Infomaxformer: Maximum Entropy Transformer for Long Time-Series Forecasting Problem

Peiwang Tang
Institute of Advanced Technology, University of Science and Technology of China
Hefei 230026, China
G60 STI Valley Industry & Innovation Institute, Jiaxing University
Jiaxing 314001, China
tpw@mail.ustc.edu.cn

Xianchao Zhang*
Key Laboratory of Medical Electronics and Digital Health of Zhejiang Province, Jiaxing University
Jiaxing 314001, China
Engineering Research Center of Intelligent Human Health Situation Awareness of Zhejiang Provincey, Jiaxing University
Jiaxing 314001, China
zhangxianchao@zjxu.edu.cn

ABSTRACT

The Transformer architecture yields state-of-the-art results in many tasks such as natural language processing (NLP) and computer vision (CV), since the ability to efficiently capture the precise long-range dependency coupling between input sequences. With this advanced capability, however, the quadratic time complexity and high memory usage prevents the Transformer from dealing with long time-series forecasting problem (LTFP). To address these difficulties: (i) we revisit the learned attention patterns of the vanilla self-attention, redesigned the calculation method of self-attention based the **Maximum Entropy Principle**. (ii) we propose a new method to sparse the self-attention, which can prevent the loss of more important self-attention scores due to random sampling.(iii) We propose Keys/Values Distilling method motivated that a large amount of feature in the original self-attention map is redundant, which can further reduce the time and spatial complexity and make it possible to input longer time-series. Finally, we propose a method that combines the encoder-decoder architecture with seasonal-trend decomposition, i.e., using the encoder-decoder architecture to capture more specific seasonal parts. A large number of experiments on several large-scale datasets show that our Infomaxformer is obviously superior to the existing methods. We expect this to open up a new solution for Transformer to solve LTFP, and exploring the ability of the Transformer architecture to capture much longer temporal dependencies.

KEYWORDS

Maximum Entropy; Transformer; Time-Series; Forecasting

ACM Reference Format:
Peiwang Tang and Xianchao Zhang. 2023. Infomaxformer: Maximum Entropy Transformer for Long Time-Series Forecasting Problem. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023,* IFAAMAS, 9 pages.

*corresponding author

1 INTRODUCTION

Defined as an ordered dataset formed with time change, time-series refer to a series of ordered observations acquired according to time sequence [9], which is widely used in commercial and industrial fields, such as biomedical field [31], economic and financial field [11], electric power [52] and transportation field [49]. As an important part of time-series analysis, time-series forecasting mainly analyze the trend, periodicity, volatility and other time-series patterns of time-series by using the time-series data observed in history and the relevant rules that have been mastered, so as to predict the situation in the future [3, 25, 46]. In practical applications, we can use a large number of past time-series to achieve long-term prediction for the future, i.e., long time-series forecasting problem (LTFP). Recent deep prediction models have made great progress, especially Transformer based models [21, 27, 32, 43, 47]. The Transformer [45] shows better performance than the recurrent neural network (RNN) model in modeling the long-term dependence of sequence data, and has achieved the best results in the natural language processing (NLP) [10, 36] and computer vision (CV) [12, 15] fields, since its advanced self-attention mechanism.

However, there are still some problems in solving LTFP of existing Transformer models. First, the self-attention mechanism has high performance, but also brings high time complexity and memory usage [33, 50]. Although some large-scale Transformer models have produced impressive results in the NLP and CV fields [4, 37], they often require dozens or even hundreds of GPUs during training, which limits the possibility of Transformer models to solve LTFP. Although there have been some researches on reducing the time complexity and memory usage of the self-attention mechanism, only realize a limited reduce of complexity to $O(LlogL)$ [23, 26, 52]. Moreover, some methods for reducing the complexity only randomly select dot-product pairs, which will cause some performance loss and lead to the long-term dependence of the sequences that cannot be well captured by the self-attention mechanism. Second, it is unreliable to find the time dependence directly from the time-series, because these dependencies may be masked by the entangled temporal patterns.

