Artificial Intelligence 318 (2023) 103886

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

Expanding the prediction capacity in long sequence time-series forecasting[☆]

Haoyi Zhou[a], Jianxin Li[a,*], Shanghang Zhang[b], Shuai Zhang[a], Mengyi Yan[a], Hui Xiong[c]

[a] *Beihang University, Beijing, China*
[b] *Peking University, Beijing, China*
[c] *Hong Kong University of Science and Technology, Hong Kong, China*

A R T I C L E   I N F O

Article history:
Received 16 December 2021
Received in revised form 12 February 2023
Accepted 22 February 2023
Available online 28 February 2023

Keywords:
Time-series
Deep learning
Forecasting
Self-attention

A B S T R A C T

Many real-world applications show growing demand for the prediction of long sequence time-series, such as electricity consumption planning. Long sequence time-series forecasting (LSTF) requires a higher prediction capacity of the model, which is the ability to capture precise long-range dependency coupling between output and input efficiently. Recent studies have shown the potential of Transformer to accommodate the capacity requirements. However, three real challenges that may have prevented expanding the prediction capacity in LSTF are that the Transformer is limited by quadratic time complexity, high memory usage, and slow inference speed under the encoder-decoder architecture. To address these issues, we design an efficient transformer-based model for LSTF, named Informer, with three distinctive characteristics. (i) a *ProbSparse* self-attention mechanism, which achieves $\mathcal{O}(L \log L)$ in time complexity and memory usage, and has comparable performance on sequences' dependency alignment. (ii) the self-attention distilling promotes dominating attention by convolutional operators. Besides, the halving of layer width is intended to reduce the expense of building a deeper network on extremely long input sequences. (iii) the generative style decoder, while conceptually simple, predicts the long time-series sequences at one forward operation rather than a step-by-step way, which drastically improves the inference speed of long-sequence predictions. Extensive experiments on ten large-scale datasets demonstrate that Informer significantly outperforms existing methods and provides a new solution to the LSTF problem.

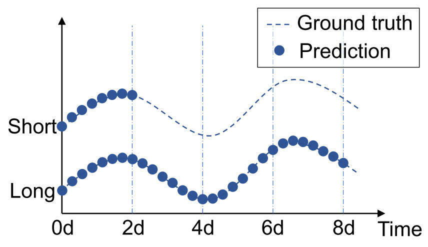
© 2023 Published by Elsevier B.V.

1. Introduction

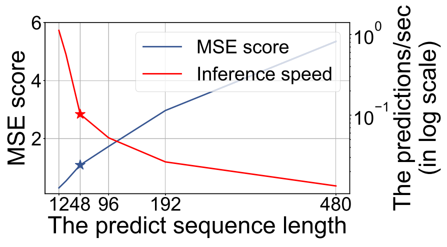
Time-series forecasting is a critical ingredient across many domains, such as sensor network monitoring [1], energy and smart grid management, economics and finance [2], and disease propagation analysis [3]. In these scenarios, we can leverage a substantial amount of time-series data on past behavior to make a forecast in the long run, namely long sequence time-series forecasting (LSTF). It is generally accepted that society places high value on targeting long-run trends rather than

[☆] This paper is an invited revision of a paper which first appeared at the 2021 AAAI Conference on Artificial Intelligence (AAAI-21).
[*] Corresponding author.
E-mail address: lijx@buaa.edu.cn (J. Li).

https://doi.org/10.1016/j.artint.2023.103886
0004-3702/© 2023 Published by Elsevier B.V.



(a) Sequence Forecasting.



(b) Run LSTM on sequences.

Fig. 1. (a) The long sequence time-series forecasting can cover an extended period than the short sequence predictions, which makes vital distinction in policy-planning and investment-protecting. (b) The prediction capacity of existing methods limits the long sequence's performance, i.e., starting from length=48, the MSE rises unacceptably high, and the inference speed drops rapidly.

the next-step status. For instance, the long-term business: forecasting the electricity consumption on clients in a long period helps to estimate the region transformer power load and manage the power supply [4]; the long-term climate: analyzing the rainfall change based on more than a hundred years' record helps to forecast the semi-century climate and build coping strategies [5]; the fine-grained management: forecasting the hourly product supply helps to optimize the inventory management, staff scheduling and topology planning, and is a crucial technology for most aspects of supply chain optimization [6]. However, prevailing forecasting methods are mostly designed under conservative settings, like predicting 48 points or less [7–12].

The increasingly long sequences strain the models' prediction capacity to the point where this trend is holding the research on LSTF problems. As an concrete example, we use the Long Short Term Memory (LSTM) network to predict the hourly temperature of an electrical transformer station. The longer forecasting helps to better estimate the power load and transformer states. We enlarge the prediction horizon from the short-term period (12 points, 0.5 days) to the long-term period (480 points, 20 days) in Fig. 1. The overall performance gap is substantial when the prediction length is greater than 48 points (the solid star in Fig. 1(b), where the mean squared error (MSE) rises to unsatisfactory performance, the inference speed gets sharp drop, and the LSTM model starts to fail.

The major challenge for LSTF is to expand the prediction capacity to meet the increasingly long sequence prediction demand, which requires **(a) extraordinary long-range alignment ability** and **(b) efficient operations on long sequence inputs and outputs**. Recently, Transformer models have shown superior performance in capturing long-range dependency than Recurrent Neural Network (RNN) models. The self-attention mechanism in Transformer can reduce the maximum length of network signals traveling paths into the theoretical shortest $\mathcal{O}(1)$ and avoid the recurrent structure, whereby Transformer shows great potential for the LSTF problem. Nevertheless, the self-attention mechanism violates requirement (b) due to its L-quadratic computation and memory consumption on L-length inputs/outputs. Some large-scale Transformer models pour resources and yield impressive results on NLP tasks [13], but the training on dozens of GPUs and expensive deploying cost make these models unaffordable on the real-world LSTF problems. The efficiency of the self-attention mechanism and Transformer architecture becomes the bottleneck of applying them to LSTF problems. Thus, in this paper, we seek to answer the question: *can we improve Transformer models to be computation, memory, and architecture-efficient, as well as maintaining higher prediction capacity?*

Vanilla Transformer [14] has three significant limitations when solving the LSTF problem:

1. **The quadratic computation of self-attention.** The atom operation of self-attention mechanism, namely canonical dot-product, causes the time complexity and memory usage per layer to be $\mathcal{O}(L^2)$.
2. **The memory bottleneck in stacking layers for long inputs.** The stack of J encoder/decoder layers makes the total memory usage to be $\mathcal{O}(J \cdot L^2)$, which limits the model's scalability in receiving long sequence inputs.
3. **The speed plunge in predicting long outputs.** The dynamic decoding of vanilla Transformer makes the step-by-step inference as slow as RNN-based model, as shown in Fig. 1(b).

There are some prior works on improving the efficiency of self-attention. The Sparse Transformer [15], LogSparse Transformer [16], and Longformer [17] all use a heuristic method to tackle the limitation 1 and reduce the complexity of self-attention mechanism to $\mathcal{O}(L \log L)$, where their efficiency gain is limited [18]. Reformer [19] also achieves $\mathcal{O}(L \log L)$ with locally-sensitive hashing self-attention, but it only works on extremely long sequences. More recently, Linformer [20] claims a linear complexity $\mathcal{O}(L)$, but the project matrix can not be fixed for real-world long sequence input, which may have the risk of degradation to $\mathcal{O}(L^2)$. Transformer-XL [21] and Compressive Transformer [22] use auxiliary hidden states to capture long-range dependency, which could amplify the limitation 1 and be adverse to break the efficiency bottleneck. All these works mainly focus on the limitation 1, and the limitation 2&3 remains unsolved in the LSTF problem. To expand the prediction capacity, we tackle all these limitations and achieve improvement beyond efficiency in the proposed Informer.

To this end, our work delves explicitly into these three issues. We investigate the sparsity in the self-attention mechanism, make improvements of network components, and conduct extensive experiments. The contributions of this paper are summarized as follows:

- We propose Informer to successfully expand the prediction capacity in the LSTF problem, which validates the Transformer-like model's potential value to capture individual long-range dependency between long sequence time-series outputs and inputs.
- We propose the *ProbSparse* self-attention mechanism to efficiently replace the canonical self-attention. It achieves the $\mathcal{O}(L\log L)$ time complexity and $\mathcal{O}(L\log L)$ memory usage on dependency alignments.
- We propose the self-attention distilling operation to privilege dominating attention scores in J-stacking layers and sharply reduce the total space complexity to be $\mathcal{O}((2-\epsilon)L\log L)$, which helps receiving long sequence input.
- We propose a generative style decoder to acquire long sequence output with only one forward step needed, simultaneously avoiding cumulative error spreading during the inference phase.

A preliminary version of this work appeared in the 35th proceedings of the AAAI Conference on Artificial Intelligence [23]. This journal version involves several improvements in enhancing the previous model from the following aspects. First, we give a new proof for the bound of the *ProbSparse* self-attention mechanism without large variance assumptions, further broadening the model's application scope. Second, we give new proof for the empirical approximation of the random sampling strategy in the query sparsity measurement, which further verifies its efficacy. Third, we propose a new mixed multi-head attention mechanism in the decoder, which improves the efficiency of the decoder in long prediction. Besides, we have conducted comprehensive experiments on more datasets and performed more detailed analyses for Informer.

2. Related work

We provide a literature review of the long sequence time-series forecasting (LSTF) problem below. LSTF problem is a special case of the time-series forecasting problem. So technically, the existing time-series forecasting models can be directly applied to the LSTF. However, there will be some disadvantages in the real-world execution or application, such as accuracy, speed and resource occupation. So, we show the related works including time-series forecasting methods, long sequence time-series input problems, attention models, and transformer-based models.

Time-series Forecasting. Existing methods for time-series forecasting can be roughly grouped into two categories: classical models and deep learning methods. Classical time-series models serve as a reliable workhorse for time-series forecasting, with appealing properties such as interpretability and theoretical guarantees [24,25]. Modern extensions include the support for missing data [26] and multiple data types [27]. Deep learning methods develop the prediction paradigm by applying RNN and their variants in a sequence-to-sequence manner, achieving cutting-edge performance [7–9]. Despite the substantial progress, existing algorithms still fail to predict long sequence time series with satisfying accuracy. Typical state-of-the-art approaches [26,27], especially deep learning ones [9,11,28,29,12], remain as a sequence to sequence prediction paradigm with step-by-step processes, which have the following limitations: (i) Even though they may be accurate for one-step prediction, they often suffer from the accumulated error through the dynamic decoding, resulting in the fail for LSTF problem [10,11]. The prediction accuracy decays along with the increase of the predicted sequence length. (ii) Due to the problem of vanishing gradient and memory constraint [30], most existing methods cannot learn from the past behavior of the whole history of the time-series. As discussed in the introduction section, we refer to the two limitations as the other side of the coin in expanding the prediction capacity. We design Informer to address the above limitations.

Besides, the long sequence time-series input (LSTI) problem also admits a long sequence of input data, which applies to the second limitation, but it has few requirements on the prediction capacity as it is in the LSTF problem. We will explore some works on the LSTI problem and expound on the difference to the LSTF problem.

Long sequence time-series input problem. In practical engineering, the researcher could truncate the inputs, summarize the sequences, or sample each batch to handle a long input sequence on simple models. However, when making precise predictions, valuable long-term dependency may be lost in this way. Instead of modifying inputs, Truncated BPTT [31] only uses last time steps to estimate the gradients in weight updates, and Auxiliary Losses [32] enhance the gradients flow by adding auxiliary gradients. Other attempts include Recurrent Highway Networks [33] and Bootstrapping Regularizer [34]. Dual-path RNN [35] is proposed to model extremely long sequences by organizing RNN blocks in the layer-wise stacking. These methods try to improve the gradient flows inside the recurrent network architecture, but their performance gain is limited when the sequence length excessively grows. The CNN-based methods [36,37] use the convolutional filter to capture the long-term dependency, and their receptive fields grow exponentially with the stacking of layers, which helps reduce computing cost but hurts the sequence alignment. For the LSTI problem, the main task is to enhance the model's ability to receive the long sequence and extract the long-range dependency from the inputs. We noticed that LSTF needs to establish the long-range dependency between outputs and inputs, namely the prediction capacity requires the model to predict a long sequence from inputs, and the LSTI techniques may help but are not feasible for it directly.

Attention model. Attention in the deep learning field is first proposed to address the computation problem of large images by selecting and only processing a sequence of regions. The attention model could build a correlation between different components and utilize the distinct data, which may benefit the development of the LSTF model. Then Bahdanau et al. [38] proposed the addictive attention to improve the word alignment of the encoder-decoder architecture in the translation task, which is the first work to use attention mechanism in sequential data. Then, its variant [39] has proposed the widely used location, general, and dot-product attention, and another variant [40] employed attention for speech recognition. Among all types of attention mechanisms, a special form is called self-attention, in which each element in the input sequence is used for calculating other elements' relevance in the same input sequence.

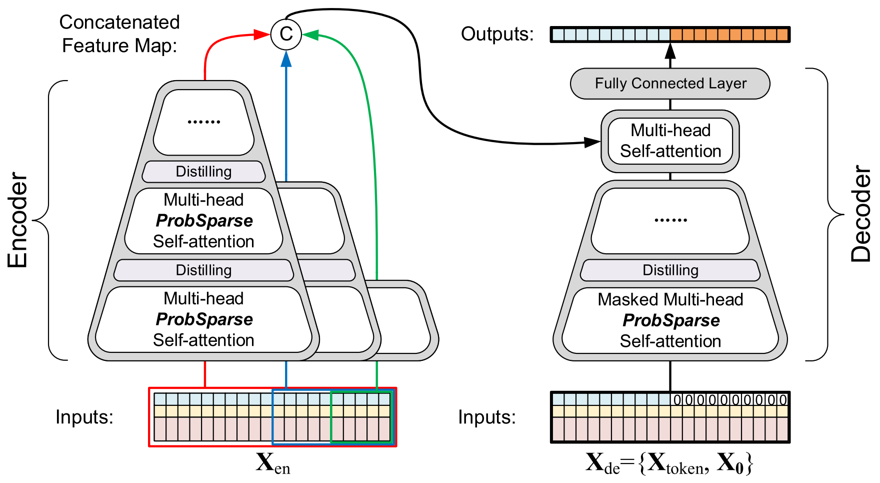

Concatenated Feature Map:
C
Outputs:
Fully Connected Layer
Multi-head Self-attention
Encoder
Decoder
......
Distilling
Multi-head *ProbSparse* Self-attention
Distilling
Multi-head *ProbSparse* Self-attention
......
Distilling
Masked Multi-head *ProbSparse* Self-attention
Inputs:
Inputs:
0 0 0 0 0 0 0 0 0 0 0 0
$\mathbf{X}_{\text{en}}$
$\mathbf{X}_{\text{de}}=\{\mathbf{X}_{\text{token}}, \mathbf{X}_{\mathbf{0}}\}$

Fig. 2. Overview of the Informer model. The left part is the Encoder, and it receives massive long sequence inputs (the first projection + the second/third embeddings). We replace the canonical self-attention with the proposed *ProbSparse* self-attention. The middle trapezoid is the self-attention distilling operation designed to sharply reduce the network stacking size by extracting the principal information. We draw three replicas of layer stacking in red, blue, and green tensor paths, and they are having-cascading selected to make the output's size aligned. The concatenated feature map builds a more robust representation that tends to generalize. For the right part, the Decoder receives long sequence inputs, pads the target elements into zero, measures the weighted attention composition of the feature map, and instantly predicts output elements (orange series) in a generative style. For the cross attention, we still maintain the vanilla one. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

The prevalent self-attention model, i.e., Transformer [14], has recently been proposed as new thinking of sequence modeling and has achieved great success, especially in the NLP field [41–44]. Moreover, the power of the Transformer's better sequence alignment ability has been validated by applying it to other fields such as speech [45], music [46], and image [47]. Furthermore, some works [48,49,16] have borrowed the architecture of Transformer and applied it to time-series forecasting problems for its ability of powerful sequence alignment.

Transformer-based model. The most related works [48,49,16] all start from a trail on applying Transformer in time-series data and fail in the LSTF problem as they use the vanilla Transformer. And some works [15,16] noticed the sparsity in the self-attention mechanism and we have discussed them in the main context. In our work, the Informer takes advantage of the Transformer's sequence alignment ability and makes a delicate design to streamline the calculation of self-attention which makes it amenable to the LSTF problem.

3. Preliminary

3.1. Sequence to sequence forecasting

Recurrent neural networks (RNN) based Encoder-Decoder architecture [50,30] has been established as a general sequence to sequence framework in prediction tasks. In the rolling forecasting setting with a fixed size window, a t-th sequence input is a series of vectors $\mathcal{X}^t = \{\mathbf{x}_1^t, \ldots, \mathbf{x}_{L_x}^t \mid \mathbf{x}_i^t \in \mathbb{R}^{d_x}\}$ of L_x elements and its target is the corresponding vectors $\mathcal{Y}^t = \{\mathbf{y}_1^t, \ldots, \mathbf{y}_{L_y}^t \mid \mathbf{y}_i^t \in \mathbb{R}^{d_y}\}$.

Long Sequence Time-series Forecasting (LSTF). The LSTF problem aims to predict the target sequence $\mathcal{Y}^t$ from the input sequence $\mathcal{X}^t$. The LSTF problem encourages a longer output's length L_y than previous works [50,30] and the feature dimension is not limited to univariate case ($d_y \geq 1$).

Encoder-Decoder Architecture. In the literature, most models are devised to "encode" the input representations $\mathcal{X}^t$ into hidden state representations $\mathcal{H}^t = \{\mathbf{h}_1^t, \ldots, \mathbf{h}_{L_h}^t\}$ and "decode" the output representations $\hat{\mathcal{Y}}^t$ from $\mathcal{H}^t$. The inference usually involves a step-by-step process named "dynamic decoding", where the decoder computes a new hidden state $\mathbf{h}_{k+1}^t$ from the previous state $\mathbf{h}_k^t$ and other necessary outputs from the k-th step then predicts the $(k+1)$-th sequence $\hat{\mathbf{y}}_{k+1}^t$. Based on this scheme, there have been a number of related attempts to address the sequence to sequence forecasting modeling, and their main difference lies in the structure of recurrent networks and the input/output strategy. Our proposed Informer generally follows the encoder-decoder mechanism and the overall architecture is given in Fig. 2.

Input Representation. The RNN models [51,7,52,30,11,53] capture the time-series pattern by the recurrent structure itself and barely relies on time stamps. The vanilla transformer [14,41] uses the point-wise self-attention mechanism, and the time stamps contain local positional contexts. However, in the LSTF problem, the ability to capture long-range independence requires global information like hierarchical time stamps (week, month and year) and agnostic time stamps [54] (holidays, events). This information is hardly leveraged in canonical self-attention and consequent query-key mismatches between the encoder and decoder bring underlying degradation on the forecasting performance.

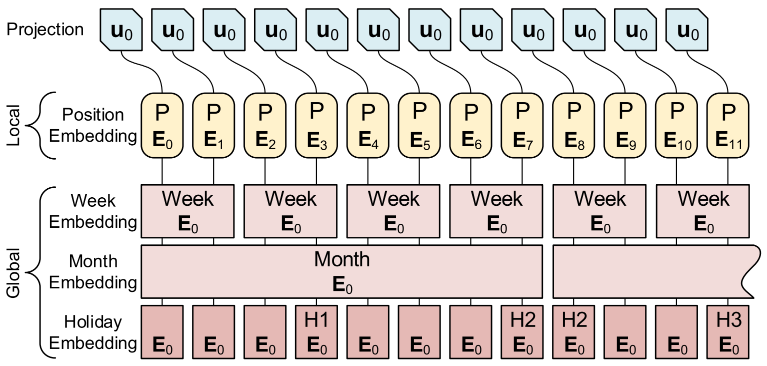


Fig. 3. The input representation of Informer. The inputs' embedding consists of three separate parts, a scalar projection, the local time stamp (Position) and global time stamp embeddings (Minutes, Hours, Week, Month, Holiday etc.)

