.757	1.313	0.906	6.12%
	168	**0.451**	**0.457**	0.490	0.481	0.931	0.752	1.038	0.832	1.002	0.846	1.824	1.138	7.96%
	336	**0.502**	**0.497**	0.505	0.484	1.128	0.873	1.118	0.864	1.362	0.952	2.117	1.280	0.59%
	720	0.583	0.560	**0.498**	**0.500**	1.215	0.896	1.049	0.812	1.397	1.291	2.415	1.520	-17.07%
Exchange Rate	96	**0.061**	**0.188**	0.197	0.323	0.847	0.752	0.559	0.587	0.968	0.812	1.065	0.829	68.98%
	192	**0.106**	**0.244**	0.300	0.369	1.204	0.895	1.168	0.835	1.040	0.851	1.188	0.906	64.70%
	336	**0.181**	**0.323**	0.509	0.524	1.672	1.036	1.423	0.949	1.659	1.081	1.357	0.976	64.36%
	720	**0.525**	**0.571**	1.447	0.941	2.478	2.478	2.160	1.150	1.941	1.127	1.510	1.016	63.72%
Electricity	96	**0.171**	**0.256**	0.201	0.317	0.274	0.368	0.263	0.359	0.258	0.357	0.312	0.402	14.93%
	192	**0.177**	**0.265**	0.222	0.334	0.296	0.296	0.273	0.374	0.266	0.368	0.348	0.433	20.27%
	336	**0.197**	**0.285**	0.231	0.338	0.300	0.394	0.277	0.373	0.280	0.380	0.350	0.433	14.72%
	720	**0.234**	**0.318**	0.254	0.361	0.373	0.439	0.290	0.378	0.283	0.376	0.340	0.420	7.87%
Traffic	96	0.613	0.395	0.613	0.388	0.719	0.391	0.638	0.354	0.684	0.384	0.732	0.423	0.00%
	192	**0.559**	**0.363**	0.616	0.382	0.696	0.379	0.647	0.354	0.685	0.390	0.733	0.420	9.25%
	336	**0.555**	**0.358**	0.622	0.337	0.777	0.420	0.669	0.364	0.733	0.408	0.742	0.420	10.77%
	720	-	-	0.660	0.408	0.864	0.472	0.707	0.386	0.717	0.396	0.755	0.423	-

- dash indicates failure due to out-of-memory.

Spatial-temporal Time Series Forecasting: besides the general TSF task, there is also a large category of data related to spatial-temporal forecasting. For example, traffic datasets *PeMS* [8] are complicated spatial-temporal time series and they have been investigated for decades. Most recent approaches [35, 32, 2, 13, 18] 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, which further proves the superb temporal modeling capabilities of SCINet.

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	-	**14.98**	6.26%
	MAPE	21.33	19.93	18.63	18.91	17.15	19.40	19.31	16.78	16.30	*15.23	-	**14.11**	7.36%
	RMSE	35.11	33.55	32.24	30.31	30.12	29.66	32.94	29.21	28.34	*28.25	-	**24.08**	8.37%
PEMS04	MAE	25.14	23.22	22.81	24.70	22.70	22.93	25.45	21.19	19.83	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	13.02	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	31.88	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	22.07	*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	35.80	*36.55	-	**34.03**	6.89%
PEMS08	MAE	22.20	22.72	21.42	17.86	18.02	18.61	19.13	17.13	16.64	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	26.22	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.

Predictability estimation: inspired by [14, 27], we use *permutation entropy* (*PE*) [5] 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[2]. The PE values of the original time series and the corresponding enhanced representations are shown in Table 5.

Table 5: Permutation entropy comparison before and after SCINet.

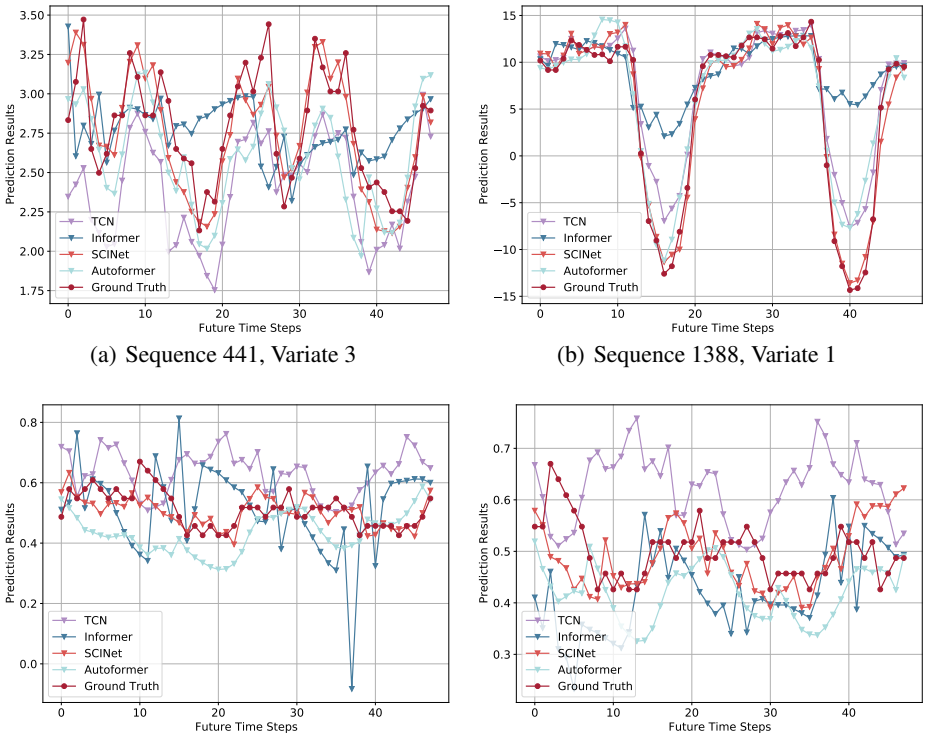
Permutation Entropy		Datasets								
		ETTh1	Traffic	Solar-Energy	Electricity	Exc-rate	PEMS03	PEMS04	PEMS07	PEMS08
Parameters	m (τ = 1)*	6	6	7	6	6	6	6	6	6
Value	Original Input	0.8878	0.9371	0.4739	0.9489	0.8260	0.9649	0.9203	0.9148	0.9390
	Enhanced Representation	0.7096	0.8832	0.3537	0.8901	0.7836	0.8377	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 [27, 14].