In order to better solve LTFP, our work explicitly and deeply discussed the above problems, studied the sparsity of self-attention mechanism, decomposed the time-series, and updated the network components. Finally, we have conducted extensive experiments on five different datasets. The final experimental results show that our

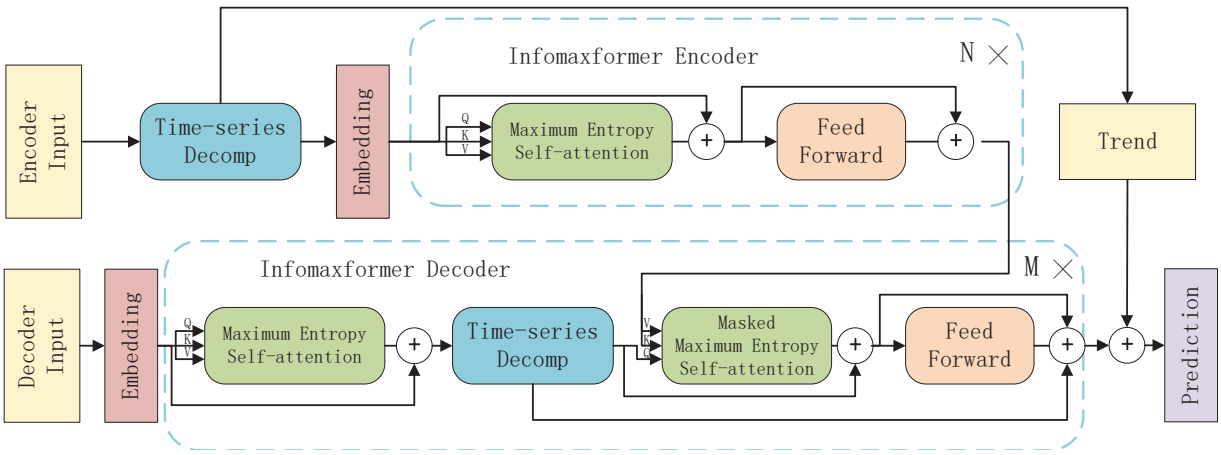


Figure 1: Infomaxformer architecture

proposed Infomaxformer can significantly improve the accuracy of prediction, and is superior to other state-of-the-art models. The contributions of this paper are summarized as follows:

- We review the calculation method of self-attention mechanism from the perspective of information entropy [41], and sparse the calculation of self-attention by using the **Maximum Entropy Principle** [20] to reduce the time complexity.
- In view of the data characteristic that local information of time-series is heavy spatial redundancy, we propose the Keys/Values Distilling method, which can further reduce the time and space complexity to $O(L)$, and help the model to accept longer sequence inputs.
- In order to decompose time-series and explain complex time-series patterns, we propose a decomposition method, which is combined with self-attention mechanism, to process complex time-series and extract more useful features.
- We have conducted extensive experiments on datasets in many different fields, and the final results show that our proposed model achieves the most advanced performance in a variety of experimental settings.