In this work, a uniform input representation is given to enhance the global positional context and local temporal context of the time-series inputs. Fig. 3 gives an intuitive overview. Assuming we have the t-th sequence input $\mathcal{X}^t$ and p types of global time stamps and the feature dimension after input representation is d_{model}. We firstly preserve the local context by a fixed position embedding:

$$\begin{aligned}\text{PE}_{(pos,2j)} &= \sin(pos/(2L_x)^{2j/d_{\text{model}}}) \\ \text{PE}_{(pos,2j+1)} &= \cos(pos/(2L_x)^{2j/d_{\text{model}}})\end{aligned} \quad , \tag{1}$$

where $j \in \{1,\ldots,\lfloor d_{\text{model}}/2\rfloor\}$. Each global time stamp is employed by a learnable stamp embeddings $\text{SE}_{(pos)}$ with a limited vocab size (up to 60, namely taking minutes as the finest granularity). That is, the self-attention's similarity computation can have access to the global context, and the computation consumption is affordable for long inputs. To align the dimension, we project the scalar context $\mathbf{x}_i^t$ into a d_{model}-dim vector $\mathbf{u}_i^t$ with 1-D convolutional filters (kernel width=3, stride=1). Thus, we have the feeding vector

$$\mathcal{X}^t_{\text{feed}[i]} = \alpha \mathbf{u}_i^t + \text{PE}_{(L_x\times(t-1)+i,\ )} + \sum_p [\text{SE}_{(L_x\times(t-1)+i)}]_p \quad , \tag{2}$$

where $i \in \{1,\ldots,L_x\}$, and α is the factor balancing the magnitude between the scalar projection and local/global embeddings. We recommend $\alpha = 1$ if the sequence input has been normalized.

4. Methodology

Existing methods for time-series forecasting can be roughly grouped into two categories. Classical time-series models serve as a reliable workhorse for time-series forecasting [24–27], and deep learning techniques mainly develop an encoder-decoder prediction paradigm by using RNN and their variants [7–9]. Our proposed Informer holds the encoder-decoder architecture while targeting the LSTF problem. Please refer to Fig. 2 for an overview and the following sections for details.

4.1. *Efficient self-attention mechanism*

The canonical self-attention [14] is defined based on the tuple inputs, i.e., query, key and value, which performs the scaled dot-product as:

$$\mathcal{A}_{cano}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V} \quad , \tag{3}$$

where $\mathbf{Q} \in \mathbb{R}^{L_Q\times d}$, $\mathbf{K} \in \mathbb{R}^{L_K\times d}$, $\mathbf{V} \in \mathbb{R}^{L_V\times d}$ and d is the input dimension. To further discuss the self-attention mechanism, let $\mathbf{q}_i$, $\mathbf{k}_i$, $\mathbf{v}_i$ stand for the i-th row in $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ respectively. Following the similar formulation [55], the i-th query's attention is defined as a kernel smoother in a probability form:

$$\mathcal{A}_{cano}(\mathbf{q}_i,\mathbf{K},\mathbf{V}) = \sum_j \frac{k(\mathbf{q}_i,\mathbf{k}_j)}{\sum_l k(\mathbf{q}_i,\mathbf{k}_l)}\mathbf{v}_j = \mathbb{E}_{p(\mathbf{k}_j|\mathbf{q}_i)}[\mathbf{v}_j] \, , \tag{4}$$

where $p(\mathbf{k}_j|\mathbf{q}_i) = k(\mathbf{q}_i,\mathbf{k}_j)/\sum_l k(\mathbf{q}_i,\mathbf{k}_l)$ and $k(\mathbf{q}_i,\mathbf{k}_j)$ selects the asymmetric exponential kernel $\exp(\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d})$. The self-attention combines the values as outputs and acquires better representations based on computing the probability $p(\mathbf{k}_j|\mathbf{q}_i)$. If the query $\mathbf{Q}$ and the key $\mathbf{K}$ are the same, the probability becomes trivial and the output is the mean of $\mathbf{V}$. Otherwise, if the

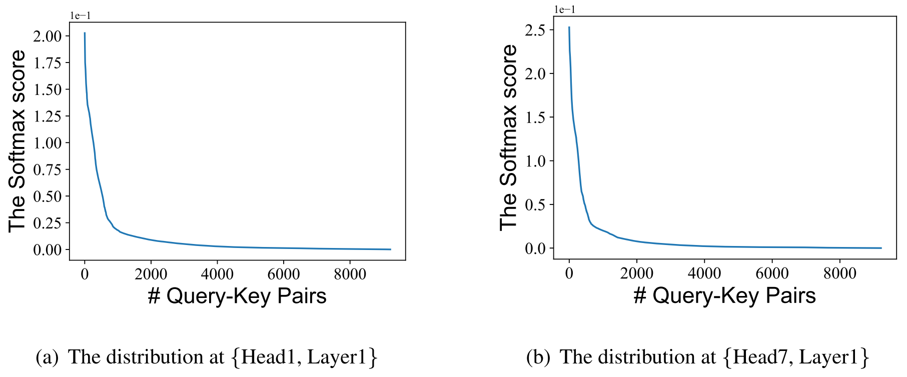


Fig. 4. The self-attention scores. The Softmax probabilities were collected from a 4-layer canonical Transformer trained on the ETTh_1 dataset. We draw the first and the last heads in the first layer and sort the scores by magnitude. Each point in the x-axis stands for an individual Query-Key pair.

query-key pairs are totally different, with similar dot-product magnitude, the output also has a high chance of being close to the mean of $\mathbf{V}$. The mixture of similarity and dissimilarity in the probability establishes the self-attention mechanism. However, the probability computation requires the quadratic times dot-product computation and $\mathcal{O}(L_Q L_K)$ memory usage. It becomes the major drawback of expanding prediction capacity when applying Transformer models in LSTF problems.

Some previous attempts have revealed that the distribution of self-attention probability has potential sparsity and they have designed "selective" counting strategies on all $p(\mathbf{k}_j|\mathbf{q}_i)$ without significantly affecting the performance. The Sparse Transformer [15] incorporates both the row outputs and column inputs, in which the sparsity arises from the separated spatial correlation. The LogSparse Transformer [16] notices the cyclical pattern in self-attention and forces each cell to attend to its previous one by an exponential step size. The Longformer [17] extends the above two works to more complicated sparse configuration. However, they are limited to the theoretical analysis from heuristic methods and tackle each multi-head self-attention with the same strategy, which narrows their further improvements.

To motivate our approach, we first perform a qualitative assessment on the learned attention patterns of the canonical self-attention. In Fig. 4, the first thousand pairs contribute the most attention scores but only occupy about 12% of all pairs and this makes a header field. The self-attention score "sparsity" forms a long tail distribution, i.e., a few dot-product pairs contribute to the major attention and others generate trivial attention. Then, the next question is how to distinguish them?

4.1.1. Query sparsity measurement

As we have discussed in Eq. (4), the i-th query's attention on all the keys forms a probability $p(\mathbf{k}_j|\mathbf{q}_i)$ and the attention's output is its composition with values $\mathbf{v}$. From the self-attention score's sparsity illustrated in Fig. 4, most i-th query's attention will not have the dot-product pairs that lie in the header field. The corresponding probability $p(\mathbf{k}_j|\mathbf{q}_i)$ is close to a uniform distribution $q(\mathbf{k}_j|\mathbf{q}_i) = 1/L_K$ and the self-attention mechanism becomes a trivial sum of values $\mathbf{V}$ and is redundant to the residential input. Otherwise, one dominant dot-product pair emerges as a "step signal" in the uniform distribution q. The more alike pairs encourage the selected query's attention probability distribution p away from the uniform distribution q. In this way, the "likeness" between the distributions p and q can be used to identify the "important" queries. Without loss of generality, we measure the "likeness" through Kullback-Leibler (KL) divergence:

$$\begin{aligned} KL(q||p) &= \sum_{j=1}^{L_K} \frac{1}{L_K} \ln \frac{1/L_K}{k(\mathbf{q}_i, \mathbf{k}_j)/\sum_l k(\mathbf{q}_i, \mathbf{k}_l)} \\ &= \ln \sum_{l=1}^{L_K} e^{\frac{\mathbf{q}_i \mathbf{k}_l^\top}{\sqrt{d}}} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} - \ln L_K \end{aligned} . \tag{5}$$

Dropping the constant $\ln L_K$, we define the i-th query's sparsity measurement as:

$$M(\mathbf{q}_i, \mathbf{K}) = \ln \sum_{j=1}^{L_K} e^{\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}}} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} , \tag{6}$$

where the first term is the Log-Sum-Exp (LSE) of $\mathbf{q}_i$ on all the keys and the second term is the arithmetic mean on them. If the i-th query gains a larger $M(\mathbf{q}_i, \mathbf{K})$, its attention probability p is more "diverse" and has a high chance to contain the dominate dot-product pairs in the header field of the long tail self-attention distribution. Note that we use the KL-based measurement as a good choice for relative entropy, and its non-negative characteristic [56] is the friendly type for floating-point calculation. Other entropy-derived formulations could be explored in future work.

Furthermore, we could build an empirical estimator $\widehat{KL}(q||p)$ to $KL(q||p)$ for efficiently applying measurements. The commonly used method is k-nearest neighbor (kNN) method-based estimator [57]. Given a set of samples $\hat{X} = \{X_1, X_2, \dots, X_N\}$ drawn i.i.d. from uniform distribution q, and another set $\hat{Y} = \{Y_1, Y_2, \dots, Y_M\}$ drawn i.i.d. from self-attention distribution p. Without loss of generality, we assume the samples in these sets are sorted in increasing order [58] and $M \le N$. Thus, $X_i = (iL_k)/N$ can be seen as N scales of N-bins histogram. Following the previous work [59], we derived a kNN-based estimator:

$$\widehat{KL}(q||p) = \frac{1}{N}\sum_{i=1}^{N} \ln \frac{\nu_i}{\epsilon_i} + \ln \frac{M}{N-1} \quad , \tag{7}$$

where ϵ_i is the distance between X_i and its k-th nearest neighbor in $\hat{X}\backslash X_i$, which makes $\epsilon_i = (lL_k)/N$ a constant. The ν_i represents the distance between X_i and its k-th nearest neighbor in $\hat{Y}$. We will show that random sampling contributes the best kNN estimator bias. And we also have the sibling sparsity measurement as:

$$\widehat{M}(\mathbf{q}_i, \mathbf{K}) = \frac{1}{N}\sum_{i=1}^{N} \ln \frac{\nu_i}{\epsilon_i} \quad . \tag{8}$$

4.1.2. ProbSparse self-attention

Based on the proposed measurement, we have the *ProbSparse* self-attention by allowing each key to only attend to the u dominant queries:

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}(\frac{\overline{\mathbf{Q}}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V} \quad , \tag{9}$$

where $\overline{\mathbf{Q}}$ is a sparse matrix of the same size of $\mathbf{q}$ and it only contains the Top-u queries under the sparsity measurement $M(\mathbf{q}, \mathbf{K})$. We can set the query counter as $u = \ln L_Q$ to be consistent with our objective complexity. It makes the *ProbSparse* self-attention only need to calculate $\mathcal{O}(\ln L_Q)$ dot-product for each query-key lookup, and the layer memory usage maintains $\mathcal{O}(L_K \ln L_Q)$. However, the $\ln L_Q$ may be too small for the LSTF problem's "sparsity" self-attention assumption in Fig. 4. Take the input size equals 720 as an example, and there will be only about 0.9% dot-product pairs in the header field. Thus we add a constant sampling factor c into the query counter and let $u = c \cdot \ln L_Q$, which makes the sampling factor a hyperparameter under LSTF problem settings.

Moreover, the *ProbSparse* Self-attention can generate different "important" queries with different sparsity patterns for each head. Specifically, we use a n-heads self-attention at identical layers, namely the Multi-head Attention:

$$\begin{aligned} &\mathcal{MA}(\mathbf{X}) = \mathrm{Concat}(\mathrm{head}_1, \dots, \mathrm{head}_n) \\ &\text{where } \mathrm{head}_k = \mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \end{aligned} \quad , \tag{10}$$

where $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^K$, $\mathbf{V} = \mathbf{X}\mathbf{W}^V$, and $\{\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V\} \in \mathbb{R}^{d_{\text{model}} \times d}$ are different subspaces' projector. The Concat$(\cdot)$ stands for the concatenate operation. We could derive that the *ProbSparse* attention varies with different projectors. This flexibility is the essential difference from the previous heuristic methods [16,17], which avoids severe information loss in fixed sparsity patterns. An appropriate head configuration could help build a more stable multi-head *ProbSparse* Self-attention.

4.1.3. Max-mean measurement

However, the traverse of all the queries for the measurement $M(\mathbf{q}_i, \mathbf{K})$ requires calculating each dot-product pairs, i.e., quadratically $\mathcal{O}(L_Q L_K)$, besides the LSE operation has the potential numerical stability issue. Motivated by this, we propose an empirical approximation for the efficient acquisition of the query sparsity measurement. We will first present the theoretical analysis and then develop the approximation.

Lemma 1. *For each query $\mathbf{q}_i \in \mathbb{R}^d$ and $\mathbf{k}_j \in \mathbb{R}^d$ in the keys set $\mathbf{K}$, we have the bound as $\ln L_K \le M(\mathbf{q}_i, \mathbf{K}) \le \max_j\{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}\} - \frac{1}{L_K}\sum_{j=1}^{L_K}\{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}\} + \ln L_K$. When $\mathbf{q}_i \in \mathbf{K}$, it also holds.*

Proof. For the individual $\mathbf{q}_i$, we can relax the discrete keys into the continuous d-dimensional variable, i.e. vector $\mathbf{k}_j$.

Firstly, we look into the minimal value. For each query $\mathbf{q}_i$, the first term of $M(\mathbf{q}_i, \mathbf{K})$ becomes the log-sum-exp of the inner-product of a fixed query $\mathbf{q}_i$ and all the keys, and we can define $f_i(\mathbf{K}) = \ln \sum_{j=1}^{L_K} e^{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}}$. From the Eq. (2) in the Log-sum-exp network [60] and its further analysis, the function $f_i(\mathbf{K})$ is convex. Moreover, the adding operation of $f_i(\mathbf{K})$ and a linear combination of $\mathbf{k}_j$ makes $M(\mathbf{q}_i, \mathbf{K})$ to be a convex function for the fixed query. Then we can take the derivation of the measurement with respect to the individual vector $\mathbf{k}_j$ as:

$$\frac{\partial M(\mathbf{q}_i,\mathbf{K})}{\partial \mathbf{k}_j}=\frac{e^{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}}}{\sum_{j=1}^{L_K}e^{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}}}\cdot\frac{\mathbf{q}_i}{\sqrt{d}}-\frac{1}{L_K}\cdot\frac{\mathbf{q}_i}{\sqrt{d}}\quad .$$

To reach the minimum value, we let $\vec{\nabla}M(\mathbf{q}_i)=\vec{0}$ and the following condition is acquired as $\mathbf{q}_i\mathbf{k}_1^\top+\ln L_K=\cdots=\mathbf{q}_i\mathbf{k}_j^\top+\ln L_K=\cdots=\ln\sum_{j=1}^{L_K}e^{\mathbf{q}_i\mathbf{k}_j^\top}$. Naturally, it requires $\mathbf{k}_1=\mathbf{k}_2=\cdots=\mathbf{k}_{L_K}$, and we have the measurement's minimum as $\ln L_K$:

$$M(\mathbf{q}_i,\mathbf{K})\geq\ln L_K\quad . \tag{11}$$

Secondly, we look into the upper bound. If we select the largest inner-product $\max_j\{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}\}$, it is easy that

$$\begin{aligned}M(\mathbf{q}_i,\mathbf{K})&=\ln\sum_{j=1}^{L_K}e^{\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}}-\frac{1}{L_K}\sum_{j=1}^{L_K}(\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}})\\&\leq\ln(L_K\cdot\max_j\{\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}\})-\frac{1}{L_K}\sum_{j=1}^{L_K}(\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}})\quad .\\&=\ln L_K+\max_j\{\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}\}-\frac{1}{L_K}\sum_{j=1}^{L_K}(\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}})\end{aligned} \tag{12}$$

Combine the Eq. (11) and Eq. (12), we have the final results. When the key set is the same with the query set, the above discussion also holds. □

Immediately, we adopt the upper bound and define the max-mean measurement as:

$$\begin{aligned}\overline{M}(\mathbf{q}_i,\mathbf{K})&=\max_j\{\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}\}-\frac{1}{L_K}\sum_{j=1}^{L_K}\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}\quad .\\&\overset{\text{def}}{=}\max_j\{\mathbf{q}_i\mathbf{k}_j^\top\}-\operatorname*{mean}_j\{\mathbf{q}_i\mathbf{k}_j^\top\}\end{aligned} \tag{13}$$

The max-operator in $\overline{M}(\mathbf{q}_i,\mathbf{K})$ is less sensitive to zero values and is numerical stable. Meanwhile, if we assumed that the k-th nearest neighbor has $k<L_K/2$ in Eq. (7), we have $\widehat{M}(\mathbf{q}_i,\mathbf{K})\leq\overline{M}(\mathbf{q}_i,\mathbf{K})$ by substituting individual elements.

The range of Top-u approximately holds in the boundary relaxation with Lemma 2 in Section 4.4. The Eq. (13) is similar to approximating skewness but it performs on the dot-product pairs. If the average of the i-th query-keys is away from the maximum, we deduce that at least one dominant query-key pair but all that the i-th query-keys pairs form a small long tail distribution in a query-wise manner. However, we do not need to go through all the query-key pairs in the quadratic calculation $\mathcal{O}(L_K L_Q)$. Under the query-wise long tail distribution, we use the random sampling strategy to acquire $\ln L_K$ keys for each individual query. We compute $L_Q\ln L_K$ pairs in total. Then we select Top-u queries by applying the max-mean measurement on the cropped matrix. Note that our goal is to find the query-key pairs in the header field, but the max-mean measurement favors most skewness distribution far from the uniform one, and the sampling strategy alleviates that by forcing the measurement focus not such a heavy-tailed distribution. If one query has more dominant pairs than others, it has a higher probability to rank as the Top-u queries. In the previous description, we omit the details of random sampling strategy to keep the consistency and we will discuss them separately in the following section.

4.1.4. Random sampling

The time-series are individual point inputs, and then the self-attention mechanism forms i.i.d. discrete distribution. Recall the Kullback-Leibler divergence in Eq. (5), the $KL(q||p)$ measures the closeness of the unknown attention distribution p to the uniform distribution q. The sampling strategy aims to reduce the KL computation complexity from $\mathcal{O}(L_K L_Q)$ to $L_Q\ln L_K$ when selecting the dominating queries. The whole process could be considered as using the empirical estimator $\widehat{KL}(q||p_i)$ for original $KL(q||p)$ and then applying the max-mean measurement in Eq. (13).

Our discussion will be organized into two parts. First, for arbitrary $p_i(x)\in P$, we investigate $\widehat{KL}(q||p_i)$ to isolate the bias. Limiting sample points to $N=\ln(L_k)$, we can draw a brief illustration. The $p_i(x)$'s uniform sampling in increasing order can minimize the bias $|\mathbb{E}[\widehat{KL}(q||p)]-KL(q||p)|$ for most conditions. Second, we empirically evaluated with numerical experiments with long-tail distributions.

Applying **Assumption 2** ($d=1$) [61], the estimation bias for $\widehat{KL}(q||p)$ becomes:

$$|\mathbb{E}[\widehat{KL}(q\|p)]-KL(q\|p)|=\mathcal{O}\left((\min\{M,N\})^{-\frac{2}{3}}\ln\min\{M,N\}\right)\quad , \tag{14}$$

under the following conditions:

(a) if $q(x) > 0$, then $p(x) > 0$;
(b) $P(q(\hat{X}) \le t) \le \mu t^{\gamma}$ and $P(p(\hat{X}) \le t) \le \mu t^{\gamma}$ with constants $\mu, \gamma \in (0, 1]$;
(c) $||\nabla^2 q|| \le C_0$ and $||\nabla^2 p|| \le C_0$, bounded for some constant C_0;
(d) $\mathbb{E}\left[\|\hat{X}\|^s\right] \le K$, and $\mathbb{E}\left[\|\hat{Y}\|^s\right] \le K$ with constants $s > 0$, $K > 0$.

