Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang and Xifeng Yan
University of California, Santa Barbara

Contribution:

1. We successfully apply Transformer architecture to time series forecasting and perform extensive experiments on both synthetic and real datasets to validate Transformer's potential value in better handling long-term dependencies than RNN-based models.

2. We propose *convolutional* self-attention by employing causal convolutions to produce queries and keys in the self-attention layer. Query-key matching aware of local context, e.g. shapes, can help the model achieve lower training loss and further improve its forecasting accuracy.

3. We propose Log-Sparse Transformer, with only $O(L(\log L)^2)$ space complexity to break the memory bottleneck, not only making fine-grained long time series modeling feasible but also producing comparable or even better results with much less memory usage, compared to canonical Transformer.

Problem Setup:

Suppose we have a collection of *N* related univariate time series $\{\mathbf{z}_{i,1:t_0}\}_{i=1}^N$,where $\mathbf{z}_{i,1:t_0} =: [\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, \cdots, \mathbf{z}_{i,t_0}]$ and $\mathbf{z}_{i,t} \in \mathbb{R}$ denotes the value of time series *i* in tine *t*. We are going to predict the next τ time steps for all time series, i.e. $\{\mathbf{z}_{i,t_0+1:t_0+\tau}\}_{i=1}^N$. Besides, let $\{\mathbf{x}_{i,1:t_0+\tau}\}_{i=1}^N$ be a set of associated time-based covariate vectors that are assumed to be known over the entire time period, e.g. day-of-the-week and hour-of-the-day. We aim to model the following conditional distribution:

$$p(\mathbf{z}_{i,t_0+1:t_0+\tau}|\mathbf{z}_{i,1:t_0}, \mathbf{x}_{i,1:t_0+\tau}; \mathbf{\Phi}) = \prod_{t=t_0+1}^{t_0+\tau} p(\mathbf{z}_{i,t}|\mathbf{z}_{i,1:t-1}, \mathbf{x}_{i,1:t}; \mathbf{\Phi})$$

We propose to model such distribution by the autoregressive Transformer decoder.

Methodology:

1. Enhancing the locality of Transformer

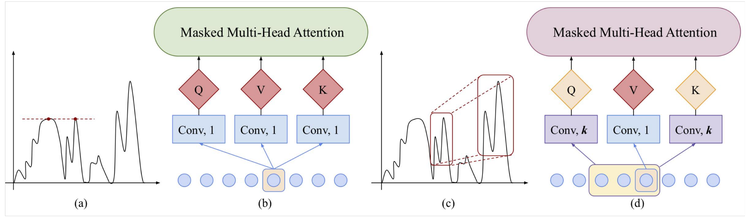


Figure 1: The comparison between canonical and our convolutional self-attention layers.

2. Breaking the memory bottleneck of Transformer

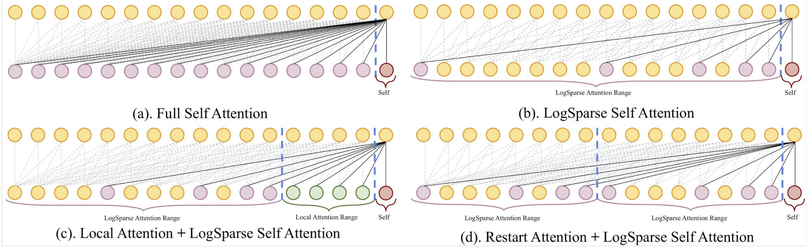
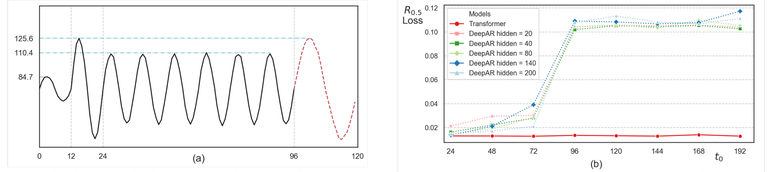


Figure 2: Illustration of different attention mechanism between adjacent layers in Transformer.

Theorem: *$\forall\, l$ and $j \le l$, there is at least one path from cell j to cell l if we stack $\lfloor log_2\, l \rfloor + 1$ layers. Moreover, for $j < l$, the number of feasible unique paths from cell j to cell l increases at a rate of $O(\lfloor log_2\, (l-j) \rfloor !)$.*

Experiments:

Synthetic datasets



(a) An example time series with $t_0 = 96$. The larger t_0 is, the longer dependencies models need to capture for accurate forecasting. (b) Performance comparison between *DeepAR* and canonical *Transformer* along with the growth of t_0.

Real-world datasets

Long-term and short-term forecasting

	ARIMA	ETS	TRMF	DeepAR	DeepState	Ours
electricity~c_{1d}	0.154/0.102	0.101/0.077	0.084/-	0.075/0.040	0.083/0.056	**0.059/0.034**
electricity~c_{7d}	0.283/0.109	0.121/0.101	0.087/-	0.082/0.053	0.085/0.052	**0.070/0.044**
traffic~c_{1d}	0.223/0.137	0.236/0.148	0.186/-	0.161/0.099	0.167/0.113	**0.122/0.081**
traffic~c_{7d}	0.492/0.280	0.509/0.529	0.202/-	0.179/0.105	0.168/0.114	**0.139/0.094**

Results summary ($R_{0.5}/R_{0.9}$-loss) of all methods.

Convolutional self-attention

	k=1	k=2	k=3	k=6	k=9
electricity~c_{1d}	0.060/**0.030**	0.058/**0.030**	**0.057**/0.031	**0.057**/0.031	0.059/0.034
traffic~c_{1d}	0.134/0.089	0.124/0.085	0.123/0.083	0.123/0.083	0.122/0.081

Average $R_{0.5}/R_{0.9}$ -loss of different kernel sizes for rolling-day prediction of 7 days.

Sparse attention

Constraint	Dataset	Full	Sparse	Full + Conv	Sparse + Conv
Memory	*electricity*~f_{1d}	0.083/0.051	0.084/**0.047**	**0.078**/0.048	0.079/0.049
	traffic~f_{1d}	0.161/0.109	0.150/0.098	0.149/0.102	**0.138/0.092**
Length	*electricity*~c_{1d}	0.082/0.047	0.084/0.047	**0.074/0.042**	0.079/0.049
	traffic~f_{1d}	0.147/0.096	0.150/0.098	0.139/**0.090**	**0.138**/0.092

Average $R_{0.5}/R_{0.9}$ -loss comparisons between sparse attention and full attention models with/without convolutional self-attention.