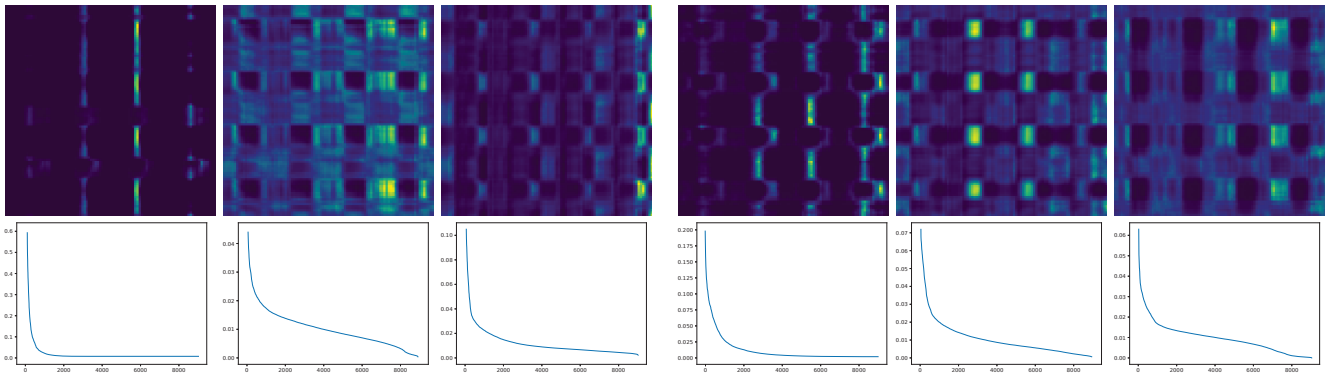
2 RELATED WORK

2.1 Time-Series Forecasting

The classical convolutional neural network (CNN) [24] model can extract the local information unrelated to the spatial position in the data [28]. In order to allow CNN to be used in the time-series, scholars designed multi-layer causal convolutions to ensure that only past information can be used for prediction [3, 6]. For the processing of long-term dependencies, the Temporal Convolutional Network (TCN) introduces the dilated convolutions, which changes the interval of original look-back window from 1 to d_l, where d_l is a layer-specific division rate. In traditional modeling, recurrent neural networ (RNN) is also widely used in the field of time-series prediction owing to its architecture naturally supports inputs and outputs with sequential relationships [29, 38, 40, 42]. The main idea is to use the memory state of RNN neurons to store all past effective information. However, RNN variants may be limited in learning the long-term dependency in the data. Since all the information in the past will decay with time and the difficult for RNN to learn the long-term memory [17]. Long Short-Term Memory networks (LSTM) [18] introduces some different operation gates to solve this problem, but it does not solve the long-term dependency well. To further these effort, attention mechanism is proposed to help the neural network to learn long-term memory information [2]. In short, the attention mechanism of time-series is to calculate the dynamic weight, find the weighted sum of past hidden states, and predict the output value with the summed state. In this way, the vector used for prediction can contain information that predicts a more informative time point for the current time point [13, 21, 27].

2.2 Sparse Attention

In the standard self-attention mechanism, each token needs to pay attention to all other tokens [45]. However, for the trained transformer, the learned attention matrix $\mathbf{A}$ is often very sparse across most data points [7]. Therefore, the computational complexity can be reduced by limiting the number of queries that want to participate in the query-key pairs through the incorporating structural bias. The existing methods can be divided into two categories: position-based and content-based sparse attention [30]. In position-based sparse attention, the attention matrix is limited to some predefined patterns [34, 48]. Although these spark patterns change in different ways, some of them can be decomposed into some atomic sparse patterns, e.g., global attention, band attention, dilated attention, random attention, block local attention [5, 14]. Many spark patterns include one or more of the above atomic sparse patterns [51]. Another work is to create a sparse graph based on the input content. A simple method is to select keywords that may have a large similarity score with a given query. In order to construct the sparse graph effectively, the maximum inner product search problem can be repeated, i.e, the key with the maximum dot product can be found by a query without calculating all dot-product terms [26, 52]. For example, Routing transformer [39] uses K-means clustering to cluster queries and keys on the same group of centroid vectors. Each query only focuses on the keys belonging



(a) Softmax scores at Head1@Encoder layer (b) Softmax scores at Head7@Encoder layer

Figure 2: The Softmax scores in the self-attention from canonical Transformer trained on $ETTh_1$ dataset

to the same cluster. Reformer [23] uses location sensitive hashing (LSH) to select key-value pairs for each query. The proposed LSH allows each token to attend only to the tokens in the same hash bucket. In Informer [52], based on query and key similarity sampling dot-product pairs, ProbSparse self-attention is proposed to reduce the time complexity of Transformer to $O(L \log L)$ and allows it to accept longer input.