Condition (a) ensures the defined KL divergence is valid. Condition (b) is the tail assumption, in which a lower γ indicates a stronger tail, and the convergence for the bias of KL divergence will be slower. For example, $\gamma = 1$ for Gaussian distribution, $\gamma = 0.5$ for Cauchy distribution, and if q, p hold different γ_q, γ_p respectively, $\gamma = min\{\gamma_q, \gamma_p\}$ [62]. Condition (c) forms the smoothness assumption, and condition (d) constrains $\max\{\nu_i\}$ and $\max\{\epsilon_i\}$.

We provide the following corollary to illustrate that, if we draw M samples $\hat{Y}$ (**Y** in matrix) i.i.d., however not in the full-length interval $(0, L_k]$, but in a subset $[L_1, L_2]$, the bias for the empirical estimation Eq. (7) will increase respectively. It indicates that random sampling, equivalent to a uniform sample in the full interval in increasing order, can minimize the estimation bias at a high probability.

Corollary 1. *Given an uniform distribution $q(x)$, cdf is $Q(x)$, and an arbitrary distribution $p(x)$, cdf is $P(x)$, if we sample N i.i.d. points $\hat{X}$ in interval $(0, L_k]$ from $q(x)$ and M i.i.d. points $\hat{Y}$ in interval $[L_1, L_2]$ from $p(x)$, where $[L_1, L_2] \subset (0, L_k]$. Then the estimation bias Eq. (14) increases in proportional to $(L_k)/(L_2 - L_1)$.*

Proof. The parameters d, γ, M, N hold when not changing q and p distribution. However, if we constraint the sample interval for $\hat{X}, \hat{Y}$ from $(0, L_k]$ to its subset $[L_1, L_2]$, e.g. $(P^{-1}(0.5), L_k]$ referring to the latter-half interval i.i.d. sample strategy (like the Max strategy), the reduced distribution of p will be defined as:

$$p'(x) = \begin{cases} 0 & P(x) < P^{-1}(L_1) \text{ or } P(x) > P^{-1}(L_2) \\ \dfrac{p(x)}{P^{-1}(L_2) - P^{-1}(L_1)} & P^{-1}(L_1) \le P(x) \le P^{-1}(L_2) \end{cases}, \tag{15}$$

where μ should be updated to $\mu' = \mu/(P^{-1}(L_2) - P^{-1}(L_1))$ considering the condition (b), and the other parameters hold.

Following Eq. (41) and Eq. (42) in [61], $\hat{X}, \hat{Y}$ will be divided into two support set S_1, S_2, where $S_p \subset (0, L_k]$, $S_q \subset [L_1, L_2]$ is the support set for distributions p and q respectively. Let $V(S_p) = N$, $V(S_q) = M$, we have:

$$S_1 = \left\{ \mathbf{y} \mid p(\mathbf{y}) > \frac{2C_1}{c_d} a_M^2 \right\}, \qquad S_2 = S_q \backslash S_1. \tag{16}$$

Note that the term $\frac{2C_1}{c_d} a_M^2$ depending on parameters d, γ, M, N. According to Eq. (26) [61], I_1, I_3 are constant form when q is a uniform distribution, and we only consider estimating the convergence speed of I_2 into two intervals S_1, S_2:

$$\begin{aligned} |I_2| &= \left| \mathbb{E}\left[\ln \frac{P_p(B(\mathbf{Y}, \nu))}{c_1 \nu p(\mathbf{Y})} \right] \right| \\ &\le \left| \mathbb{E}\left[\ln \frac{P_p(B(\mathbf{Y}, \nu))}{c_1 \nu p(\mathbf{Y})} \mathbf{1}(\mathbf{Y} \in S_1) \right] \right| + \left| \mathbb{E}\left[\ln \frac{P_p(B(\mathbf{Y}, \nu))}{c_1 \nu p(\mathbf{Y})} \mathbf{1}(\mathbf{Y} \in S_2) \right] \right| \end{aligned}. \tag{17}$$

For S_1, we apply the Eq. (47) and Eq. (53) respectively in [61]:

$$\begin{aligned} &\left| \mathbb{E}\left[\ln \frac{P_p(B(\mathbf{Y}, \nu))}{c_1 \nu p(\mathbf{Y})} \mathbf{1}(\mathbf{Y} \in S_1) \right] \right| \propto V(S_1) \\ &\left| \mathbb{E}\left[\ln \frac{P_p(B(\mathbf{Y}, \nu))}{c_1 \nu p(\mathbf{Y})} \mathbf{1}(\mathbf{Y} \in S_2) \right] \right| \propto \mu \end{aligned}. \tag{18}$$

According to the division for S_q in Eq. (16), $V(S_1)$ increases as $p(x)$ is replaced with $p'(x)$, and $p'(x) = p(x)/(P^{-1}(L_2) - P^{-1}(L_1)) > p(x)$, $x \in [L_1, L_2]$, which naturally leads to more elements $\mathbf{x}$ fall into S_1, namely $V(S_1)$ increasing. It also applies to $\mu' = \mu/(P^{-1}(L_2) - P^{-1}(L_1)) > \mu$. Considering the Eq. (17), the estimation bias increases in proportion to $(L_k)/(L_2 - L_1)$. □

The Corollary 1 indicates that the best strategy to minimize the KL divergence estimation bias is random sampling, which is equivalent to uniform sampling $\lfloor \ln(L_k) \rfloor$ nodes from $\hat{Y}$ in increasing order. Other i.i.d. sampling strategies can be briefly formulated as sampling $M = \lfloor \ln(L_k) \rfloor$ nodes i.i.d. in l intervals $(0, L_1] \cup (L_2, L_3] \cup \ldots (L_{l-1}, L_l] \subset (0, L_k]$, and the bias also increases by using Eq. (17) for l times. Moreover, if p obeys a long-tail distribution, e.g. lognormal distribution, if we apply Max sample strategy in interval $(P^{-1}(0.5), L_k]$, since $P^{-1}(0.5)$ could be rather small (e.g., $P^{-1}(0.5) = \mu(LogNormal(\mu = 0.1,\ \sigma^2 = 1))) = e^{0.6}$), $p'(x), \mu$ in Eq. (15) increases little (3.7%), due to the nature of long-tail distribution that the tail-part sample points are approximately uniform distributed in $[P^{-1}(0.5), L_k]$. Thus, the Max sample strategy has a similar estimation bias to the random sample strategy in long-tail distribution conditions.

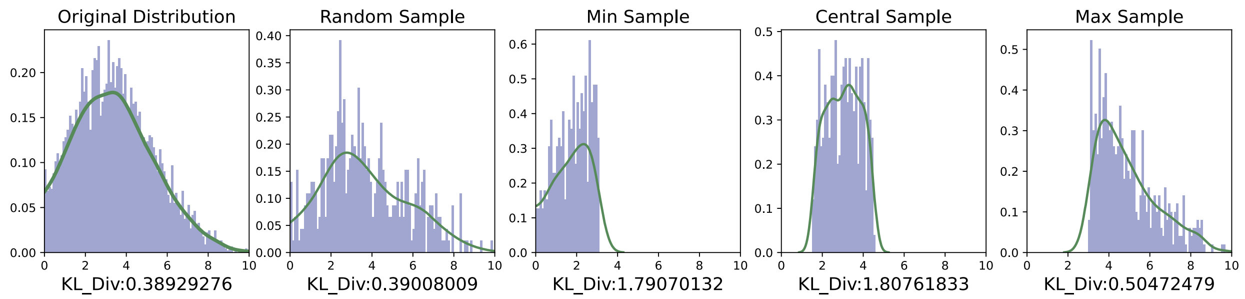


Fig. 5. The histogram and KL divergence with different sample strategies (Gaussian distribution).

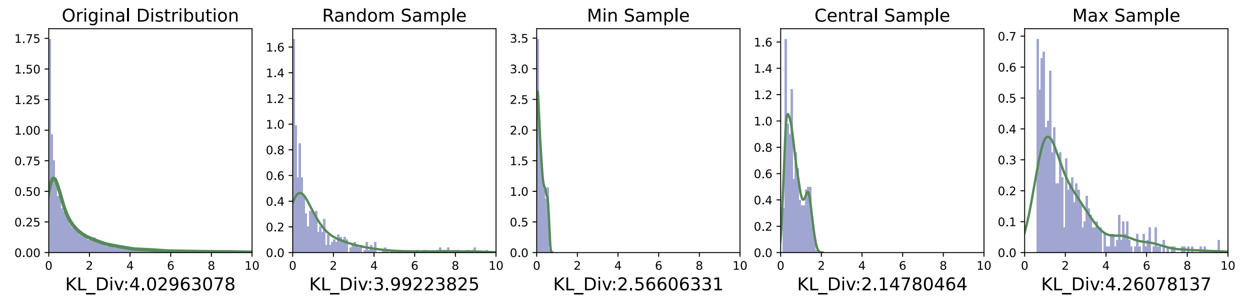


Fig. 6. The histogram and KL divergence with different sample strategies (Weibull distribution).

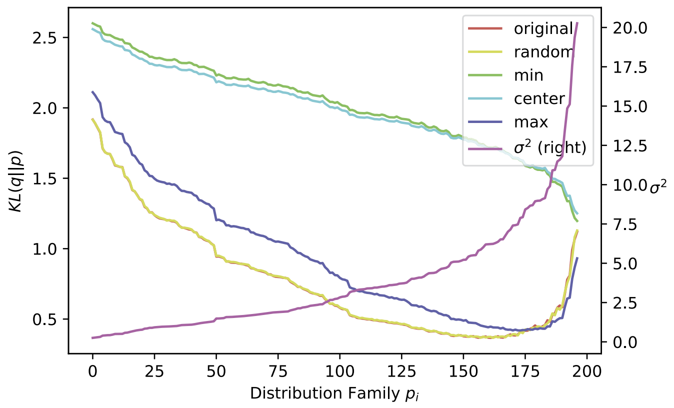


Fig. 7. The case study for the Gaussian distribution with a long-tailed variance.

To empirically estimate the KL divergence, we only select ten samples from the distribution p, and the bin number is set to 100. Fig. 5 stands for a Gaussian distribution $\mathcal{N}(\mu = 3, \sigma^2 = 5)$, the random sampling strategy shows superior performance in reflecting the original distribution. Fig. 6 refers to a Weibull distribution in long-tail condition, in which over 30% of the sample points are in $(0, 0.5]$, only random sampling can minimize the divergence estimation bias, while the max sampling shows potential. The experiment section will include a complete evaluation with real-world datasets.

We focus on the distribution family $P = \{p_1, p_2, \cdots, p_m\}$, the p can be approximated as a Gaussian Distribution bounded in $(0, L_k]$ in real-world applications. As we discussed above, due to the long-tail distribution of the attention feature map, $Var(p_i(x))$ has drastic numerical fluctuation, which also affects the estimation of KL divergence. We provide a case study for this factor. In Fig. 7, a set of Gaussian distributions family $P = \{p_1, p_2, \cdots, p_{200}\}$, $p_i \sim \mathcal{N}(3, \sigma_i^2)$, and the variance $\{\sigma_1^2, \sigma_2^2, \cdots, \sigma_{200}^2\} \sim LogNormal(1, 1)$ in increasing order. The lognormal distribution is a typical long-tail distribution, and in this case, over 90% of variance $(\{\sigma_1^2, \sigma_2^2, \cdots, \sigma_{180}^2\})$ is below 10, and most of the sample strategies have little bias with the actual KL divergence. However, in the extreme condition, which refers to $\{\sigma_{181}^2, \sigma_{182}^2, \cdots, \sigma_{200}^2\} > 10$, sample points from the minimum-half or the central-half curves in $p(x)$ lose much information from $p(x)$. The difference is the KL divergence between a Gaussian distribution and a truncated Gaussian distribution [63]. However, random sampling and maximum-half sampling still minimize the KL divergence estimation.

Algorithm 1: The pseudo-code for *ProbSparse* Self-attention.

```
Function forward($\mathbf{Q}, \mathbf{K}, \mathbf{V}, c$):
    /* This is the single-head forward procedure.                    */
    Input: tensor $\mathbf{Q} \in \mathbb{R}^{L_Q \times d}$, $\mathbf{K} \in \mathbb{R}^{L_K \times d}$, $\mathbf{V} \in \mathbb{R}^{L_V \times d}$; sampling factor $c$
    Output: the self-attention feature map $\mathbf{S}$
1   set $u = c \ln L_Q$ and $U = L_Q \ln L_K$                  // The hyperparameters.
2   randomly select $U$ dot-product pairs as the cropping operator $[\cdot]_U$
3   set the sample matrix $\bar{\mathbf{S}} = [\mathbf{Q}\mathbf{K}^\top]_U$
4   compute the measurement $\overline{M} = \max(\bar{\mathbf{S}}) - \mathrm{mean}(\bar{\mathbf{S}})$ for each query
5   select Top-$u$ queries under $\overline{M}$ as $\bar{\mathbf{Q}}$
6   set $\mathbf{S}_h = \mathrm{Softmax}(\bar{\mathbf{Q}}\mathbf{K}^\top/\sqrt{d}) \cdot \mathbf{V}$      // The header field.
7   set $\mathbf{S}_t = \mathrm{mean}(\mathbf{V})$                        // The tail field.
8   build $\mathbf{S} = \{\mathbf{S}_h, \mathbf{S}_t\}$ by the original row arrangement accordingly
return
```

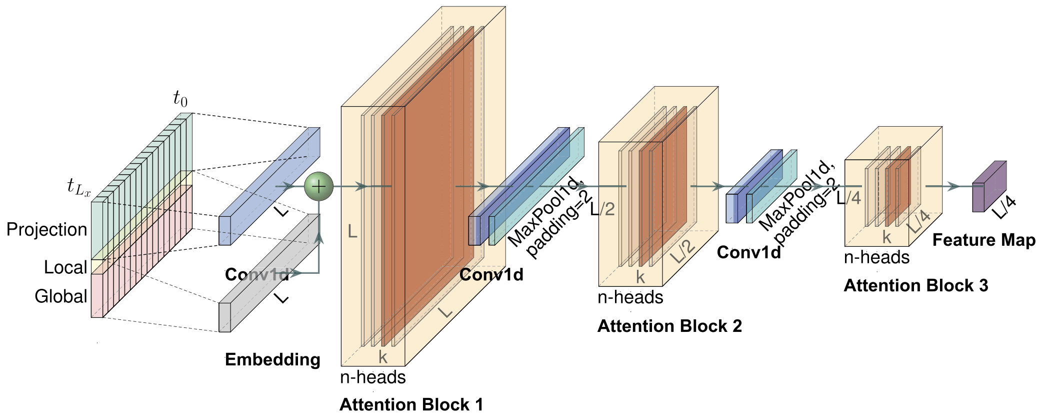


Fig. 8. The single stack in Informer's encoder. (1) The horizontal stack stands for an individual one of the layer stacking replicas in the encoder. (2) The presented one is the main stack receiving the whole input sequence. Then the second stack takes half slices of the input, and the subsequent stacks repeat. (3) The red layers are dot-product matrixes, and they get cascade decrease by applying self-attention distilling on each layer. (4) Concatenate all stacks' feature maps as the encoder's output.

4.1.5. *Implementation*

We have presented the pseudo-code in Algorithm (1). As discussed above, step 2 and step 3 keep the logarithmic complexity. Moreover, we compute the measurement $\overline{M}$ through rolling all the queries, which makes a linear complexity. The following steps could be highly efficient vector operations and maintain logarithmic total memory usage. In practice, the input length is typically equivalent during queries and keys' self-attention computation, i.e., $L_Q = L_K = L$, such that the total *ProbSparse* self-attention time complexity and space complexity are $\mathcal{O}(L \ln L)$. For the masked multi-head self-attention in the decoder, we can achieve this by applying the positional mask on step 6 and using cusum$(\cdot)$ to replace mean$(\cdot)$ of step 7.

4.2. *Encoder*

The encoder is designed to extract the robust long-range dependency of the long sequential inputs, which allows for processing longer sequential inputs under the memory usage limitation. After the input representation, the t-th sequence input $\mathcal{X}^t$ has been shaped into a matrix $\mathbf{X}^t_{\mathrm{en}} \in \mathbb{R}^{L_x \times d_{\mathrm{model}}}$. We clip it into different series on the time dimension and feed them into multiple layer stacking with self-attention distilling. Then we concatenate the final feature map from each truncated stacks as the hidden representation of sequential inputs. A sketch of the encoder's design is given in Fig. 8 for clarity.

4.2.1. *Self-attention distilling*

As the natural consequence of the *ProbSparse* self-attention mechanism, the encoder's feature map has redundant combinations of value $\mathbf{V}$. We use the distilling operation to privilege the superior ones with dominating features and make a focused self-attention feature map in the next layer. It trims the input's time dimension sharply, seeing the n-heads weights matrix (overlapping red squares) of Attention blocks in Fig. 8. Inspired by the dilated convolution [64,65], our "distilling" procedure forwards from j-th layer into $(j+1)$-th layer as:

$$\mathbf{X}^t_{j+1} = \mathrm{MaxPool}\left(\mathrm{ELU}(\, \mathrm{Conv1d}([\mathbf{X}^t_j]_{\mathrm{AB}})\,) \right) \quad , \tag{19}$$

where $[\cdot]_{\text{AB}}$ represents the attention block. It contains the multi-head *ProbSparse* self-attention and the essential operations, where Conv1d$(\cdot)$ performs an 1-D convolutional filters (kernel width=3) on time dimension with the ELU$(\cdot)$ activation function [66]. We add a max-pooling layer with stride 2 and down-sample $\mathbf{X}^t$ into its half slice after stacking a layer. The down-sampling operation reduces the whole memory usage to be $\mathcal{O}((2-\epsilon)L\log L)$, where ϵ is a small number.

The self-attention mechanism forces the information pair-wisely exchanging with each other in inputs and generates a new representation with projected inputs in Eq. (10). Meanwhile, the distilling operation privileges the superior ones with dominating features and makes a focused self-attention feature map in the next layer. As the Sparse Transformer [15] indicated, they had also noticed the importance of reducing redundant self-attention pairwise scores for better extraction of high-level concepts. Besides, our distilling operation trims the time dimension sharply, which can dramatically reduce the computation cost of Eq. (9) and the memory requirement of self-attention scores featuremap. It is illustrated as the n-heads weights matrix (overlapping red squares) of Attention blocks in Fig. 8.

4.2.2. Multiple layer stacking

To enhance the robustness of the distilling operation, we clip the input matrix $\mathbf{X}^t_{\text{en}}$ into sequential halving slices by the time dimension as $\{\ldots, \mathbf{X}^t_{(i)} \in \mathbb{R}^{L_{(i)}\times d_{\text{model}}}, \ldots\}$ and $L_{(i)} = L_x/2^{(i-1)}, i \in \{1, 2, 3, \ldots\}$. For each slice, we feed it into the individual layer stack and progressively decrease the number of self-attention distilling layers by dropping one layer at a time, like multiple pyramids in Fig. 8, such that their output dimension is aligned. Thus, we concatenate all the stacks' outputs and have the final hidden representation as

$$\mathbf{X}^t_{\text{en\_out}} = \text{Concat}(\mathbf{X}^t_{(0)}, \mathbf{X}^t_{(1)}, \ldots) \quad . \tag{20}$$

It can be considered a way to increase the resolution of multi-scale long-range dependencies and complementary in applying the distilling operation. Since we utilize the distilling operation on each layer and halve the time dimension, the total computational cost will not exceed that of a single feeding in the vanilla Transformer.