[2]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.

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.

Visualization: we present the qualitative results on some randomly selected sequences of the ETTh1 dataset in Fig. 3, which clearly shows the superior performance of of SCINet in obtaining the trend and seasonality of time series.



(a) Sequence 441, Variate 3 (b) Sequence 1388, Variate 1

(c) Sequence 2745, Variate 4 (d) Sequence 2753, Variate 5

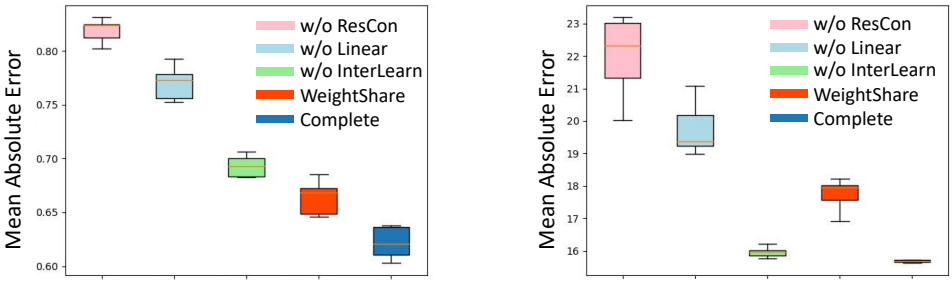
Figure 3: The prediction results (Horizon = 48) of SCINet, Autoformer, Informer, and TCN on randomly-selected sequences from ETTh1 dataset.

4.3 Ablation studies

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

SCIBlock: 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 sub-sequences when there are longer look-back windows for such interactions.



(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.

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

We also conduct comprehensive ablation studies on the impact of K (number of stacks) and L (number of levels), and the selection of operator in the interact learning mechanism. These results are shown in the supplementary materials due to space limitation.

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 explicitly modeling spatial relations, the forecasting accuracy could be further improved by incorporating dedicated spatial 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 modeling and forecasting, motivated by the unique properties of time series data compared to generic sequence data. The proposed SCINet is a hierarchical downsample-convolve-interact structure with a rich set of convolutional filters. It iteratively extracts and exchanges information at different temporal resolutions and learns an effective representation with enhanced predictability. Extensive experiments on various real-world TSF datasets demonstrate the superiority of our model over state-of-the-art methods.

References

[1] Mohammad Taha Bahadori and Zachary Chase Lipton. Temporal-clustering invariance in irregular healthcare time series. *arXiv preprint arXiv:1904.12206*, 2019.

[2] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33:17804–17815, 2020.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. In *ICLR*, 2018.

[5] Christoph Bandt and Bernd Pompe. Permutation entropy: a natural complexity measure for time series. *Physical review letters*, 88(17):174102, 2002.

[6] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *stat*, 1050:16, 2017.

[7] George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

[8] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1):96–102, 2001.

[9] Pierpaolo D'Urso, Livia De Giovanni, and Riccardo Massari. Trimmed fuzzy clustering of financial time series based on dynamic time warping. *Annals of operations research*, 299(1):1379–1395, 2021.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[13] Rongzhou Huang, Chuyin Huang, Yubao Liu, Genan Dai, and Weiyang Kong. Lsgcn: Long short-term traffic prediction with graph convolutional networks. In *IJCAI*, pages 2355–2361, 2020.

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

[15] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[16] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2019.

[17] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *SIGIR*, 2018.

[18] Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *AAAI*, 2021.

[19] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS*, 2019.

[20] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 2021.

[21] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[22] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2021.

[23] Spyros Makridakis, Allan Andersen, Robert Carbone, Robert Fildes, Michele Hibon, Rudolf Lewandowski, Joseph Newton, Emanuel Parzen, and Robert Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting*, 1(2):111–153, 1982.

[24] Nam Nguyen and Brian Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *AAAI*, 2021.

[25] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 2016.

[26] Boris Oreshkin, Dmitri Carpo, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR*, 2020.

[27] Frank Pennekamp, Alison C Iles, Joshua Garland, Georgina Brennan, Ulrich Brose, Ursula Gaedke, Ute Jacob, Pavel Kratina, Blake Matthews, Stephan Munch, et al. The intrinsic predictability of ecological time series and its potential to guide forecasting. *Ecological Monographs*, 89(2):e01359, 2019.

[28] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *NeurIPS*, 2018.

[29] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[30] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. volume 32, 2019.

[31] Shun-Yao Shih, Fan-Keng Sun, and Hung yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, pages 1–21, 2019.

[32] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference*

on Artificial Intelligence, volume 34, pages 914–921, 2020.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[34] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

[35] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and C. Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, 2019.

[36] Jiehui Xu, Jianmin Wang, Mingsheng Long, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34, 2021.

[37] Hao-Yi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 5
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the Appendix and link in abstract.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Detail seen in Appendix.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]