3 METHODOLOGY

The problem of long time-series forecasting is to input the past sequence $\mathcal{X} = \left\{x_1, \cdots, x_{L_x} | x_i \in \mathbb{R}^{d_x}\right\}$, and the output is to predict corresponding future sequence $\mathcal{Y} = \left\{y_{L_x+1}, \cdots, y_{L_x+L_y} | y_i \in \mathbb{R}^{d_y}\right\}$, where L_x and L_y are the lengths of input and output sequences respectively, and d_x and d_y are the feature dimensions of input $\mathcal{X}$ and output $\mathcal{Y}$ respectively. The LTFP encourages a longer input's length L_x and a longer output's length L_y than previous works.

Our proposed Infomaxformer holds the encoder-decoder architecture and combines it with the decomposition structure to solve LTFP. Please refer to Figure 1 for an overview and the following sections for details.

3.1 Vanilla Self-attention Mechanism

The scaled dot-product attention mechanism in original Transformer [45] performs as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V} \tag{1}$$

i.e., $Attention$ is defined as an operation of ternary matrix, where $\mathbf{Q}(queries) \in \mathbb{R}^{L_Q \times d}$, $\mathbf{K}(keys) \in \mathbb{R}^{L_K \times d}$, $\mathbf{V}(values) \in \mathbb{R}^{L_V \times d}$, and d is the feature dimension. To further discuss the self-attention mechanism, the $Softmax$ function is expanded, and use q_i, k_i and v_i to represent the i-th row in $\mathbf{Q}$, K and $\mathbf{V}$ respectively. For the time-series with input length L, i represents the i-th data. Therefore, the original self-attention mechanism for the i-th data can be expressed as:

$$\mathcal{A}(i) = \sum_j^L \frac{e^{\frac{q_i k_j^T}{\sqrt{d}}}}{\sum_l^L e^{\frac{q_i k_l^T}{\sqrt{d}}}} v_j = \sum_j^L \frac{k(q_i, k_j)}{\sum_l^L k(q_i, k_l)} v_j \tag{2}$$

which $k(q_i, k_j) = exp(q_i k_j^T / \sqrt{d})$ [44].

Let $p(q_i, k_j) = k(q_i, k_j) / \sum_l^L k(q_i, k_l)$, $Attention$ can be abbreviated as:

$$\mathcal{A}(i) = \sum_j^L p\left(q_i, k_j\right) v_j \tag{3}$$

where $p\left(q_i, k_j\right)$ is the probability of v_i, then $\mathcal{A}(i)$ is the expectation of matrix $\mathbf{V}$. For the probability $p(q_i, k_j)$, it requires the quadratic times dot-product computation and $O(L_Q L_K)$ memory usage, which is the main reason why the traditional self-attention mechanism cannot handle long time-series (it is easy to lead to out-of-memory), and also the main disadvantage that limits its prediction ability.

Many previous studies have shown that the probability distribution of self-attention mechanism has potential sparsity [7, 30], and a selection strategy is designed for all $p(q_i, k_j)$ without significantly affecting the performance of the model [5, 14, 34, 48]. To motivate our approach, we first revisit the learned attention patterns of the vanilla self-attention and make a qualitative evaluation. According to Figure 2, in the first layer of encoder, the scores follows an obvious long tail distribution, and the Softmax scores has obvious blocking phenomenon, especially in the second and third layers. So a few dot-product pairs contribute to the major attention, and others generate negligible attention. Then, how to "select" them?

3.2 Reformulation Via The Lens of Information Entropy

We now provide the intuition to reformulate Equation (3) via the lens of information entropy [41]. Information entropy is a basic concept of information theory, which describes the uncertainty of possible events of information sources. Its formula is as follows:

$$H\left(x_i\right) = -\sum_{i=1}^L p(x_i) ln p(x_i) \tag{4}$$

where $p(x_i)$ represents the probability that the random event X is x_i. Any information has redundancy, which is related to the occurrence probability (uncertainty) of each symbol in the information. The probability and the amout of information generated are positively correlated. Information is used to eliminate random uncertainty,

and information entropy is a measure of the amount of information needed to eliminate uncertainty, i.e., the amount of information that an unknown event may contain.