4.3. Decoder

We use a standard decoder structure [14] in Fig. 2 and it is composed of a stack of two identical multi-head attention layers. The generative inference is employed to alleviate the speed plunge in long prediction. We feed the decoder with the following vectors

$$\mathbf{X}^t_{\text{de}} = \text{Concat}(\mathbf{X}^t_{\text{token}}, \mathbf{X}^t_{\mathbf{0}}) \in \mathbb{R}^{(L_{\text{token}}+L_y)\times d_{\text{model}}} \quad , \tag{21}$$

where $\mathbf{X}^t_{\text{token}} \in \mathbb{R}^{L_{\text{token}}\times d_{\text{model}}}$ is the start token, $\mathbf{X}^t_{\mathbf{0}} \in \mathbb{R}^{L_y\times d_{\text{model}}}$ is a placeholder for the target sequence (set scalar as 0). Masked multi-head attention is applied in the *ProbSparse* self-attention computing by setting masked dot-products to $-\infty$. It prevents each position from attending to coming positions, which avoids auto-regressive. A fully connected layer acquires the final output, and its outsize d_y depends on whether we perform univariate or multivariate forecasting.

4.3.1. Generative inference

Start token is efficiently applied in NLP's "dynamic decoding" [41], and we extend it into a generative way. Instead of choosing specific flags as the token, we sample a L_{token} long sequence in the input sequence, such as an earlier slice before the output sequence. Take predicting 168 points as an example (7-day temperature prediction in the experiment section), we will take the known 5 days before the target sequence as "start-token", and feed the generative-style inference decoder with $\mathbf{X}_{\text{de}} = \{\mathbf{X}_{5d}, \mathbf{X}_{\mathbf{0}}\}$. $\mathbf{X}_{\mathbf{0}}$ contains the target sequence's time stamp, i.e., the context at the target week. Then our proposed decoder predicts outputs by one forward procedure rather than the time-consuming "dynamic decoding" in the conventional encoder-decoder architecture. A detailed performance comparison is given in the computation efficiency section.

4.3.2. Mixed multi-head attention

We propose the mixed multi-head attention mechanism to enlarge the perception field of the multi-head attention for specific data distributions. The adjacent time slices in sequence contain coherent information. Therefore, we design the mixed multi-head attention to enable each time slice to exchange information with nearby slices, which is the natural choice from the convolutional filters on inputs in Eq. (2). It provides different data perspectives for the following fully connected layers. The most straightforward implementation is to change the Concat$(\cdot)$ operation at the feature map of multi-head self-attention:

$$\begin{aligned} \mathcal{MA}(\mathbf{X}) &= \text{Mixed-Concat}(\text{head}_1, \ldots, \text{head}_n) \\ \text{where }\ \text{head}_k &= \mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \end{aligned} \quad , \tag{22}$$

where the Mixed-Concat$(\cdot)$ denotes exchanging the time dimensional index with the head index during the tensor concatenation.

4.3.3. Loss function

We choose the MSE loss function on prediction $\hat{\mathcal{Y}}^t$ w.r.t. the target sequences $\mathcal{Y}^t$:

$$L_{\text{MSE}} = \frac{1}{L_y}\sum_{i=1}^{L_y}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2 \quad , \tag{23}$$

and the loss is propagated back from the decoder's outputs across the entire model. The prediction could also be extend to include the "start token" to involve more gradients updating.

4.4. Analysis

We analyze how the boundary relaxation in Eq. (13) affects the Top-u query selection in the *ProbSparse* Self-attention. The discussion begins with the final result:

Lemma 2. *Assuming $\mathbf{k}_j \sim \mathcal{N}(\mu, \Sigma)$, then $\forall M_m = \max_i M(\mathbf{q}_i, \mathbf{K})$, there exists $\kappa > 0$ such that $\forall \mathbf{q}_1, \mathbf{q}_2 \in \{\mathbf{q}|M(\mathbf{q}, \mathbf{K}) \in (M_m - \kappa, M_m]\}$, if $\overline{M}(\mathbf{q}_1, \mathbf{K}) > \overline{M}(\mathbf{q}_2, \mathbf{K})$ in the interval, we have a high probability that $M(\mathbf{q}_1, \mathbf{K}) > M(\mathbf{q}_2, \mathbf{K})$.*

Proof. To make the further discussion simplify, we let the pairwise attention score $a_{i,j} = \mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}$ for $i = 1, \cdots, L_Q$ and $j = 1, \cdots, L_K$, then we have the i-th query array $\mathcal{A}_i = [a_{i,1}, \cdots, a_{i,L_k}]$ running through all keys. Defining the last term of $\overline{M}$ as $\operatorname{mean}_j(\mathcal{A}_i) = \frac{1}{L_K}\sum_{j=1}^{L_K}(\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d})$, we can immediately rewrite it into $\overline{M}(\mathbf{q}_i, \mathbf{K}) = \max_j(\mathcal{A}_i) - \operatorname{mean}_j(\mathcal{A}_i)$. As for $M(\mathbf{q}_i, \mathbf{K})$, we rewrite pairwise attention score as $a_{i,j} = \operatorname{mean}_j(\mathcal{A}_i) + \Delta a_{i,j}$, where $\sum_{j=1}^{L_k}\Delta a_{i,j} = 0$. Then we can derive from Eq. (6):

$$\begin{aligned} M(\mathbf{q}_i, \mathbf{K}) &= \ln\sum_{j=1}^{L_K} e^{\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}} - \frac{1}{L_K}\sum_{j=1}^{L_K}(\mathbf{q}_i\mathbf{k}_j^\top/\sqrt{d}) \\ &= \ln(\sum_{j=1}^{L_k} e^{\operatorname{mean}_j(\mathcal{A}_i)} e^{\Delta a_{i,j}}) - \operatorname{mean}_j(\mathcal{A}_i) \\ &= \ln(\sum_{j=1}^{L_k} e^{\Delta a_{i,j}}) \end{aligned} \quad . \tag{24}$$

In the following description, we omit the j in $\max(\cdot)$ and $\operatorname{mean}(\cdot)$ for simplicity. Recalling that $\mathbf{k_j} \sim \mathcal{N}(\mu, \Sigma)$ follows multivariate Gaussian distribution, which means that $\mathbf{k}_1, \cdots, \mathbf{k}_n$ are i.i.d. Gaussian distributions. So $\mathcal{A}_i$ are i.i.d. variables and we have $\mathcal{A}_i \sim \mathcal{N}(\operatorname{mean}(\mathcal{A}_i), \sigma_i^2)$ for $i \in \{1, \cdots, L_Q\}$.

Next, we define the bias term as $A_i = [\Delta a_{i,1}, \cdots, \Delta a_{i,L_k}]$. If we consider that $L_k \to \infty$ under the Wiener-Khinchin law of large numbers, we have $A_i \sim \mathcal{N}(0, \sigma_i^2)$ for $i \in \{1, \cdots, L_Q\}$ based upon the constraint $\sum_{j=1}^{L_k}\Delta a_{i,j} = 0$. We define the sum of Log-normal as $\operatorname{SLN}(A_i) = \Sigma_{j=1}^{L_k}\exp(\Delta a_{i,j})$, and we select following two sets:

$$\begin{aligned} A^1 &= \{\max(A_1), \max(A_2), \cdots, \max(A_n)\} \\ A^2 &= \{\operatorname{SLN}(A_1), \operatorname{SLN}(A_2), \cdots, \operatorname{SLN}(A_n)\} \end{aligned} \quad .$$

Then we build the top-u subset of A^1 and A^2 as A^1_{top-u} and A^2_{top-u} by a descending order. The A^1 represents the $\overline{M}$ measurement while the A^2 is associated with M. Before we further explore the distribution of $M(\mathbf{q}_i, \mathbf{K})$, we reformulate the Lemma 2 into its equivalent form:

Proposition 1. *For the set $A = \{A_1, \cdots, A_{L_Q}\}$, we assume that $\{\sigma_1^2, \sigma_2^2, \cdots, \sigma_{L_Q}^2\}$ follows a truncated Gaussian distribution with a value $\mu(\sigma^2) = \operatorname{mean}(\sigma_1^2, \cdots, \sigma_{L_Q}^2)$. If $\max(A_i) \in A^1_{top-u}$, then $\operatorname{SLN}(A_i) \in A^2_{top-u}$ holds in the probability:*

$$p \geq \Phi(\frac{\sqrt{\mu(\sigma^2)} + \ln(2)}{\exp(0.5)})(1 - (\Phi(1))^{L_K})^{L_Q} - \epsilon \quad ,$$

where $\Phi(x)$ is the standard normal distribution function and ϵ is small. In the empirical evaluation, we choose $L_Q, L_K \in [500, 1000]$ and $u < \lfloor\frac{L_Q}{2}\rfloor$. Then, we have a probability over 99% that the conclusion stands when $\mu(\sigma^2) \geq 9$, $\exp(\mu(\sigma^2)) \gg L_K$.

We divide the proof into four parts, and the proof sketch is given as:

- Firstly, we introduce the preliminary in approximating the sum and the difference of Log-normal distributions, which is based on the shifted normal process transferred through a shifted Log-normal process. (Section 4.4.1)
- Secondly, we prove the probability $Pr(\mathrm{SLN}(A_1) - \mathrm{SLN}(A_2) \geq 0)$ depends on the $\mu(\sigma^2)$ and the initial values S_{10}, S_{20}. (Section 4.4.2)
- Then, we extend the conclusion to the top-u case, which reveals that Lemma 2 and Proposition 1 are equivalent. (Section 4.4.3)
- And we give a numerical simulation for empirical evaluation. (Section 4.4.4)

Combining above results, we have the final conclusion. □

4.4.1. Part 1: reformulation to the difference of two Log-normal distributions

We compare two individual sums, namely $S_1 = \mathrm{SLN}(A_1)$ and $S_2 = \mathrm{SLN}(A_2)$, with the condition that $\max(A_1) > \max(A_2)$. Recall that $M(\mathbf{q}_i, \mathbf{K}) = \ln(S_i)$ from Eq. (24), the probability $p_{1,2} = Pr(S_1 - S_2 > 0)$ is the major result of Proposition 1.

The Log-normal distribution sum problem equals approximating the distribution of $\mathrm{SLN}(A_i)$, which has been well-introduced [67,68]. However, there could hardly be a tangible probability density function (PDF) for the sum/difference of two Log-normal distributions or more. In most cases [69,70], if $X_1, \cdots, X_n$ are i.i.d. log-normal distributed variables, then the sum $Y = \Sigma_{j=1}^{n} X_j$ can be reasonably approximated by a distribution in the right tail [71]. From the prevailing right-tail estimators, Fenton–Wilkinson method [72,73] builds another log-normal distribution at the right tail based on matching the mean and variance, which has been applied in finance [74], telecommunication [75] and insurance [76].

Using the Fenton–Wilkinson method, S_1, S_2 can be approximated with two log-normal distributions respectively, denoted as Z_1, Z_2. Accordingly, the variances of the log-normal distributions $Z_1 \sim Lognormal(\mu_{Z_1}, \sigma_{Z_1}^2)$, $Z_2 \sim Lognormal(\mu_{Z_2}, \sigma_{Z_2}^2)$ can be reasonably calculated [73] as:

$$\begin{aligned}
\sigma_{Z_1}^2 &= \ln\left[\left(e^{\sigma_1^2} - 1\right)\frac{\sum e^{2\mu}}{\left(\sum e^{\mu}\right)^2} + 1\right] \approx \sigma_1^2 - \ln(L_K) \quad , \\
\sigma_{Z_2}^2 &= \ln\left[\left(e^{\sigma_2^2} - 1\right)\frac{\sum e^{2\mu}}{\left(\sum e^{\mu}\right)^2} + 1\right] \approx \sigma_2^2 - \ln(L_K) \quad .
\end{aligned} \tag{25}$$

The approximation holds when estimating the right tail for the sum of Log-normal distributions, which also requires $\exp(\sigma_i^2) \gg L_K$ in the Proposition 1, namely $\sigma_i^2 > \ln(L_K)$ for all elements $A_i \in A_{top-u}^1$.

Then we utilize the $Z = Z_1 - Z_2$ to measure the difference of two Log-normal distributions S_1 and S_2. Note that the error bound between the measurement is small [69,71] but has no close-form. Based on the theorem [77], S_1 and S_2 are two log-normal stochastic variables depending on the stochastic differential equations $\frac{dS_i}{S_i} = \sigma_i' dW_i$, $i = 1, 2$. $dW_{1,2}$ presents a standard Weiner process associated with $S_{1,2}$ respectively, and $\sigma_i'^2 = \sigma_{Z_i}^2 = \sigma_i^2 - \ln(L_K) = \mathrm{Var}(\ln S_i)$, $S^{\pm} \equiv S_1 \pm S_2$, $S_0^{\pm} \equiv S_{10} \pm S_{20}$. As for the joint probability distribution function $P(S_1, S_2, t; S_{10}, S_{20}, t_0)$, the values of S_1 and S_2 at time $t > t_0$ are acquired by initial value S_{10} and S_{20} at initial time t_0. We follow the same assumption [77] and set $S_{10} = \exp(\max(A_1))$, $S_{20} = \exp(\max(A_2))$ as the initial values for S_1, S_2 at time t_0.

Next, by adopting Lie-Trotter splitting method [78,77], we can deduce that $S^- = S_1 - S_2$ is shifted log-normal distributed $f^{LN}(x)$, which is guided by Eq. (2.11) in [77]. The scaling coefficient $\tilde{S}_0^-$ is:

$$\begin{aligned}
\tilde{S}_0^- &= (S_{10} - S_{20}) + \left(\frac{\sigma_1^2 + \sigma_2^2 - 2\ln(L_K)}{\sigma_1^2 - \sigma_2^2}\right)(S_{10} + S_{20}) \\
\tilde{\sigma}_-^2 &= \frac{(\sigma_1^2 - \sigma_2^2)^2}{4(\sigma_1^2 + \sigma_2^2 - 2\ln(L_K))}
\end{aligned} \quad . \tag{26}$$

Actually, a shifted log-normal distribution can be transformed into a shifted normal distribution. To simplify the calculation of probability p from $f^{LN}(x)$, we have a shifted normal distribution $f^N(x)$ with the scaling coefficient $\ln(\tilde{S}_0^-)$:

$$\sigma_X^2 = e^{\tilde{\sigma}_-^2} - 1, \qquad \mu_X = 1.$$

Note that above the derivation requires the constraint $\epsilon = \tilde{\sigma}_-^2(t - t_0) \leq 1$ holds.

4.4.2. Part 2: approximation of the probability

The variance of the shifted normal distribution $f^N(x)$ is σ_X^2. Compared to the standard normal distribution PDF curve, the shifted normal distribution PDF curve $f^N(x)$ actually moves to the right at the length $L = \frac{\ln(\tilde{S}_0^-)}{\sigma_X}$, where we have $\tilde{S}_0^- \geq S_{10} - S_{20} \geq 0$. Then we will show that L decides the probability $Pr(S_1 - S_2 > 0)$.

Corollary 2. *$Pr(\bigcap_{i=1}^{L_Q} \max(A_i) \geq \sigma_i) \to 1$ when $L_Q \leq 1000$. Specifically, the condition $\mathrm{SLN}(A_i) > \exp(\max(A_i)) \geq \exp(\sigma_i)$ holds at the probability $P_1 = (1-(\Phi(1))^{L_K})^{L_Q}$.*

Proof. Considering $A_i \sim \mathcal{N}(0, \sigma_i^2)$ and $i \in [1, \cdots, L_Q]$, if all elements in A_i are less than σ_i, namely $\max(A_i) < \sigma_i$, then the probability of event $Pr(\max(A_i) < \sigma_i)$ is $(\Phi(1))^{L_K}$, where $\Phi(x)$ is the standard normal distribution function. For the inverse event of $\max(A_i) < \sigma_i$, we have:

$$Pr(\max(A_i) \geq \sigma_i) = 1 - (\Phi(1))^{L_K} \qquad ,$$

where $\Phi(1) \approx 0.86$ and $L_K \geq 500$. Furthermore, if for all $A_i \in A$, $\max(A_i)$ is greater than or equal to σ_i respectively, which is equal to the event $\bigcap_{i=1}^{L_Q} \max(A_i) \geq \sigma_i$, the probability depends on L_Q-times events:

$$Pr(\bigcap_{i=1}^{L_Q} \max(A_i) \geq \sigma_i) = (1-(\Phi(1))^{L_K})^{L_Q} \qquad ,$$

where $L_K \geq 500, L_Q \leq 1000$ and $A_1, \cdots, A_{L_Q}$ are i.i.d. variables. Through a simple numerical estimation, we have a high probability that $\max(A_i) \geq \sigma_i$, namely $\mathrm{SLN}(A_i) \geq \exp(\sigma_i)$ holds for $\forall i \in [1, \cdots, L_Q]$, where the error is less than 10^{-5}. □

Corollary 3. *If $\max(A_i) \in A^1_{top-u}$, then we have $\max(A_i)^2 \geq \mu(\sigma^2)$.*

Proof. We define two subsets of A^1 as $B^1 = \{\max(A_j) | \max(A_j)^2 \geq \mu(\sigma^2)\}$, $B^2 = \{\max(A_j) | \sigma_j^2 \geq \mu(\sigma^2)\}$. By using Corollary 2 and $\forall A_j \in B^2$, we have $\max(A_j)^2 \geq \sigma_j^2 \geq \mu(\sigma^2)$, then $A_j \in B^1$, which leads to $B_2 \subset B_1$. Recall the assumption of truncated Gaussian distribution on σ_i^2, we have $|B^1| \geq |B^2| = \lfloor \frac{L_Q}{2} \rfloor > |A^1_{top-u}|$. Then we use the approach of contradiction proof. If $\exists i \in [1, \cdots, n]$, the condition $\max(A_i) \in A^1_{top-u}$ holds. However $\max(A_i)^2 < \mu(\sigma^2)$, which means $\max(A_i) \notin B^1$. Then $\forall j$ that $\max(A_j) \in B^1$, we have $\max(A_j)^2 \geq \mu(\sigma^2) > \max(A_i)^2$. It leads to the conclusion $\max(A_j) \in A^1_{top-u}$, because A^1_{top-u} is the subset containing Top-u elements of A^1. Thus we have $B^1 \subset A^1_{top-u}$, however $|A^1_{top-u}| = u < \lfloor \frac{L_Q}{2} \rfloor \leq |B^1|$, which is contradiction. So we have $A^1_{top-u} \subset B^1$, and the above corollary holds. □

Corollary 4. *$\tilde{S}_0^- \geq 2\exp(\sqrt{\mu(\sigma^2)})$ holds at the probability $P_1 = (1-(\Phi(1))^{L_K})^{L_Q}$.*

Proof. Based on the definition of $\tilde{S}_0^-$ in Eq. (26), we have $\tilde{S}_0^- \geq S_{10} + S_{20}$. From Corollary 2 and Corollary 3, we derive the conclusion that $S_{10} \geq S_{20} \geq \exp(\sqrt{\mu(\sigma^2)})$ holds at probability $P_1 = (1-(\Phi(1))^{L_K})^{L_Q}$, so we reach the corollary's result. □

Furthermore, we perform the probability estimation. From the constraint of Lie-Trotter method [78,77], we have $\tilde{\sigma}_-^2(t - t_0) \leq 1$. We follow the setting $t - t_0 = 1$ [77] and it obtains $\tilde{\sigma}_-^2 \leq 1$. Combining the two inequality, we have the length L as:

$$\Phi(\frac{\ln(\tilde{S}_0^-)}{\sigma_X}) = \Phi(\frac{\ln(\tilde{S}_0^-)}{\sqrt{\exp(\tilde{\sigma}_-^2 - 1)}}) \geq \Phi(\frac{\ln(\tilde{S}_0^-)}{\exp(0.5)}) \geq \Phi(\frac{\sqrt{\mu(\sigma^2)} + \ln(2)}{\exp(0.5)}) .$$

Considering the probability P_1 in Corollary 4, the main result is:

$$p = Pr(S_1 - S_2 > 0) \geq \Phi(\frac{\mu(\sigma) + \ln(2)}{\exp(0.5)})(1-(\Phi(1))^{L_K})^{L_Q} \qquad . \tag{27}$$

We give a numerical simulation in Section 4.4.4, where the settings are $L_K = 1000$, $n = 500$ and $\mu(\sigma) = 5$, then $Pr(S_1 - S_2 > 0) \geq 99\%$. From the theoretical perspective, we can easily find the probability highly relies on the initial values $\{S_{10}, S_{20}\}$, the difference $S_{10} - S_{20}$ and the mean value of variances $\mu(\sigma^2)$. Firstly, we know that approximating $\tilde{S}_0^-$ by $S_{10} - S_{20}$ is suitable in most conditions [77]. So if $S_{10} - S_{20}$ is small, namely the distributions of $\{A_1, A_2\}$ are equal to some extent, the term $\Phi(\frac{\ln(\tilde{S}_0^-)}{\sqrt{\exp(\tilde{\sigma}_-^2 - 1)}})$ will be small enough (below 50%) to break the conclusion, which fits our numerical simulation. Secondly, from Corollary 3, the probability $Pr(S_1 - S_2 > 0)$ is related to $\mu(\sigma^2)$, which builds an adequate lower bound for $\tilde{S}_0^-$.