Maximum Entropy Principle *When only some knowledge about the unknown distribution is mastered, the probability distribution with the largest entropy value should be selected [20].*

It is difficult to determine the probability distribution of random variables. Generally, only the average values or the values under certain limited conditions can be measured. There can be many (even infinite) distributions that meet the measured values. The maximum entropy principle is a criterion for selecting the statistical characteristics of random variables that best meet the objective conditions, also known as the Maximum Information Principle. Based on this principle, it is effective to select a distribution with maximum entropy as the distribution of the random variable.

Maximum Entropy Self-attention From Equation (3), the i-th query's attention on all the keys are defined as a probability distributions $\mathbf{p}_i$ and the output is its composition with values $\mathbf{V}$. According to the maximum entropy principle, the dominant dot-product pairs encourage the corresponding entropy of $\mathbf{p}_i$ to be maximum. However, the traversing of all the $\mathbf{p}_i$ still needs to calculate each dot-product pair, i.e., the time complexity is $O(L^2)$. Motivated by this, we propose a very simple but effective approximation method to obtain the query information entropy measurement.

Proposition 3.1. *For all probability distributions $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, if $\sigma_{p_i} < \sigma_{p_j}$, it can be considered that $H(i) > H(j)$.*

If the i-th query's $\mathbf{p}_i$ gains a smaller variance, its information entropy is larger and has a higher possibility to contain the dominate dot-product pairs. Variance is a measure of the degree of dispersion of a group of data. The variance of data subject to the same distribution is the same, so we only need to randomly sample constant U from $\mathbf{K}$ to calculate the variance of the i-th query's probability distribution $\mathbf{p}_i$, which only need to calculate $O(L_Q)$ dot-product for each query-key lookup and the layer memory usage maintains $O(L_Q)$. Then, select sparse Top-u from $\mathbf{Q}$ as $\bar{\mathbf{Q}}$ to calculate the standard dot-product pair, so the time complexity and memory usage maintains $O(uL_K)$. However, the rest of queries can't be left without any calculation.

Theorem 3.2. *In the case of discrete sources, for discrete sources with $\boldsymbol{L}$ symbols, the information entropy can reach the maximum value only when they appear with equal probability, that is, the average uncertainty of sources with equal probability distribution is the maximum*

Based on the proposed measurement and Theorem 3.2, we have the maximum entropy self-attention, i.e., MEA (the pseudo-code is in Appendix):

$$\mathcal{A}(i) = \begin{cases} \sum_j^L p\left(q_i, k_j\right) v_j & \text{, if top-u} \\ \sum_j^L v_j / L & \text{, otherwise} \end{cases} \tag{5}$$

3.3 Embedding Method

As shown in Figure 3, the input embedding consists of three parts, a scalar, a local position and a global time stamp. We use scalar

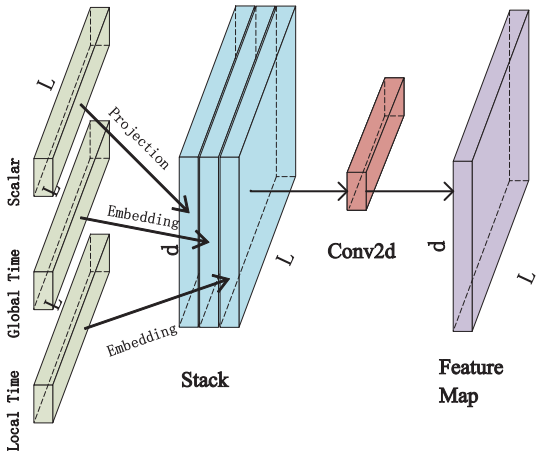


Figure 3: The Embedding Method

projection SP, local position embedding PE [45] and time embedding TE [52] to deal with the above parts respectively:

$$SP = Conv1d(x_i^t) \tag{6}$$

$$\begin{aligned} PE_{(i,2j)} &= sin\left(i/10000^{2j/d_{model}}\right) \\ PE_{(i,2j+1)} &= cos\left(i/10000^{2j/d_{model}}\right) \end{aligned} \tag{7}$$

$$TE = E(month) + E(day) + E(hour) + E(minute) \tag{8}$$

For the Equation (6), we project the scalar context x_i^t into d_{model}-dim vector with 1-D convolutional filters. The kernel width is 3, stride is 1, the input channel is d_x and the output channel is d_{model}. For the Equation (7), $i \in \{1, \ldots, L_x\}$, $j \in \{1, \ldots, \lfloor d_{model}/2 \rfloor\}$, d_{model} is the feature dimension after embedding. For the Equation (8), E is a learnable stamp embeddings with limited vocab size (up to 60, namely taking minutes as the finest granularity).

For the three different features finally obtained, instead of the method of addition [46, 52], we stacked them together and reduced their dimension through a two-dimensional convolution, that is, the input channel of convolution is 3 and the output channel is 1:

$$\mathcal{X} = Conv2d(Stack(SP, PE, TE)) \tag{9}$$

where the kernel width and stride is $(1, 1)$.

3.4 Keys/Values Distilling

In previous works, a feed-forward network with a single hidden layer is proposed to linearly project the queries, keys and values [45, 52]. As the natural consequence of the original sequence linear project, the queries, keys and values have a lot of redundant features. We use the distilling operation to privilege the superior keys and values with dominating features and make a focused feature map in the self-attention mechanism. It trims the time dimension of the input sharply, does not arbitrarily delete the feature of the input sequence, but recombines them into a new heads weights matrix.

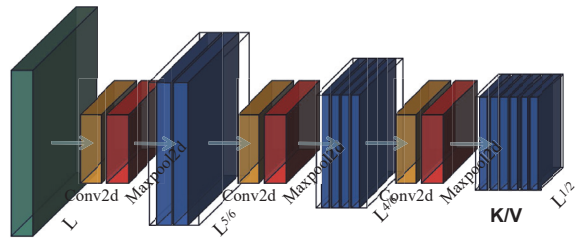


Figure 4: The Keys/Values Distilling

As shown in Figure 4, we distilling keys/values using a three-step convolution operation:

$$\begin{aligned}\mathcal{K}_1 &= Maxpool2d(Relu(Conv2d(\mathcal{X})))\\ \mathcal{K}_2 &= Maxpool2d(Relu(Conv2d(\mathcal{K}_1)))\\ \mathcal{K}_3 &= Maxpool2d(Relu(Conv2d(\mathcal{K}_2)))\end{aligned} \tag{10}$$

where $\mathcal{X} \in \mathbb{R}^{L_x \times d_{model}}$. $Conv2d()$ performs an 2-D convolutional filters (kernel size=(2, 2)) with the *Relu* activation function, the number of input channels is the number of heads, and the number of output channels is h times the number of input channels, so after three-step convolution, the number of output channels, i.e., the number of heads, is h^3 (this can be modified as needed). $Maxpool2d()$ performs an 2-D max pooling (kernel size and stride is (l, h)). Therefore, after three-step convolution, $\mathcal{K}_3 \in \mathbb{R}^{\frac{L_X}{l^3} \times \frac{d_{model}}{h^3}}$, i.e. $\mathcal{K} \in \mathbb{R}^{L_K \times \frac{d_{model}}{h^3}}$, $\mathcal{V} \in \mathbb{R}^{L_V \times \frac{d_{model}}{h^3}}$, L_K and L_V is L_X/l^3.

Complexity Analysis: Now we know that L_K is L_X/l^3, so the time complexity and space complexity of our MEA is $O(uL_X/l^3)$. We set $u = c\sqrt{L_Q}$, $l = L_X^{1/6}$, c is a constant sampling factor, u varies linearly with L_Q, so:

$$uL_X/l^3 = c\sqrt{L_Q}L_X/(L_X^{1/6})^3 = c\sqrt{L_Q}\sqrt{L_X} \tag{11}$$

where $L_Q = L_X = L$. Consequently, our time complexity and space complexit can reach linear $O(L)$.