4.4.3. Part 3: extension to the whole set and the corresponding interval

We will deduce the above derivation into the general case in Proposition 1. For the whole set A, we select two subsets $A_{top-u}^{\max}, A_{top-u}^{\mathrm{SLN}} \subseteq A$, and they satisfy:

$$A_{top-u}^{\max} = \{A_i | \max(A_i) \in A_{top-u}^{1}\}, \qquad A_{top-u}^{\mathrm{SLN}} = \{A_i | \mathrm{SLN}(A_i) \in A_{top-u}^{2}\}.$$

In other words, $A_{top-u}^{\max}, A_{top-u}^{\mathrm{SLN}}$ are partial elements in A constructing $A_{top-u}^{1}, A_{top-u}^{2}$ respectively. Since we select the top-u elements, $|A_{top-u}^{1}| = |A_{top-u}^{2}| = |A_{top-u}^{\max}| = |A_{top-u}^{\mathrm{SLN}}| = u$. The previous Top-$u$ conclusion is equal to prove that $|A_{top-u}^{\max} \cap A_{top-u}^{\mathrm{SLN}}| = u$ holds at a high probability.

Rolling back to our problem, if the condition $A_{top-u}^{\max} \neq A_{top-u}^{\mathrm{SLN}}$, which indicates

$$U = A_{top-u}^{\max} \cap A_{top-u}^{\mathrm{SLN}}, \qquad |U| = u' < u.$$

We let $M = A_{top-u}^{\max} \setminus U, N = A_{top-u}^{\mathrm{SLN}} \setminus U$, where $|M| = |N| = u - u' = \Delta u$. Immediately, we can rewrite them into $M = \{A_{m_1}, \cdots, A_{m_{\Delta u}}\}$, $N = \{A_{n_1}, \cdots, A_{n_{\Delta u}}\}$. So $\forall i, j \in [1, \cdots, \Delta u]$, we have:

$$\max(A_{m_i}) > \max(A_{n_j}), \qquad \mathrm{SLN}(A_{m_i}) < \mathrm{SLN}(A_{n_j}), \tag{28}$$

which is the intersection of $(\Delta u)^2$-times events. Moreover, the probability $p_{\Delta u} = Pr(|M| = |N| = \Delta u)$ has a close-form formulation as:

$$\begin{aligned} p_{\Delta u} &= Pr(\bigcap_{i,j=1}^{\Delta u} [\max(A_{m_i}) > \max(A_{n_j}), \mathrm{SLN}(A_{m_i}) < \mathrm{SLN}(A_{n_j})]) \\ &= (1-p)^{(\Delta u)^2} \end{aligned} . \tag{29}$$

For the event that $A_{top-u}^{\max}$ and A_{top-u}^{SLN} are totally overlapped, namely the total number of element reaches $|U| = u$, we have the following probability:

$$\begin{aligned} Pr(|U| = u) =& 1 - \sum_{\Delta u=1}^{u} p_{\Delta u} \\ =& 1 - [(1-p) + (1-p)^{2^2} + \cdots + (1-p)^{u^2}] \\ =& p - \epsilon \end{aligned} , \tag{30}$$

where $\epsilon = \sum_{i=2}^{u} (1-p)^{i^2} < (1-p)^4 + (1-p)^5 + \cdots = \frac{(1-p)^4}{p}$. Then, the Proposition 1 matches the top-u isotonicity conclusion, where the element selection in building A_{top-u}^{1} and A_{top-u}^{2} from A is totally overlapped at a high probability $p - \epsilon$. Note that if $p > 99\%$, as in the case study, we have $\epsilon < 10^{-4}$.

From Lemma 1, we have $M(\mathbf{q}_i, \mathbf{K}) \geq \ln L_K$. Within $M(\mathbf{q}_i, \mathbf{K}) \in [\ln L_K, M_m]$ and $\forall u < \lfloor \frac{L_Q}{2} \rfloor$, there exists $\kappa < (0, M_m - \ln L_K]$ that satisfies $|\{\mathbf{q_i} | M(\mathbf{q}_i, \mathbf{K}) \in (M_m - \kappa, M_m]\}| = u$ by sorting the elements of set $[M(\mathbf{q}_i, \mathbf{K}) | i \in [1, L_Q]]$ in a descending order. Then we can easily choose the $u-th$ element $M(\mathbf{q}_u, \mathbf{K})$ within $\kappa = M_m - M(\mathbf{q}_u, \mathbf{K})$. Finally, the Proposition 1 holds at probability $p - \epsilon$.

4.4.4. Part 4: the numerical simulation

We perform an illustrative numerical simulation [77] of the probability approximation in Fig. 9. We draw the curve of the shifted Log-normal distribution $f^{LN}(x)$ under different $\{S_{10}, S_{20}, \sigma_1^2, \sigma_2^2\}$ settings. By adding the value of $S_{10} - S_{20}$ from 50 to 100, the bell-shaped curve shifted to the right direction. Meanwhile, the probability density $Pr(S_1 - S_2 > 0)$ gets closer to 1, which is the area under the curve in the first quadrant. This phenomenon also applies to adding the $\mu(\sigma^2)$. Generally speaking, if the set A has larger maximum elements or larger variance, we will have a higher probability that Lemma 2 holds.

4.4.5. Part 5: connections to $\widehat{M}$

We will illustrate that $\overline{M}(\mathbf{q}_i, \mathbf{K})$ can be a rough KL divergence estimator, and is proportion to kNN estimator Eq. (8) in the long-tail distribution scenario with the random sampling strategy. We divide the sample points $\hat{Y}$ in a dense subset S_1 and a sparse subset S_2. Suppose p obeys a long-tail distribution, which means S_2 contains most of the sample points and is distributed uniformly close to the tail part. In that case, we can estimate $\widehat{KL}(q||p)$ by calculating the length of S_2, which is equivalent to the length of the tail for a long-tail distribution. Naturally, it could be approximated by $\overline{M}(\mathbf{Y}) = \max(\mathbf{Y}) - \mathrm{mean}(\mathbf{Y})$. Intuitively, the discussion holds in the assumption that the dense subset S_1 contains few sample points, the sparse subset S_2 contains most of the sample points, and the sample points in S_2 uniformly distribute when they are close to the tail. Then, we can estimate the KL divergence by the tail length. Thus, Max-mean measurement can be conducted on the Random/Max sampling strategy in the long-tail distribution.

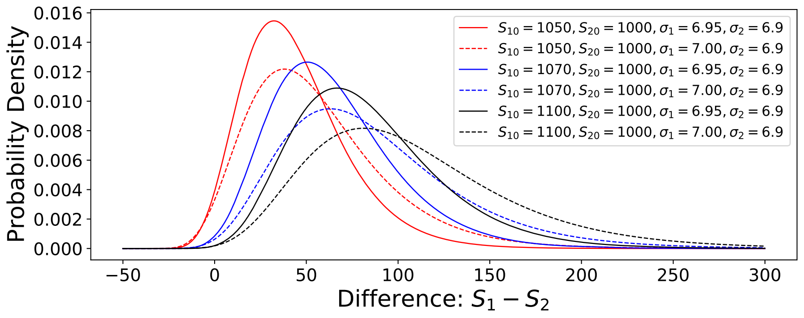


Fig. 9. The results of numerical simulation. We approximate the $p = Pr(M(\mathbf{q}_1, \mathbf{k}) > M(\mathbf{q}_2, \mathbf{k}))$, $\{S_{10}, S_{20}\}$ represent the initial values of the Weiner process mentioned in Section 4.4.1, and σ_1, σ_2 denote the variance for A_1, A_2 respectively. Under the top-u scenario, the three solid/dash lines with same color show that $\mu(\sigma^2)$ has a positive relation with p. And the solid/dash lines in different colors show that with the same variance σ_1, σ_2, the difference of initial value $S_{10} - S_{20}$ also has a positive correlation with p.

4.5. Limitations and following-up works

The proposed Informer improves the model capacity dramatically by an efficient *ProbSparse* self-attention mechanism. On the other side, it inevitably makes the proposed Informer easily over-fitting to the short sequence data. Recently, there has been rapid progress in applying Transformer models in the time-series. The FEDformer [79] leverages frequency tricks to reduce the quadratic attention computation and acquire a linear one. The autoformer [80] adds an auto-correlations mechanism to the self-attention for better time-series modeling. A complete survey could be found [81]. Almost all the works follow the Informer's generative style decoder, and they can generate the outputs from one-step forwarding. And the manipulations on the attention feature map could be coupled with the *ProbSparse* attention for its query-wise reduction of computing complexity, which also helps alleviate the over-fitting problem.

5. Experiment

In this section, we conduct extensive experiments and compare the performances of ten methods (including two our methods) on ten datasets.

5.1. Datasets

We extensively perform experiments on ten datasets, including four collected real-world datasets for LSTF and six public benchmark datasets.

ETT (Electricity Transformer Temperature)[1]: The ETT is a crucial indicator in the electric power long-term deployment, which is affected by massive periodical factors, and the target of forecasting several days ahead makes the problem intractable. We collected 2-year data from four separated counties in China. To explore the granularity on the LSTF problem, we create separate datasets as {ETTh$_1$, ETTh$_2$, ETTh$_3$, ETTh$_4$} for 1-hour-level and {ETTm$_1$, ETTm$_3$, ETTm$_4$} for 15-minute-level. Each data point consists of 7 features, including the predictive value "oil temperature", and 6 different types of power load features. The train/val/test is 12/4/4 months.

ECL (Electricity Consuming Load)[2]: It collects the electricity consumption (Kwh) of 321 clients. Due to the missing data [16], we convert the dataset into hourly consumption of 2 years and set 'MT_320' as the target value. The train/val/test is 15/3/4 months.

Weather[3]: This dataset contains local climatological data for nearly 1,600 U.S. locations, 4 years from 2010 to 2013, where data points are collected every 1 hour. Each data point consists of 12 features, including the predictive feature "wet bulb" and 11 different climate/weather features. The train/val/test is 28/10/10 months.

PM2.5[4]: Beijing Multi-Site Air-Quality Data Data Set contains hourly air pollutants data of 12 nationally-controlled air-quality monitoring sites from the Beijing Municipal Environmental Monitoring Center. The period is from March 1st, 2013 to February 28th, 2017, and data points are collected every 1 hour. The meteorological data in each air-quality site are matched with the nearest weather station from the China Meteorological Administration. We select the data of the Nongzhanguan monitoring site as the PM2.5 dataset. The train/val/test is 255/37/73 days in a chronological order.

Solar[5]: The solar dataset contains the solar power production records sampled every 10 minutes from 137 PV plants in Alabama State in 2006. The train/val/test is 255/37/73 days in a chronological order.

[1] We collected the ETT dataset and published it at https://github.com/zhouhaoyi/ETDataset.
[2] ECL dataset was acquired at https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.
[3] Weather dataset was acquired at https://www.ncei.noaa.gov/data/local-climatological-data/.
[4] PM2.5 dataset was acquired at https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data.
[5] Solar dataset was acquired at: http://www.nrel.gov/grid/solar-power-data.html.

Table 1
The Informer network components in details.

Encoder:			N
Inputs	1x3 Conv1d	Embedding ($d = 512$)	3
ProbSparse Self-attention Block	Multi-head ProbSparse Attention ($h = 16$, $d = 32$)		
	Add, LayerNorm, Dropout ($p = 0.1$)		
	Pos-wise FFN ($d_{\text{inner}} = 2048$), GELU		
	Add, LayerNorm, Dropout ($p = 0.1$)		
Distilling	1x3 conv1d, ELU		
	Max pooling (stride $= 2$)		
Decoder:			N
Inputs	1x3 Conv1d	Embedding ($d = 512$)	2
Masked PSB	add Mask on Attention Block		
Self-attention Block	Multi-head Attention ($h = 8$, $d = 64$)		
	Add, LayerNorm, Dropout ($p = 0.1$)		
	Pos-wise FFN ($d_{\text{inner}} = 2048$), GELU		
	Add, LayerNorm, Dropout ($p = 0.1$)		
Final:			
Outputs	FCN ($d = d_{\text{out}}$)		

Traffic[6]: The traffic dataset contains 15 months of daily data from the California Department of Transportation. It describes the occupancy rate (0 1) of different car lanes of San Francisco bay area freeways. The data points are collected every 10 minutes, and the columns were aggregated to obtain hourly traffic data. It contains 963 sequences, and the length is 10,560.

Exchange-Rate[7]: The exchange rate dataset is a collection of the daily exchange rates of eight countries: Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore. The data is collected from 1990 to 2016.

5.2. *Experimental details*

We briefly summarize the basics, network components and setups in this section.

5.2.1. *Baselines*

We have selected five time-series forecasting methods as comparison, including ARIMA [82], Prophet [54], LSTMa [38] and LSTnet [4] and DeepAR [28]. To better explore the *ProbSparse* self-attention's performance in our proposed Informer, we incorporate the canonical self-attention variant (Informer†), the efficient variant Reformer [19] and the most related work LogSparse self-attention [16] in the experiments. The details of network components are given in Table 1.

5.2.2. *Hyper-parameter tuning*

We conduct grid search over the hyper-parameters. For Informer, the layer of encoder is chosen from {6, 4, 3, 2} and the layer of decoder is set as 2. The head number of multi-head attention is chosen from {8, 16}, and the dimension of multi-head attention's output is set as 512. Informer contains a 3-layer stack and a 2-layer stack (1/4 input) in the encoder, and a 2-layer decoder. The decoder's start token is a segment truncated from the encoder's input sequence, so the length of decoder's start token must be less than the length of encoder's input. Our proposed methods are optimized with Adam optimizer and its learning rate starts from $1e^{-4}$, decaying 10 times smaller every 2 epochs and the total epochs is 5. We set the comparison methods as recommended and the batch size is 32. **Setup:** The input of each dataset is zero-mean normalized. Under the LSTF settings, we prolong the prediction windows size L_y progressively, i.e., {1d, 2d, 7d, 14d, 30d, 40d} in {ETTh, ECL, Weather}, {6h, 12h, 24h, 72h, 168h} in ETTm. The Prophet use the series-to-point prediction setting. The RNN-based methods perform a dynamic decoding with left shifting on the prediction windows. Our proposed Informer and Informer† performs generative decoding. **Metrics:** We use two evaluation metrics, including $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y} - \hat{\mathbf{y}})^2$ and $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\mathbf{y} - \hat{\mathbf{y}}|$ on each prediction window (averaging for multivariate prediction), and roll the whole set with stride $= 1$. **Platform:** All the models were trained/tested on a single Nvidia V100 32GB GPU. The source code is available at https://github.com/zhouhaoyi/Informer2020.

5.3. *Results and analysis*

Table 2 and Table 3 summarize the univariate/multivariate evaluation results of all the methods on eight datasets. We gradually prolong the prediction horizon as a higher requirement of prediction capacity, where the LSTF problem setting is precisely controlled to be tractable on one single GPU for each method. The best results are highlighted in boldface.

[6] Traffic dataset was acquired at: https://archive.ics.uci.edu/ml/datasets/PEMS-SF.
[7] Exchange-Rate dataset was acquired at: https://github.com/laiguokun/LSTNet.

Table 2
Univariate long sequence time-series forecasting results on ten datasets.