3.5 Time-Series Decomposition

In order to make long-term prediction under the input of long time-series, we use the concept of decomposition to learn complex time patterns, which can separate the time-series into trend and seasonal [8, 19, 46]. These two parts respectively represent two features including long-term development trend and seasonality of the time-series, which are different in different time-series. To overcome such a problem, we introduce a time-series decomposition block (TSD), which can propose the development trend and seasonality of the time-series from the input. Specifically, we use the moving average to smooth out periodic fluctuations, extract long-term trends, and highlight seasonality. For an input sequence $\mathcal{X} \in \mathbb{R}^{L \times d}$ of length L, this process can be formulated as:

$$\begin{aligned}\mathcal{X}_t &= AvgPool(\mathcal{X})\\ \mathcal{X}_s &= \mathcal{X} - \mathcal{X}_t\end{aligned} \tag{12}$$

where $\mathcal{X}_s, \mathcal{X}_t \in \mathbb{R}^{L \times d}$ represent extracted seasonal and trend, respectively. We use $\mathcal{X}_t, \mathcal{X}_s = TimeSeriesDecomp(\mathcal{X})$ to summarize above equations.

3.6 Encoder and Decoder

Encoder: As shown in Figure 1, the encoder focuses on modeling of the seasonal part. Our encoder layers are composed of two sub-blocks. The first is a MEA mechanism, and the second is a simple, position-wise fully connected feed-forward network (MLP). We employ residual connections [16] around each of the sub-blocks, but unlike previous structures [12, 45], layer normalization [1] was not performed. The input of the encoder is only the seasonal part $\mathcal{X}_s$ of the input sequence $\mathcal{X}$, and the output only contains the seasonal information of the past and will be used as cross information to help the decoder better predict the seasonal information of the future sequence.

$$\begin{aligned}\mathcal{X}_t, \mathcal{X}_s &= TimeSeriesDecomp(\mathcal{X})\\ \mathcal{X}_{s1}^n &= \mathcal{X}_s^{n-1} + MEA(\mathcal{X}_s^{n-1})\\ \mathcal{X}_s^n &= \mathcal{X}_{s1}^n + MLP(\mathcal{X}_{s1}^n)\end{aligned} \tag{13}$$

where $n = 1 \ldots N$, $\mathcal{X}_t^0 = \mathcal{X}_t$, $\mathcal{X}_{eno} = \mathcal{X}_t^N$, N is the number of layers of the encoder, and MLP consists of two 1-D convolution operations.

Decoder: In addition to the two sub-blocks in each encoder layer, the decoder in classic Transformer also inserts a third sub-block in the two sub-blocks, which performs self-attention on the output of the encoder layers. On this basis, we insert the fourth sub-block in the decoder, i.e., the time-series decomposition block. Similar to the encoder, we employ residual connections around each of the sub-blocks, but did not perform layer normalization. We input the following vectors to the decoder:

$$\mathcal{X}_{dei} = Concat(\mathcal{X}_{label}, \mathcal{X}_0) \in \mathbb{R}^{(L_{label}+L_y) \times d_{model}} \tag{14}$$

where $\mathcal{X}_{label} \in \mathbb{R}^{L_{label} \times d_{model}}$ is start token, L_{label} is the label length, $\mathcal{X}_0 \in \mathbb{R}^{L_y \times d_{model}}$ is a placeholder for the target sequence (set the scalar to 0). By setting masked dot-products to negative infinity, masked multi-head attention is applied to the MEA calculation (MMEA). This masking ensures that the prediction of position i can only rely on the known outputs of positions less than i, which avoids auto-regressive. A fully connected layer acquires the final output, and its outsize is d_y. For the time-series, the trend changes are not obvious, but the specific seasonal is different. We use the decoder to predict the seasonal of future data, and use the average of input data to approximate the trend part of future data.

$$\begin{aligned}\mathcal{X}_1^n &= \mathcal{X}^{n-1} + MEA(\mathcal{X}^{n-1})\\ \mathcal{X}_t^n, \mathcal{X}_s^n &= TimeSeriesDecomp(\mathcal{X}_1^n)\\ \mathcal{X}_{s1}^n &= \mathcal{X}_s^n + MMEA(\mathcal{X}_s^n, \mathcal{X}_{eno})\\ \mathcal{X}^n &= \mathcal{X}_{s1}^n + MLP(\mathcal{X}_{s1}^n) + \mathcal{X}_t^n\\ \mathcal{Y} &= Mean(\mathcal{X}_t) + \mathcal{X}^M\end{aligned} \tag{15}$$

where $n = 1 \ldots M$, $\mathcal{X}^0 = \mathcal{X}_{dei}$, M is the number of layers of the decoder.