Methods		Informer		Informer†		LogTrans		Reformer		LSTMa		DeepAR		ARIMA		Prophet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh$_1$	24	0.072	0.206	**0.058**	**0.186**	0.083	0.229	0.222	0.389	0.114	0.272	0.107	0.280	0.108	0.230	0.115	0.275
	48	**0.122**	**0.273**	0.141	0.302	0.131	0.293	0.284	0.445	0.193	0.358	0.142	0.327	0.155	0.274	0.168	0.330
	168	**0.172**	**0.330**	0.207	0.375	0.187	0.355	1.522	1.191	0.236	0.392	0.239	0.422	0.396	0.504	1.224	0.763
	336	0.242	0.417	**0.225**	**0.398**	0.230	0.428	1.860	1.124	0.590	0.698	0.445	0.552	0.468	0.593	1.549	1.820
	720	0.269	0.435	**0.257**	**0.421**	0.273	0.463	2.112	1.436	0.683	0.768	0.658	0.707	0.659	0.766	2.735	3.253
ETTh$_2$	24	**0.093**	**0.240**	0.099	0.241	0.102	0.255	0.263	0.437	0.155	0.307	0.098	0.263	3.554	0.445	0.199	0.381
	48	**0.115**	**0.270**	0.129	0.277	0.129	0.298	0.458	0.545	0.190	0.348	0.141	0.321	3.190	0.474	0.304	0.462
	168	**0.169**	**0.334**	0.180	0.350	0.206	0.392	1.029	0.879	0.385	0.514	0.205	0.394	2.800	0.595	2.145	1.068
	336	0.205	**0.375**	**0.198**	0.377	0.217	0.397	1.668	1.228	0.558	0.606	0.604	0.607	2.753	0.738	2.096	2.543
	720	**0.232**	**0.395**	0.245	0.402	0.263	0.413	2.030	1.721	0.640	0.681	0.429	0.580	2.878	1.044	3.355	4.664
ETTh$_3$	24	**0.084**	0.211	0.091	0.214	0.098	0.229	0.381	0.267	0.330	**0.194**	0.475	0.378	0.211	0.282	0.354	0.199
	48	**0.138**	**0.285**	0.139	0.291	0.180	0.286	0.627	0.774	0.467	0.379	0.434	0.335	0.294	0.355	0.462	0.351
	168	**0.198**	0.357	0.207	**0.356**	0.239	0.373	1.367	2.818	0.679	0.699	0.745	0.822	0.486	0.465	1.153	2.510
	336	**0.258**	**0.410**	0.272	0.433	0.260	0.457	1.757	4.723	0.552	0.612	0.741	0.925	0.604	0.540	2.819	18.71
	720	**0.230**	**0.390**	0.236	0.428	0.235	0.400	2.993	6.832	1.075	1.414	1.226	1.901	0.802	0.912	5.138	59.55
ETTh$_4$	24	**0.257**	0.392	**0.257**	0.398	0.269	0.397	0.433	0.426	0.443	0.376	0.377	0.267	0.376	0.466	0.351	**0.227**
	48	**0.396**	0.509	0.401	0.514	0.433	0.547	0.534	0.632	0.621	0.639	0.608	0.578	0.423	0.573	0.404	**0.283**
	168	0.427	0.558	**0.425**	**0.553**	0.470	0.599	0.987	1.537	0.749	0.912	0.949	1.307	0.526	0.620	0.715	1.004
	336	**0.489**	0.604	0.494	**0.590**	0.494	0.646	1.550	3.164	1.035	1.391	0.857	1.125	0.599	0.677	1.594	6.503
	720	0.567	**0.655**	**0.555**	0.665	0.592	0.700	1.925	3.661	0.945	1.374	0.834	1.170	0.684	0.749	4.351	59.26
Weather	24	**0.117**	**0.251**	0.119	0.256	0.136	0.279	0.231	0.401	0.131	0.254	0.128	0.274	0.219	0.355	0.302	0.433
	48	**0.178**	0.318	0.185	**0.316**	0.206	0.356	0.328	0.423	0.190	0.334	0.203	0.353	0.273	0.409	0.445	0.536
	168	**0.246**	**0.388**	0.259	0.404	0.309	0.439	0.654	0.634	0.341	0.448	0.293	0.451	0.503	0.599	2.441	1.142
	336	**0.297**	**0.416**	0.310	0.422	0.359	0.484	1.792	1.093	0.456	0.554	0.585	0.644	0.728	0.730	1.987	2.468
	720	**0.359**	**0.466**	0.361	0.471	0.388	0.499	2.087	1.534	0.866	0.809	0.499	0.596	1.062	0.943	3.859	1.144
ECL	48	0.351	0.457	0.318	0.438	0.380	0.489	0.971	0.884	0.493	0.539	**0.204**	**0.357**	0.879	0.764	0.524	0.595
	168	0.422	0.496	0.430	0.514	0.424	0.519	1.671	1.587	0.723	0.655	**0.315**	**0.436**	1.032	0.833	2.725	1.273
	336	0.479	0.528	0.501	0.552	0.514	0.563	3.528	2.196	1.212	0.898	**0.414**	**0.519**	1.136	0.876	2.246	3.077
	720	**0.500**	**0.547**	0.520	0.571	0.558	0.609	4.891	4.047	1.511	0.966	0.513	0.565	1.251	0.933	4.243	1.415
	960	**0.532**	**0.608**	0.584	0.638	0.624	0.645	7.019	5.105	1.545	1.006	0.657	0.683	1.370	0.982	6.901	4.264
PM25	24	0.667	**0.525**	0.691	0.572	0.730	0.592	**0.543**	0.785	0.675	0.862	0.632	0.910	0.679	0.735	0.868	1.408
	48	0.906	**0.637**	0.964	0.663	0.918	0.700	0.716	1.305	0.923	1.157	**0.887**	1.221	1.091	0.974	1.038	1.996
	168	1.209	**0.781**	1.259	0.830	1.248	0.823	1.469	4.508	**1.011**	1.839	1.258	1.448	1.707	1.325	2.179	3.206
	336	**1.215**	**0.776**	1.272	0.787	1.273	0.788	1.873	5.002	1.908	1.853	1.521	1.553	1.714	1.229	4.311	4.226
	720	**1.262**	**0.802**	1.287	0.875	1.285	0.862	2.322	6.558	1.926	1.937	1.941	1.683	1.693	1.139	7.960	7.290
Solar	36	0.208	**0.286**	**0.194**	0.297	0.209	0.298	3.737	7.528	0.600	0.748	0.274	0.296	0.552	0.555	0.446	0.382
	72	0.255	**0.306**	**0.215**	0.333	0.286	0.348	5.006	8.442	0.950	1.666	0.324	0.362	0.649	0.685	0.445	0.388
	144	**0.281**	0.339	0.308	**0.316**	0.315	0.359	5.679	8.953	0.687	1.096	0.338	0.385	0.688	0.748	0.413	0.347
	288	**0.297**	**0.348**	0.321	0.378	0.317	0.379	5.725	9.052	0.854	1.420	0.371	0.448	0.737	0.828	0.441	0.387
	720	**0.332**	**0.359**	0.357	0.414	0.395	0.392	6.876	9.691	0.853	1.595	0.385	0.431	0.992	0.875	0.499	0.402
Traffic	24	**0.167**	**0.256**	0.174	0.275	0.171	0.281	2.216	1.784	0.397	0.441	0.239	0.304	0.280	0.366	0.182	0.318
	48	**0.181**	**0.275**	0.182	0.288	0.186	0.292	4.454	1.759	0.721	0.656	0.265	0.323	0.234	0.359	0.214	0.335
	168	**0.198**	**0.282**	0.241	0.354	0.304	0.390	3.502	1.548	0.461	0.479	0.291	0.348	0.290	0.391	0.365	0.427
	336	0.287	0.356	**0.246**	**0.331**	0.321	0.403	2.438	1.286	3.711	1.598	0.339	0.389	0.317	0.417	1.252	0.684
	720	**0.296**	**0.388**	0.306	0.394	0.314	0.395	3.792	1.622	2.694	1.320	0.469	0.475	0.410	0.491	5.932	1.233
Exchange-Rate	24	0.029	0.125	0.036	0.156	0.194	0.361	0.112	0.250	0.504	0.557	2.527	1.266	**0.027**	**0.123**	0.143	0.292
	48	**0.041**	**0.147**	0.067	0.202	0.218	0.386	1.007	0.995	0.928	0.771	3.071	1.445	0.042	0.151	0.152	0.312
	168	**0.146**	**0.227**	0.163	0.259	0.631	0.690	12.852	3.579	0.922	0.786	4.775	1.803	0.178	0.312	0.227	0.384
	336	**0.368**	**0.302**	0.377	0.309	1.351	1.081	16.162	4.015	1.845	1.176	7.391	2.388	0.397	0.491	0.301	0.428
	720	**0.714**	**0.671**	0.719	0.677	2.479	1.388	11.239	3.297	1.141	0.897	11.019	3.194	0.948	0.809	0.742	0.688
Count		68		19		0		1		2		7		2		2	

5.3.1. *Univariate time-series forecasting*

Under this setting, each method attains predictions as a single variable over time. From Table 2, we observe that: **(1)** The proposed model Informer significantly improves the inference performance (wining-counts in the last column) across all datasets, and their predict error rises smoothly and slowly within the growing prediction horizon, which demonstrates the success of Informer in expanding the prediction capacity in the LSTF problem. **(2)** The Informer beats its canonical degradation Informer† mostly in wining-counts, i.e., 52>17, which supports the query sparsity assumption in providing a comparable attention feature map. Our proposed method also out-performs the most related work LogTrans and Reformer. We note that the Reformer keeps dynamic decoding and performs poorly in LSTF, while other methods benefit from the generative style decoder as nonautoregressive predictors. **(3)** The Informer model shows significantly better results than recurrent neural networks LSTMa. Our method has a MSE decrease of 40.9% (at 168), 51.4% (at 336) and 58.0% (at 720). This reveals a shorter network path in the self-attention mechanism acquires better prediction capacity than the RNN-based

Table 3
Multivariate long sequence time-series forecasting results on ten datasets.

Methods		Informer		Informer†		LogTrans		Reformer		LSTMa		LSTnet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh$_1$	24	**0.577**	**0.629**	0.620	0.647	0.736	0.696	0.991	0.754	0.650	0.624	1.293	0.901
	48	**0.645**	**0.665**	0.682	0.671	0.766	0.727	1.313	0.906	0.702	0.675	1.456	0.960
	168	**0.980**	**0.812**	0.997	0.827	1.002	0.896	1.824	1.138	1.212	0.867	1.997	1.214
	336	1.028	0.891	**0.994**	**0.863**	1.082	0.932	2.117	1.280	1.324	0.974	2.655	1.369
	720	**1.135**	**0.936**	1.141	0.937	1.297	1.091	2.415	1.520	1.960	1.322	2.143	1.380
ETTh$_2$	24	**0.520**	**0.623**	0.753	0.727	0.828	0.750	1.531	1.613	1.143	0.813	2.742	1.457
	48	**0.998**	**0.831**	1.051	0.877	1.806	1.034	1.871	1.735	1.449	0.921	3.567	1.687
	168	1.604	**1.122**	1.965	1.212	4.070	1.681	4.660	1.846	4.117	1.674	**1.542**	2.513
	336	1.823	1.177	1.520	**1.085**	3.875	1.763	4.028	1.688	3.434	1.549	**1.544**	2.591
	720	**2.484**	**1.371**	2.647	1.405	3.013	1.552	5.381	2.015	3.963	1.788	2.625	3.709
ETTh$_3$	24	**0.172**	**0.289**	0.173	0.313	0.203	0.290	0.645	0.754	0.592	0.737	0.775	1.165
	48	**0.347**	**0.417**	0.372	0.453	0.373	0.441	0.780	1.055	0.741	1.124	0.845	1.396
	168	**1.022**	**0.739**	1.048	0.779	1.083	0.762	1.362	2.854	1.149	1.956	0.946	1.501
	336	**1.232**	**0.811**	1.269	0.827	1.295	0.865	1.380	3.160	1.355	2.489	1.280	1.898
	720	**1.387**	**0.854**	1.417	0.864	1.430	0.905	1.592	3.743	1.688	3.676	1.446	1.874
ETTh$_4$	24	0.368	0.476	**0.364**	**0.469**	0.378	0.498	0.531	0.490	0.703	0.798	1.294	2.941
	48	0.632	0.627	**0.627**	**0.624**	0.695	0.647	0.782	1.018	0.869	1.163	1.359	3.121
	168	**0.733**	**0.704**	0.739	0.743	0.801	0.732	1.235	2.230	0.994	1.499	1.437	3.395
	336	**1.372**	**0.940**	1.414	0.967	1.390	0.948	1.672	4.256	1.606	4.325	1.507	3.764
	720	**1.612**	**1.021**	1.644	1.051	1.623	1.090	2.031	5.242	1.822	5.131	1.765	4.553
Weather	24	**0.385**	**0.441**	0.399	0.447	0.435	0.477	0.655	0.583	0.546	0.570	0.615	0.545
	48	0.520	0.489	**0.493**	**0.488**	0.526	0.525	0.729	0.666	0.829	0.677	0.660	0.589
	168	**0.672**	**0.595**	0.683	0.632	0.727	0.671	1.318	0.855	1.038	0.835	0.748	0.647
	336	**0.702**	**0.620**	0.707	0.634	0.754	0.670	1.930	1.167	1.657	1.059	0.782	0.683
	720	**0.831**	**0.731**	0.834	0.741	0.885	0.773	2.726	1.575	1.536	1.109	0.851	0.757
ECL	48	0.325	**0.393**	**0.309**	0.411	0.335	0.418	1.404	0.999	0.486	0.572	0.319	0.405
	168	0.340	0.434	**0.323**	**0.420**	0.348	0.456	1.515	1.069	0.574	0.602	0.374	0.426
	336	0.359	**0.441**	0.371	0.449	**0.353**	0.449	1.601	1.104	0.886	0.795	0.399	0.457
	720	0.406	0.483	**0.390**	**0.460**	0.405	0.489	2.009	1.170	1.676	1.095	0.536	0.535
	960	**0.460**	**0.548**	0.492	0.550	0.477	0.589	2.141	1.387	1.591	1.128	0.605	0.599
PM25	24	**0.636**	**0.403**	0.663	0.432	0.713	0.446	0.490	1.012	0.699	1.293	0.681	0.909
	48	**0.788**	**0.475**	0.789	0.476	0.844	0.515	0.650	1.485	0.902	1.847	0.794	0.947
	168	**0.913**	**0.555**	0.913	0.610	0.968	0.572	2.444	3.543	0.963	2.007	0.997	0.950
	336	0.936	0.565	**0.934**	**0.557**	1.021	0.619	2.647	4.460	1.032	2.028	1.192	0.935
	720	**0.941**	**0.569**	0.959	0.593	1.010	0.585	2.892	5.007	1.768	4.318	1.311	0.979
Solar	36	**0.167**	**0.241**	0.188	0.275	0.234	0.272	1.716	5.019	0.425	0.608	0.824	0.806
	72	0.212	**0.257**	**0.208**	0.262	0.273	0.290	1.962	5.876	0.643	1.207	0.814	0.764
	144	**0.221**	**0.268**	0.253	0.297	0.259	0.295	2.112	6.765	0.752	1.376	0.792	0.733
	288	**0.229**	**0.277**	0.293	0.324	0.240	0.356	2.361	6.883	1.053	2.128	0.819	0.792
	720	**0.248**	**0.275**	0.270	0.281	0.333	0.322	2.768	7.202	1.306	2.425	0.811	0.771
Traffic	24	0.344	0.326	**0.322**	**0.320**	0.386	0.328	1.640	0.961	0.828	0.438	0.717	0.590
	48	0.395	0.350	**0.339**	0.325	0.381	**0.324**	1.305	0.830	1.001	0.520	0.739	0.604
	168	0.380	0.348	**0.365**	**0.331**	0.418	0.347	2.642	1.323	1.678	0.810	0.724	0.592
	336	0.443	0.399	0.447	0.352	**0.428**	**0.347**	1.884	1.027	2.400	1.074	0.730	0.595
	720	0.454	0.406	0.485	0.368	**0.450**	**0.361**	1.960	1.042	1.997	0.926	0.805	0.640
Exchange-Rate	24	0.221	0.346	0.260	0.389	**0.190**	**0.345**	1.001	0.890	1.806	1.082	8.617	2.205
	48	**0.219**	**0.320**	0.312	0.405	0.280	0.428	2.046	1.220	2.244	1.197	8.579	2.182
	168	0.766	0.977	**0.727**	0.984	0.820	**0.753**	2.264	1.300	2.371	1.260	8.903	2.253
	336	0.889	1.015	**0.853**	**1.011**	1.076	0.879	3.104	1.516	2.508	1.297	9.883	2.425
	720	**0.903**	**0.837**	1.011	0.953	1.166	0.894	3.739	1.721	7.325	2.280	9.495	2.429
Count		64		25		9		0		0		2	

models. **(4)** The proposed method outperforms DeepAR, ARIMA and Prophet on MSE by decreasing 35.5% (at 168), 39.3% (at 336), and 40.9% (at 720) in average. The statistical models are designed for capturing long-range trending in time-series, and they show better performance than RNN-based models on longer sequences. Combined with the previous finding 3, it demonstrates that the key factor affecting the prediction capacity in LSTF is the ability of capturing long-range dependency, which is the starting point in designing our model Informer. On the ECL dataset, DeepAR performs better on shorter horizons (≤ 336), and our method surpasses on longer horizons. We attribute this to a specific example, in which the effectiveness of prediction capacity is reflected with the problem scalability.

5.3.2. Multivariate time-series forecasting

We change the univariate setting to a multivariate one, and some univariate methods are inappropriate. Instead, we add the LSTnet as the state-of-art baseline. On the contrary, our proposed Informer is easy to change from univariate

Table 4
Univariate long sequence time-series forecasting results on three datasets (fine-grained).

Methods		Informer		Informer†		LogTrans		Reformer		LSTMa		DeepAR		ARIMA		Prophet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTm_1$	24	**0.065**	**0.187**	0.074	0.190	0.075	0.232	0.095	0.228	0.121	0.233	0.091	0.243	0.090	0.206	0.120	0.290
	48	0.106	0.249	**0.103**	**0.240**	0.178	0.360	0.249	0.390	0.305	0.411	0.219	0.362	0.179	0.306	0.133	0.305
	96	0.149	**0.289**	**0.141**	0.294	0.259	0.443	0.920	0.767	0.287	0.420	0.364	0.496	0.272	0.399	0.174	0.346
	288	**0.211**	0.372	0.214	**0.369**	0.400	0.572	1.108	1.245	0.524	0.584	0.948	0.795	0.462	0.558	0.452	0.534
	672	**0.266**	**0.405**	0.269	0.435	0.528	0.662	1.793	1.528	1.064	0.873	2.437	1.352	0.639	0.697	2.747	1.174
$ETTm_3$	24	**0.031**	0.103	0.033	0.104	0.068	0.143	0.236	0.118	0.163	**0.068**	0.440	0.339	0.132	0.144	0.264	0.143
	48	0.054	0.160	**0.052**	0.162	0.062	0.187	0.425	0.374	0.310	0.210	0.797	0.892	0.213	0.188	0.271	**0.137**
	96	**0.092**	**0.223**	0.105	0.224	0.137	0.251	0.690	0.872	0.434	0.346	0.915	1.197	0.242	0.204	0.299	0.159
	288	**0.162**	**0.302**	**0.162**	0.310	0.167	0.314	0.914	1.021	1.040	1.813	1.431	2.379	0.270	0.323	0.472	0.423
	672	**0.239**	**0.390**	0.277	0.409	0.260	0.397	1.056	1.427	1.050	2.105	1.593	2.788	0.325	0.470	1.145	2.860
$ETTm_4$	24	0.161	0.247	**0.160**	**0.245**	0.187	0.291	0.289	0.205	0.254	0.288	0.430	0.357	0.164	0.279	0.343	0.260
	48	0.258	0.351	0.258	0.350	**0.257**	0.395	0.548	0.615	0.456	0.451	0.615	0.595	0.358	0.435	0.338	**0.234**
	96	**0.302**	0.403	0.319	0.411	0.303	0.408	0.922	1.778	0.791	1.080	0.750	0.803	0.483	0.550	0.359	**0.248**
	288	0.531	0.579	0.559	0.572	0.542	0.608	1.210	1.875	1.140	1.854	0.924	1.354	0.611	0.805	**0.501**	**0.435**
	672	**0.635**	**0.704**	0.643	0.728	0.655	0.734	1.315	1.994	1.297	2.787	1.081	1.635	0.691	0.822	0.980	1.914
Count		16		8		1		0		1		0		0		5	

Table 5
Multivariate long sequence time-series forecasting results on three datasets (fine-grained).

Methods		Informer		Informer†		LogTrans		Reformer		LSTMa		LSTnet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTm_1$	24	0.403	**0.524**	**0.396**	0.550	0.439	0.583	0.724	0.607	0.621	0.629	1.968	1.170
	48	0.534	0.613	**0.524**	**0.590**	0.577	0.627	1.098	0.777	1.392	0.939	1.999	1.215
	96	**0.748**	0.754	0.767	**0.712**	0.768	0.792	1.433	0.945	1.339	0.913	2.762	1.542
	288	1.359	**1.009**	1.462	1.079	1.862	1.820	1.820	1.094	1.740	1.124	**1.357**	2.076
	672	**1.809**	**1.158**	1.831	1.203	2.069	1.961	2.187	1.232	2.736	1.555	1.917	2.941
$ETTm_3$	24	0.134	**0.246**	**0.131**	0.259	0.152	0.263	0.406	0.340	0.520	0.671	0.727	1.018
	48	**0.158**	**0.264**	0.183	0.274	0.172	0.295	0.572	0.615	0.793	1.400	0.801	1.159
	96	**0.188**	**0.304**	0.231	0.352	0.205	0.342	0.706	0.937	1.005	1.933	0.728	1.053
	288	0.578	0.533	**0.575**	**0.528**	0.581	0.556	0.912	1.105	1.281	2.954	0.839	1.292
	672	**0.919**	**0.686**	0.942	0.744	0.973	0.700	1.271	1.954	1.462	3.118	1.090	1.397
$ETTm_4$	24	0.171	0.303	0.176	0.339	**0.170**	**0.302**	0.385	0.281	0.589	0.688	1.329	2.996
	48	0.248	0.370	0.273	0.404	**0.247**	**0.364**	0.482	0.423	0.718	0.907	1.373	3.091
	96	**0.417**	**0.500**	0.450	0.519	0.420	0.501	0.771	1.207	0.868	1.196	1.285	2.923
	288	**0.929**	**0.755**	0.938	0.776	0.960	0.809	1.035	1.541	1.192	2.951	1.456	3.555
	672	**0.981**	**0.778**	0.991	0.784	1.029	0.811	1.589	1.906	1.265	2.642	1.405	3.302
Count		18		7		4		0		0		1	

prediction to multivariate one by adjusting the final FCN layer. From Table 3, we observe that: **(1)** The proposed model Informer greatly outperforms other methods and the findings 1 & 2 in the univariate settings still hold for the multivariate time-series. It demonstrates our models are not limited on specific cases. **(2)** The Informer model shows better results than RNN-based LSTMa and CNN-based LSTnet, and the MSE decreases 19.7% (at 168), 19.2% (at 336), 22.8% (at 720) in average. Compared with the univariate results, the overwhelming performance is reduced, and such phenomena can be caused by the anisotropy of feature dimensions' prediction capacity. It reminds us of the Informer's potential application on the multivariate LSTF problem.

5.3.3. LSTF with granularity consideration

We perform an additional comparison to explore the performance with various time granularities, including hour-level and minute-level. Recall that the coarse-grained dataset $ETTh_1$, $ETTh_2$, $ETTh_3$, $ETTh_4$ are separated for 1-hour-level and the fine-grained datasets $ETTm_1$, $ETTm_3$, $ETTm_4$ are separated for 15-minute-level. The fine-grained univariate LSTF results and the multivariate LSTF results on $ETTm_{\{1,3,4\}}$ are summarized in Table 4 and Table 5, respectively. Since the $ETTm_2$ has serious data imperfection, we disregard it in this experiment. The {96, 288, 672} of $ETTm_{\{1,3,4\}}$ is the aligned sequences with respect to {24, 48, 168} of $ETTh_{\{1,3,4\}}$, and the former is collected at minutes-level and the latter at hour-level. For univariate forecasting, the proposed Informer defeats the previous strongest baseline LSTMa by decreasing the MSE 62.9% (at 96), 65.9% (at 288), 67.8% (at 672) in average. For multivariate forecasting, the Informer also outperforms the LSTMa by decreasing the MSE 59.1% (at 96), 32.9% (at 288), 31.1% (at 672) in average. Moreover, the Informer is the strongest Transformer-based model and achieves best performance on the longest horizon (=672) of all datasets. It demonstrates that Informer can be effective for predicting long sequence time-series even if the sequences are at different granularity levels.

Table 6
Univariate long sequence time-series forecasting results on four datasets (short-term).

Methods		Informer		Informer$^+$		LSTMa		LSTnet		Transformer		TFTransformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh$_1$	1	0.010	0.074	0.009	0.067	**0.005**	**0.050**	0.045	0.198	0.013	0.211	0.011	0.201
	6	0.059	0.201	0.020	0.109	**0.015**	**0.086**	0.053	0.234	0.025	0.118	0.023	0.109
	12	0.068	0.213	**0.045**	**0.169**	0.115	0.279	0.357	1.325	0.052	0.182	0.047	0.171
ETTh$_2$	1	0.007	0.062	0.004	0.048	**0.003**	**0.037**	0.041	0.412	0.005	0.051	0.004	0.049
	6	0.031	0.128	**0.027**	**0.119**	0.062	0.171	0.123	0.828	0.033	0.132	0.028	0.121
	12	0.062	0.188	**0.056**	**0.175**	0.119	0.235	0.367	1.638	0.064	0.192	0.060	0.184
Weather	1	**0.016**	0.077	**0.016**	**0.074**	0.017	0.077	0.058	0.638	0.018	0.079	**0.016**	0.075
	6	**0.035**	0.126	**0.035**	**0.124**	0.057	0.161	0.093	0.781	0.039	0.135	0.037	0.129
	12	0.060	0.171	**0.056**	**0.165**	0.093	0.207	0.177	1.372	0.063	0.181	0.058	0.168
ECL	1	0.051	0.152	0.050	0.148	0.072	0.187	0.123	0.862	0.053	0.155	**0.049**	**0.146**
	6	**0.122**	0.256	**0.122**	**0.255**	0.273	0.386	0.414	0.973	0.130	0.264	0.124	0.259
	12	0.143	0.280	**0.142**	**0.273**	0.425	0.488	1.223	1.582	0.151	0.207	0.146	0.291
Solar	1	0.017	0.060	0.016	0.058	**0.014**	**0.047**	0.042	0.362	0.020	0.065	0.015	0.051
	6	0.060	0.136	0.057	0.125	0.056	0.103	0.138	0.637	0.065	0.147	**0.055**	**0.101**
	12	**0.091**	0.168	0.101	0.176	0.118	**0.143**	0.352	1.128	0.112	0.140	0.095	0.157
PM25	1	0.050	0.122	0.051	0.121	0.048	0.117	0.056	0.622	0.054	0.124	**0.047**	**0.115**
	6	**0.250**	0.287	0.254	0.286	0.255	**0.279**	0.478	0.989	0.271	0.296	0.253	0.289
	12	**0.419**	**0.391**	0.439	0.401	0.526	0.414	1.151	1.532	0.445	0.412	0.427	0.396
Count		7		16		10		0		0		7	

Table 7
Multivariate long sequence time-series forecasting results on four datasets (short-term).

Methods		Informer		Informer$^+$		LSTMa		LSTnet		Transformer		TFTransformer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh$_1$	1	0.136	0.251	**0.118**	**0.232**	0.165	0.282	0.779	0.549	0.131	0.247	0.123	0.235
	6	0.379	**0.421**	**0.370**	0.428	0.639	0.511	0.920	0.658	0.383	0.433	0.375	0.431
	12	0.494	0.511	**0.480**	**0.487**	0.715	0.556	0.965	0.692	0.497	0.515	0.491	0.508
ETTh$_2$	1	0.270	0.416	**0.174**	**0.307**	0.244	0.366	3.127	1.345	0.197	0.354	0.186	0.348
	6	0.722	0.708	0.647	0.648	**0.507**	**0.524**	3.141	1.360	0.679	0.673	0.637	0.636
	12	1.221	0.916	1.014	0.827	**0.662**	**0.607**	3.147	1.361	1.332	0.929	0.995	0.815
Weather	1	0.161	0.202	0.159	**0.197**	0.156	0.201	0.495	0.458	0.164	0.206	**0.154**	**0.197**
	6	**0.226**	**0.277**	0.231	0.284	0.264	0.306	0.537	0.500	0.235	0.287	0.230	0.283
	12	**0.270**	**0.326**	0.272	0.327	0.324	0.357	0.555	0.519	0.275	0.331	0.274	0.329
ECL	1	0.183	0.308	**0.159**	**0.200**	0.246	0.338	0.631	0.579	0.169	0.237	0.171	0.251
	6	0.222	0.341	**0.203**	**0.321**	0.285	0.358	0.657	0.604	0.233	0.355	0.212	0.335
	12	0.240	0.348	**0.222**	**0.335**	0.305	0.371	0.663	0.608	0.251	0.356	0.228	0.338
Solar	1	0.017	0.061	0.017	0.062	0.014	0.044	0.577	0.697	0.021	0.066	**0.013**	**0.042**
	6	0.060	0.135	0.061	0.137	0.057	0.127	0.062	0.724	0.071	0.144	**0.054**	**0.122**
	12	**0.087**	0.161	0.092	0.167	0.120	**0.159**	0.642	0.752	0.101	0.177	0.091	0.165
PM25	1	**0.049**	0.120	0.050	0.121	**0.049**	**0.119**	0.711	0.584	0.052	0.126	0.050	0.121
	6	0.252	0.287	**0.244**	**0.285**	0.271	0.297	0.800	0.638	0.264	0.291	0.251	0.286
	12	**0.433**	0.400	0.437	0.402	0.445	**0.396**	0.834	0.653	0.445	0.409	0.436	0.403
Count		8		16		8		0		0		6	

5.3.4. Short-term forecasting

We also show our Informer's performance under the short-term forecasting setting (horizon$\leq$24) and perform extra experiments on the more complicated multivariate case. Meanwhile, for a complete comparison, we introduce two new baseline models in this experiment: the temporal fusion transformer model [83] and the vanilla transformer model [14]. From Table 6, we can find that on univariate short-term forecasting tasks, our Informer achieves higher performance than the baseline methods (25 wins in total). From Table 7, we can also find that our proposed model Informer achieves better results on short-term multivariate cases (24 wins in total), which is more evident in data ETTh$_1$, Weather, and ECL. Nevertheless, the LSTMa becomes a strong baseline for extreme cases like a next-step prediction. The LSTMa matches the near predictions' requirement, and the Informer is fit for the forecasting in the long run. For transformer-based models, we can see that the vanilla transformer model achieves higher performance than the LSTMa and LSTnet, meaning self-attention can bring more information to the final decision layer to better forecast. However, the vanilla transformer performs better than the informer models and the temporal fusion transformer, which indicates that the recurrent prediction mechanism produces more accumulated errors than the generative prediction mechanism. These results demonstrate that the Informer has the potential scalability in forecasting tasks.

5.3.5. Case study

We perform an inference process on each of the comparison methods on ETTm$_1$ to visualize the forecasting results. Fig. 10 presents a slice of the predicts of 8 models. The most related works, LogTrans and Reformer, show acceptable

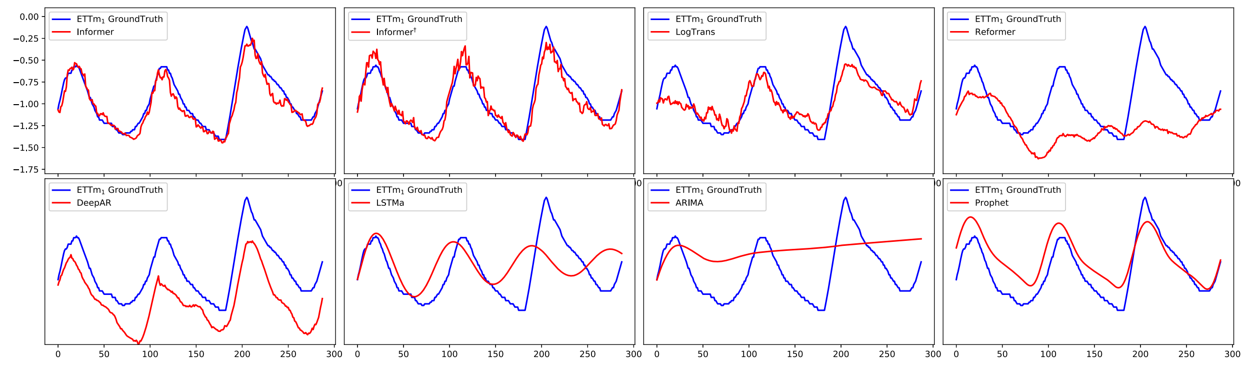


Fig. 10. The predicts (len=336) of Informer, Informer†, LogTrans, Reformer, DeepAR, LSTMa, ARIMA and Prophet on the ETTm dataset. The red / blue curves stand for slices of the prediction / ground truth.

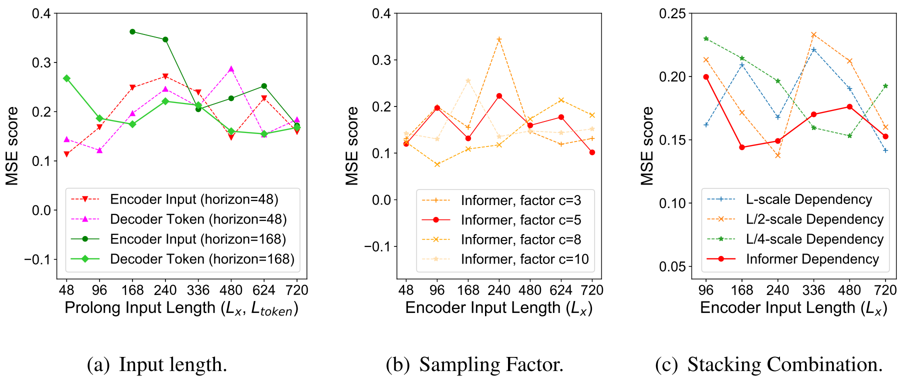


Fig. 11. The parameter sensitivity of three components in Informer.

results. The LSTMa model is not amenable to the long sequence prediction task. The ARIMA and DeepAR can capture the long trend of the long sequences. And the Prophet detects the changing point and fits it with a smooth curve better than the ARIMA and DeepAR. Our proposed model Informerand Informer† show significantly better results than the above methods.

5.4. Parameter sensitivity

We perform the sensitivity analysis of the proposed Informer model. All the experiments are conducted on ETTh1 under the univariate setting.

5.4.1. Input length

Fig. 11(a) revels how the MSE score varying as the length of inputs increasing. When predicting short sequences (like 48), initially increasing input length of encoder/decoder degrades performance, but further increasing causes the MSE to drop because it brings repeat short-term patterns. However, the MSE gets lower with longer inputs in predicting long sequences (like 168). That is because, the longer encoder input may contain more dependencies, and the longer decoder token has rich local information.

5.4.2. Sampling factor

The sampling factor controls the information bandwidth of *ProbSparse* self-attention in Eq. (9). A proper sampling factor helps matching the "sparsity" self-attention assumption. We start selecting from the small factor (=3) to large ones, and the general performance increases a little and stabilizes at last in Fig. 11(b). The selecting results verify our query sparsity assumption that there are redundant dot-product pairs in the self-attention mechanism. We set the sample factor $c = 5$ (the red line) as the default recommendation.

5.4.3. The combination of layer stacking

The replica of Layers is complementary for the self-attention distilling, and we investigate the individual stack {L, L/2, L/4}'s behavior in Fig. 11(c). With the encoder's input length grows, stacks with different scale show similar "down-up-down" trends. The longer stack is more sensitive to the inputs, partly due to receiving more long-term information. Our empirical selection (the red line), i.e., joining L and L/4, is the most robust strategy.

Table 8
Ablation study of the *ProbSparse* self-attention mechanism.

Prediction length		336			720		
Encoder's input		336	720	1440	720	1440	2880
Informer	MSE	0.259	0.245	0.236	0.271	0.261	0.257
	MAE	0.423	0.416	0.401	0.435	0.431	0.422
Informer†	MSE	0.241	0.224	-	0.259	-	-
	MAE	0.403	0.394	-	0.423	-	-
LogTrans	MSE	0.263	0.231	-	0.273	-	-
	MAE	0.478	0.431	-	0.423	-	-
Reformer	MSE	1.875	1.865	1.861	2.243	2.174	2.113
	MAE	1.144	1.129	1.125	1.536	1.497	1.434

[1] Informer† uses the canonical self-attention mechanism.
[2] The '-' indicates failure for the out-of-memory.

Table 9
Ablation study of the self-attention distilling.

Prediction length		336					480				
Encoder's input		336	480	720	960	1200	336	480	720	960	1200
Informer†	MSE	0.241	0.208	0.224	0.199	0.186	0.197	0.243	0.213	0.192	0.174
	MAE	0.403	0.385	0.394	0.381	0.369	0.388	0.392	0.383	0.377	0.362
Informer‡	MSE	0.229	0.215	0.204	-	-	0.224	0.208	0.197	-	-
	MAE	0.398	0.387	0.377	-	-	0.381	0.376	0.370	-	-

[1] Informer‡ removes the self-attention distilling from Informer†.
[2] The '-' indicates failure for the out-of-memory.

5.5. *Ablation study: how well Informer works?*

We also conducted additional experiments on ETTh$_1$ to confirm the effectiveness of the proposed components in Informer accordingly.

5.5.1. *The performance of* ProbSparse *self-attention mechanism*

In the overall results Table 2 & 3, we limited the problem setting to make the memory usage feasible for the canonical self-attention. In this study, we compare our methods with LogTrans and Reformer, and thoroughly explore their extreme performance. To isolate the memory efficient problem, we first reduce settings as {batch size=8, heads=8, dim=64}, and maintain other setups in the univariate case. We replace the self-attention mechanism with LogTrans and Reformer and perform a comparison accordingly. In Table 8, the *ProbSparse* self-attention shows better performance than the counterparts. The LogTrans gets OOM in extreme cases because its public implementation is the mask of the full-attention, which still has $\mathcal{O}(L^2)$ memory usage. Our proposed *ProbSparse* self-attention avoids this from the simplicity brought by the query sparsity assumption in Eq. (13), referring to the pseudo-code in Algorithm 1, and reaches smaller memory usage. Another interesting finding is that a longer encoder's input length helps Informer get better performance. When predicting horizon equals 336, the Informer can not exceed others with longer inputs. However, the situation reversed for a longer horizon ($= 720$). Thus, the longer LSTF problem, the larger prediction capacity is required for processing longer inputs.

5.5.2. *The performance of self-attention distilling*

In this study, we use Informer† as the benchmark to eliminate additional effects of *ProbSparse* self-attention. The other experimental setup is aligned with the settings of univariate Time-series. From Table 9, Informer† has fulfilled all the experiments and achieves better performance after taking advantage of long sequence inputs ($= 1200$). The comparison method Informer‡ removes the distilling operation and reaches OOM with longer inputs (> 720). Regarding the benefits of long sequence inputs in the LSTF problem, we conclude that the self-attention distilling is worth adopting, especially when a longer prediction is required.

5.5.3. *The performance of generative style decoder*

In this study, we testify the potential value of our decoder in acquiring a "generative" results. Unlike the existing methods, the labels and outputs are forced to be aligned in the training and inference, our proposed decoder's predicting relies solely on the time stamp, which can predict with offsets. We do not retrain the model and only add offsets to the target's original time stamps during the inference stage. From Table 10, we can see that the general prediction performance of Informer‡ resists with the offset increasing, while the counterpart fails for the dynamic decoding. It proves the decoder's ability to capture individual long-range dependency between arbitrary outputs and avoid error accumulation.

Table 10
Ablation study of the generative style decoder.

Prediction length		336					480				
Prediction offset		+0	+12	+24	+48	+72	+0	+48	+96	+144	+168
Informer‡	MSE	0.207	0.209	0.211	0.211	0.216	0.198	0.203	0.203	0.208	0.208
	MAE	0.385	0.387	0.391	0.393	0.397	0.390	0.392	0.393	0.401	0.403
Informer§	MSE	0.201	-	-	-	-	0.392	-	-	-	-
	MAE	0.393	-	-	-	-	0.484	-	-	-	-

[1] Informer§ replaces our decoder with dynamic decoding one in Informer‡.
[2] The '-' indicates failure for the unacceptable metric results.

Table 11
Ablation study of the stamp embedding.

Method		ETTh$_1$ (Univariate)					ETTh$_1$ (Multivariate)				
Prediction length		24	48	168	336	720	24	48	168	336	720
Informer	MSE	0.072	0.122	0.172	0.242	0.269	0.577	0.645	0.980	1.028	1.135
	MAE	0.206	0.273	0.330	0.417	0.435	0.629	0.665	0.812	0.891	0.936
Informer-SE	MSE	0.080	0.193	0.268	0.286	0.274	0.711	0.999	1.119	1.204	1.401
	MAE	0.227	0.372	0.449	0.467	0.453	0.618	0.772	0.856	0.891	0.975
Informer†	MSE	0.058	0.141	0.207	0.225	0.257	0.620	0.682	0.997	0.994	1.141
	MAE	0.186	0.302	0.375	0.398	0.421	0.647	0.671	0.827	0.863	0.937
Informer†-SE	MSE	0.080	0.192	0.256	0.198	0.428	0.516	0.649	1.094	1.162	1.267
	MAE	0.223	0.367	0.435	0.377	0.587	0.516	0.612	0.857	0.865	0.939

[1] '-SE' removes the stamp embedding of Informer.

Table 12
Ablation study of the mixed multi-head attention.

Method		ETTh$_1$ (Univariate)					ETTh$_1$ (Multivariate)				
Prediction length		24	48	168	336	720	24	48	168	336	720
Informer	MSE	0.072	0.122	0.172	0.242	0.269	0.577	0.645	0.980	1.028	1.135
	MAE	0.206	0.273	0.330	0.417	0.435	0.629	0.665	0.812	0.891	0.936
Informer-mix	MSE	0.083	0.132	0.195	0.173	0.074	0.589	0.946	1.294	1.358	1.357
	MAE	0.223	0.287	0.365	0.340	0.213	0.552	0.719	0.946	0.972	0.977
Informer†	MSE	0.058	0.141	0.207	0.225	0.257	0.620	0.682	0.997	0.994	1.141
	MAE	0.186	0.302	0.375	0.398	0.421	0.647	0.671	0.827	0.863	0.937
Informer†-mix	MSE	0.059	0.119	0.159	0.131	0.090	0.478	0.627	0.972	0.947	1.154
	MAE	0.192	0.273	0.327	0.288	0.239	0.498	0.599	0.781	0.766	0.881

[1] '-mix' replace the mixed multi-head attention with the original attention.

5.5.4. The performance of stamp embedding

In this study, we testify to the effectiveness of the proposed stamp embedding, which aims to bring accurate time stamp information to the inputs and outputs. From Table 11, we can see that the Informer models without stamp embedding suffer a performance decrease in this ablation study experiment. Stamp embedding provides more accurate time information than the simple positional embedding method. Meanwhile, removing the stamp embedding makes it hard to perform generative-style decoding.

5.5.5. The performance of mixed multi-head attention

In this study, we use Informer and Informer† as benchmarks to eliminate the effect of the mixed multi-head attention. From Table 12, we can see that, the mixed multi-head attention can help Informer models to gain additional performance. This demonstrated that the mixture of local features with respect to different heads can enrich the reception fields, it helps to produce more representative features before the fully connected layer.

5.6. The effectiveness of random sampling strategy

We have conducted experiments to show the performance of different sampling strategies. Then Informer applies the random sampling strategy, and the following counterparts replace it with variants. The '+full' indicates using all the pairs like canonical self-attention as the baselines. To be consistent with Section 4.1.4, the '+norm' selects the largest vector-norm of the query-key pairs, namely the max strategy, the '+avg' stands for the central strategy. We also include a non-negative matrix factorization (NMF) method as baselines '+NMF'. Table 13 and Table 14 show that the random sampling achieves best performance at most times (21/40). The random sampling beats the '+full' methods because it could simplify the KL measurement between a uniform distribution and unknown distribution with less noise. The core idea is to distinguish the dominating keys rather than fully reconstructing the KL measurement. The max strategy shows a potential solution as

Table 13
Univariate forecasting results for different sampling strategies.

Methods		Informer		+full		+norm		+avg		+NMF	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh_1	24	**0.072**	**0.206**	0.099	0.250	0.092	0.238	0.089	0.240	0.133	0.308
	48	0.122	0.273	0.124	0.285	**0.097**	**0.243**	0.162	0.327	0.120	0.288
	168	0.172	0.330	0.107	0.258	**0.100**	**0.252**	0.123	0.277	0.217	0.394
	336	0.242	0.417	**0.100**	**0.251**	0.114	0.265	0.144	0.301	0.206	0.377
	720	0.269	0.435	0.095	0.249	**0.084**	**0.220**	0.102	0.245	0.167	0.337
ETTh_2	24	0.093	0.240	0.081	0.220	**0.075**	**0.207**	0.080	0.218	0.083	0.222
	48	**0.115**	**0.270**	0.213	0.371	0.135	0.286	0.170	0.332	0.140	0.291
	168	**0.169**	**0.334**	0.267	0.414	0.238	0.394	0.297	0.444	0.231	0.388
	336	**0.205**	**0.375**	0.272	0.424	0.275	0.424	0.250	0.407	0.264	0.416
	720	**0.232**	**0.395**	0.240	0.400	0.249	0.407	0.260	0.416	0.270	0.422
Count		10		2		8		0		0	

Table 14
Multivariate forecasting results for different sampling strategies.

Methods		Informer		+full		+norm		+avg		+NMF	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh_1	24	0.577	0.629	**0.458**	**0.473**	0.517	0.532	0.505	0.515	0.517	0.515
	48	**0.645**	0.665	0.747	0.684	0.689	**0.635**	0.737	0.662	0.689	0.633
	168	0.980	0.812	1.178	0.885	1.021	0.789	**0.969**	**0.761**	1.054	0.809
	336	**1.028**	**0.891**	1.078	0.901	1.229	0.893	1.296	0.918	1.328	0.937
	720	**1.135**	**0.936**	1.429	0.967	1.410	0.993	1.300	0.946	1.372	0.962
ETTh_2	24	0.520	0.623	**0.416**	**0.499**	0.501	0.541	0.527	0.576	0.590	0.602
	48	**0.998**	**0.831**	2.295	1.231	1.795	1.115	2.029	1.142	2.185	1.224
	168	**1.604**	**1.122**	3.106	1.411	3.266	1.423	3.278	1.569	2.634	1.307
	336	**1.823**	**1.177**	3.125	1.499	2.539	1.316	3.300	1.524	2.861	1.309
	720	2.484	1.371	3.761	1.635	2.680	1.281	**2.251**	**1.253**	2.355	1.337
Count		11		4		1		4		0	

Table 15
L-related computation statics of each layer.

Methods	Training		Testing
	Time	Memory	Steps
Informer	$\mathcal{O}(L \log L)$	$\mathcal{O}(L \log L)$	1
Transformer	$\mathcal{O}(L^2)$	$\mathcal{O}(L^2)$	L
LogTrans	$\mathcal{O}(L \log L)$	$\mathcal{O}(L^2)$	$1^{\star}$
Reformer	$\mathcal{O}(L \log L)$	$\mathcal{O}(L \log L)$	L
LSTM	$\mathcal{O}(L)$	$\mathcal{O}(L)$	L

[1] The LSTnet is hard to present in a closed form.
[2] The $\star$ denotes applying our proposed decoder.

we discussed, especially for the heavily long-tailed distribution. Considering the computation cost and implementation, we recommend the random sampling strategies as the current finest solution.

5.7. Computation efficiency

With the multivariate setting and all the methods' current finest implement, we evaluate the iterative efficiency on both training and testing process. The input length is tuned from 48 to 720 in Fig. 12. During the training phase, the Informer (red line) achieves the best training efficiency among Transformer-based methods. During the testing phase, our methods are much faster than others with the generative style decoding, which is crucial for real-world deployment. The comparisons of theoretical time complexity and memory usage are summarized in Table 15. The performance of Informer is aligned with the runtime experiments. Note that the LogTrans focus on improving the self-attention mechanism, and we apply our proposed decoder in LogTrans for a fair comparison (the $\star$ in Table 15).

6. Conclusion

In this paper, we studied the long-sequence time-series forecasting problem and proposed Informer to predict long sequences. Specifically, we designed the *ProbSparse* self-attention mechanism and distilling operation to handle the challenges of quadratic time complexity and quadratic memory usage in vanilla Transformer. Also, the carefully designed generative decoder alleviates the limitation of traditional encoder-decoder architecture. The experiments on real-world data demonstrated the effectiveness of Informer for expanding the prediction capacity in the LSTF problem.

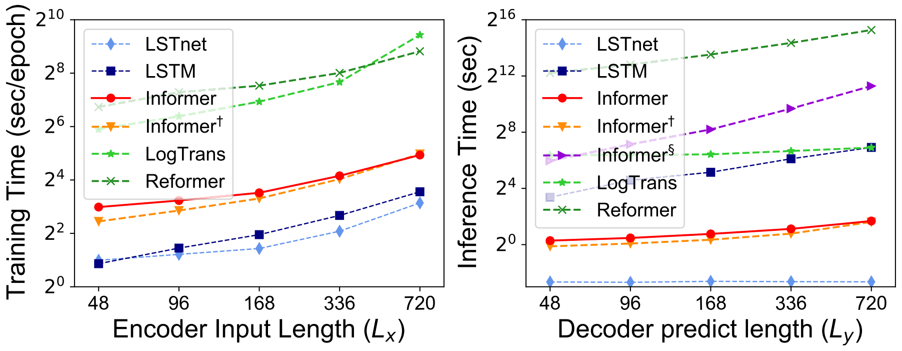


Fig. 12. The total runtime of training/testing phase.

In future work, we plan to investigate better input representations for time-series data with rich information types, such as spatial structure and semantic connections. Besides, with the in-depth applications in some critical decision-making scenarios, the input time-series data may contain outliers and malicious attacks due to the inevitable error-prone data collection. Improving the model's robustness to outliers or attacks is also an interesting direction.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Haoyi ZHOU reports financial support was provided by National Natural Science Foundation of China.

Jianxin LI reports financial support was provided by National Natural Science Foundation of China.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by grants from the Natural Science Foundation of China (62202029 and U20B2053). Thanks for the computing infrastructure provided by Beijing Advanced Innovation Center for Big Data and Brain Computing. This work was also sponsored by CAAI-Huawei MindSpore Open Fund (CAAIXSJLJJ-2022-059A). We gratefully acknowledge the support of MindSpore, CANN (Compute Architecture for Neural Networks) and Ascend AI Processor used for this research. This work is also supported in part by Foshan HKUST Projects (FSUST21-FYTRI01A, FSUST21-FYTRI02A).

References

[1] S. Papadimitriou, P. Yu, Optimal multi-scale patterns in time series streams, in: ACM SIGMOD, 2006, pp. 647–658.
[2] Y. Zhu, D.E. Shasha, Statstream: statistical monitoring of thousands of data streams in real time, in: VLDB, 2002, pp. 358–369.
[3] Y. Matsubara, Y. Sakurai, W.G. van Panhuis, C. Faloutsos, FUNNEL: automatic mining of spatially coevolving epidemics, in: ACM SIGKDD, 2014, pp. 105–114.
[4] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long- and short-term temporal patterns with deep neural networks, in: ACM SIGIR, 2018, pp. 95–104.
[5] B. Praveen, S. Talukdar, S. Mahato, J. Mondal, P. Sharma, A. Reza, M.T. Islam, A. Rahman, et al., Analyzing trend and forecasting of rainfall changes in India using non-parametrical and machine learning approaches, Sci. Rep. 10 (1) (2020) 1–21.
[6] C. Faloutsos, J. Gasthaus, T. Januschowski, Y. Wang, Forecasting big time series: old and new, Proc. VLDB Endow. 11 (12) (2018) 2102–2105.
[7] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[8] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, in: ICLR, 2018.
[9] R. Yu, S. Zheng, A. Anandkumar, Y. Yue, Long-term forecasting using tensor-train rnns, arXiv:1711.00073.
[10] Y. Liu, C. Gong, L. Yang, Y. Chen, Dstp-rnn: a dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction, Expert Syst. Appl. 143 (2020) 113082.
[11] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, G.W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: IJCAI, 2017, pp. 2627–2633.
[12] R. Wen, K. Torkkola, B. Narayanaswamy, D. Madeka, A multi-horizon quantile recurrent forecaster, in: NIPS, 2017.
[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, in: NIPS, vol. 33, 2020, pp. 1877–1901.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS, 2017, pp. 5998–6008.
[15] R. Child, S. Gray, A. Radford, I. Sutskever, Generating long sequences with sparse transformers, arXiv:1904.10509.
[16] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, in: NIPS, vol. 32, 2019.
[17] I. Beltagy, M.E. Peters, A. Cohan, Longformer: the long-document transformer, arXiv preprint, arXiv:2004.05150.
[18] J. Qiu, H. Ma, O. Levy, S.W.-t. Yih, S. Wang, J. Tang, Blockwise self-attention for long document understanding, in: Findings of EMNLP, ACL, 2020.

[19] N. Kitaev, L. Kaiser, A. Levskaya, Reformer: the efficient transformer, in: ICLR, 2019.
[20] S. Wang, B. Li, M. Khabsa, H. Fang, H. Ma Linformer, Self-attention with linear complexity, arXiv:2006.04768.
[21] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q.V. Le, R. Salakhutdinov, Transformer-xl: attentive language models beyond a fixed-length context, in: ACL, 2019.
[22] J.W. Rae, A. Potapenko, S.M. Jayakumar, T.P. Lillicrap, Compressive transformers for long-range sequence modelling, arXiv:1911.05507.
[23] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, in: AAAI, vol. 35, 2021, pp. 11106–11115.
[24] G.E. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.
[25] W.D. Ray, Time Series: Theory and Methods by P.J. Brockwell, R.A. Davis, J. R. Stat. Soc., Ser. A, Stat. Soc. 153 (3) (1990) 400.
[26] M. Seeger, S. Rangapuram, Y. Wang, D. Salinas, J. Gasthaus, T. Januschowski, V. Flunkert, Approximate Bayesian inference in linear state space models for intermittent demand forecasting at scale, arXiv:1709.07638.
[27] M.W. Seeger, D. Salinas, V. Flunkert, Bayesian intermittent demand forecasting for large inventories, in: NIPS, 2016, pp. 4646–4654.
[28] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, Deepar: probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191.
[29] S. Mukherjee, D. Shankar, A. Ghosh, N. Tathawadekar, P. Kompalli, S. Sarawagi, K. Chaudhury, Armdn: associative and recurrent mixture density networks for eretail demand forecasting, arXiv:1803.03800.
[30] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: NIPS, 2014, pp. 3104–3112.
[31] C. Aicher, N.J. Foti, E.B. Fox, Adaptively truncating backpropagation through time to control gradient bias, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 799–808.
[32] T. Trinh, A. Dai, T. Luong, Q. Le, Learning longer-term dependencies in rnns with auxiliary losses, in: ICML, PMLR, 2018, pp. 4965–4974.
[33] J.G. Zilly, R.K. Srivastava, J. Koutník, J. Schmidhuber, Recurrent highway networks, in: ICML, 2017, pp. 4189–4198.
[34] Y. Cao, P. Xu, Better long-range dependency by bootstrapping a mutual information regularizer, in: AISTATS, PMLR, 2020, pp. 3991–4001.
[35] Y. Luo, Z. Chen, T. Yoshioka, Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation, in: ICASSP, IEEE, 2020, pp. 46–50.
[36] D. Stoller, M. Tian, S. Ewert, S. Dixon, Seq-u-net: a one-dimensional causal u-net for efficient sequence modelling, arXiv:1911.06393.
[37] S. Bai, J.Z. Kolter, V. Koltun, Convolutional sequence modeling revisited, in: ICLR, 2018.
[38] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: ICLR, 2015.
[39] T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, in: EMNLP, ACL, 2015, pp. 1412–1421.
[40] W. Chan, N. Jaitly, Q. Le, O. Vinyals, Listen, attend and spell: a neural network for large vocabulary conversational speech recognition, in: ICASSP, IEEE, 2016, pp. 4960–4964.
[41] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova Bert, Pre-training of deep bidirectional transformers for language understanding, arXiv:1810.04805.
[42] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, 2018.
[43] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R.R. Salakhutdinov, Q.V. Le, Xlnet: generalized autoregressive pretraining for language understanding, in: NIPS, vol. 32, 2019.
[44] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: a robustly optimized bert pretraining approach, arXiv preprint, arXiv:1907.11692.
[45] L. Dong, S. Xu, B. Xu, Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition, in: ICASSP, IEEE, 2018, pp. 5884–5888.
[46] C.A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A.M. Dai, M.D. Hoffman, M. Dinculescu, D. Eck, Music transformer: generating music with long-term structure, in: ICLR, 2019.
[47] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, D. Tran, Image transformer, in: ICML, PMLR, 2018, pp. 4055–4064.
[48] H. Song, D. Rajan, J.J. Thiagarajan, A. Spanias, Attend and diagnose: clinical time series analysis using attention models, in: AAAI, 2018.
[49] J. Ma, Z. Shou, A. Zareian, H. Mansour, A. Vetro, S.-F. Chang, Cdsa: cross-dimensional self-attention for multivariate, geo-tagged time series imputation, arXiv:1905.09904.
[50] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches, in: EMNLP, 2014, pp. 103–111.
[51] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[52] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in: NIPS 2014 Deep Learning and Representation Learning Workshop, 2014.
[53] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, S.-D. Lin, A memory-network based solution for multivariate time-series forecasting, in: AAAI, 2019.
[54] S.J. Taylor, B. Letham, Forecasting at scale, Am. Stat. 72 (1) (2018) 37–45.
[55] Y.-H.H. Tsai, S. Bai, M. Yamada, L.-P. Morency, R. Salakhutdinov, Transformer dissection: an unified understanding for transformer's attention via the lens of kernel, in: ACL, 2019, pp. 4335–4344.
[56] P. Brémaud, An Introduction to Probabilistic Modeling, Springer Science & Business Media, 2012.
[57] B.W. Silverman, M.C. Jones, E. Fix and J.L. Hodges (1951): an important contribution to nonparametric discriminant analysis and density estimation: commentary on Fix and Hodges (1951), Int. Stat. Rev. (Rev. Int. Stat.) (1989) 233–238.
[58] F. Pérez-Cruz, Kullback-Leibler divergence estimation of continuous distributions, in: 2008 IEEE International Symposium on Information Theory, IEEE, 2008, pp. 1666–1670.
[59] Q. Wang, S.R. Kulkarni, S. Verdú, Divergence estimation for multidimensional densities via k-nearest-neighbor distances, IEEE Trans. Inf. Theory 55 (5) (2009) 2392–2405.
[60] G.C. Calafiore, S. Gaubert, C. Possieri, Log-sum-exp neural networks and posynomial models for convex and log-log-convex data, IEEE Trans. Neural Netw. Learn. Syst. 31 (3) (2019) 827–838.
[61] P. Zhao, L. Lai, Minimax optimal estimation of kl divergence for continuous distributions, IEEE Trans. Inf. Theory 66 (12) (2020) 7787–7811.
[62] P. Zhao, Nearest neighbor methods with applications in functional estimation and machine learning, Ph.D. thesis, University of California, Davis, 2021.
[63] F. Nielsen, Statistical divergences between densities of truncated exponential families with nested supports: duo Bregman and duo Jensen divergences, Entropy 24 (3) (2022) 421.
[64] F. Yu, V. Koltun, T. Funkhouser, Dilated residual networks, in: CVPR, 2017, pp. 472–480.
[65] A. Gupta, A.M. Rush, Dilated convolutions for modeling long-distance genomic dependencies, in: ICML Workshop on Computational Biology, 2017.
[66] D. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), in: ICLR, 2016.
[67] D. Dufresne, Sums of lognormals, in: ARC, 2008, pp. 1–6.
[68] J.A. Vargasguzman, Change of support of transformations: conservation of lognormality revisited, Math. Geosci. 37 (6) (2005) 551–567.
[69] S. Asmussen, P.-O. Goffard, P.J. Laub, Orthonormal polynomial expansions and lognormal sum densities, in: Risk and Stochastics: Ragnar Norberg, World Scientific, 2019, pp. 127–150.
[70] M.B. Hcine, R. Bouallegue, On the approximation of the sum of lognormals by a log skew normal distribution, arXiv preprint, arXiv:1502.03619.
[71] S. Asmussen, L. Rojas-Nandayapa, Asymptotics of sums of lognormal random variables with gaussian copula, Stat. Probab. Lett. 78 (16) (2008) 2709–2714.

[72] L.F. Fenton, The sum of log-normal probability distributions in scatter transmission systems, IRE Trans. Commun. Syst. 8 (1) (1960) 57–67.
[73] N. Marlow, A normal limit theorem for power sums of independent random variables, Bell Syst. Tech. J. 46 (9) (1967) 2081–2089.
[74] D. Dufresne, The log-normal approximation in financial and other computations, Adv. Appl. Probab. 36 (3) (2004) 747–773.
[75] J.A. Gubner, A new formula for lognormal characteristic functions, IEEE Trans. Veh. Technol. 55 (5) (2006) 1668–1671.
[76] P. Embrechts, G. Puccetti, L. Rüschendorf, R. Wang, A. Beleraj, An academic response to Basel 3.5, Risks 2 (1) (2014) 25–48.
[77] C.-F. Lo, The sum and difference of two lognormal random variables, J. Appl. Math. (2012).
[78] H.F. Trotter, On the product of semi-groups of operators, Proc. Am. Math. Soc. 10 (4) (1959) 545–551.
[79] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: frequency enhanced decomposed transformer for long-term series forecasting, in: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, S. Sabato (Eds.), International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, in: Proceedings of Machine Learning Research, vol. 162, PMLR, 2022, pp. 27268–27286, https://proceedings.mlr.press/v162/zhou22g.html.
[80] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: decomposition transformers with auto-correlation for long-term series forecasting, in: M. Ranzato, A. Beygelzimer, Y.N. Dauphin, P. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 22419–22430, https://proceedings.neurips.cc/paper/2021/hash/bcc0d400288793e8bdcd7c19a8ac0c2b-Abstract.html.
[81] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, L. Sun, Transformers in time series: a survey, CoRR abs/2202.07125, arXiv:2202.07125, https://arxiv.org/abs/2202.07125.
[82] A.A. Ariyo, A.O. Adewumi, C.K. Ayo, Stock price prediction using the arima model, in: ICCMS, IEEE, 2014, pp. 106–112.
[83] B. Lim, S.Ö. Arık, N. Loeff, T. Pfister, Temporal fusion transformers for interpretable multi-horizon time series forecasting, Int. J. Forecast. 37 (4) (2021) 1748–1764.