Loss Function: Our loss function is calculated by the mean square error (MSE) between the model output data y_o and the real data y. and the loss is propagated back from the decoder's outputs across the entire model.

Table 1: Multivariate long time-series forecasting results on five cases

Methods		Infomaxformer		Autoformer		Informer		Reformer		LogTrans		Transformer		LSTM	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	24	**0.190**	**0.305**	0.195	0.312	0.310	0.400	0.280	0.381	0.231	0.338	0.244	0.350	0.338	0.419
	48	**0.209**	**0.321**	0.221	0.332	0.357	0.425	0.273	0.370	0.287	0.373	0.260	0.359	0.334	0.412
	96	**0.218**	**0.328**	0.230	0.340	0.367	0.434	0.291	0.381	0.292	0.377	0.278	0.373	0.330	0.409
	192	**0.239**	**0.346**	0.280	0.347	0.362	0.434	0.344	0.420	0.295	0.385	0.286	0.377	0.327	0.407
	384	**0.278**	**0.377**	-		0.450	0.483	0.327	0.404	0.323	0.397	0.290	0.378	0.320	0.403
$ETTh_1$	24	**0.478**	**0.489**	0.499	0.516	1.130	0.871	0.624	0.578	0.505	0.513	0.882	0.723	1.232	0.839
	48	**0.552**	**0.528**	0.562	0.552	1.231	0.905	0.727	0.638	0.568	0.544	1.311	0.954	1.261	0.873
	96	**0.564**	**0.543**	0.611	0.579	1.345	0.953	0.930	0.743	0.714	0.629	1.957	1.199	1.268	0.873
	192	**0.556**	**0.544**	0.724	0.625	1.643	1.061	1.124	0.821	0.865	0.717	1.758	1.136	1.266	0.871
	384	**0.597**	**0.570**	-		1.499	1.004	1.270	0.862	0.952	0.749	1.405	0.981	1.273	0.873
$ETTh_2$	24	**0.436**	**0.476**	0.437	0.493	2.048	1.173	0.975	0.787	0.621	0.617	1.016	0.814	3.291	1.385
	48	**0.629**	**0.544**	0.658	0.643	3.047	1.471	1.652	1.027	1.168	0.985	2.199	1.242	3.378	1.408
	96	**0.593**	**0.533**	0.686	0.646	6.882	2.258	3.301	1.427	2.279	1.265	5.862	2.052	3.488	1.432
	192	**0.703**	**0.607**	0.792	0.671	5.070	1.885	3.774	1.617	4.207	1.776	4.045	1.675	3.489	1.434
	384	**0.575**	**0.557**	-		4.080	1.669	3.363	1.465	3.032	1.526	3.549	1.516	3.486	1.430
$ETTm_1$	24	**0.330**	**0.397**	0.414	0.438	0.354	0.401	0.430	0.453	0.882	0.666	0.355	0.403	1.121	0.791
	48	**0.418**	**0.454**	0.537	0.505	0.533	0.521	0.578	0.544	0.951	0.707	0.514	0.526	1.130	0.799
	96	**0.494**	**0.502**	0.545	0.517	0.592	0.571	0.710	0.612	0.558	0.540	0.740	0.657	1.141	0.805
	192	**0.558**	**0.531**	0.605	0.534	0.768	0.682	0.896	0.702	0.591	0.565	0.700	0.641	1.141	0.806
	384	**0.611**	**0.561**	-		0.938	0.765	1.072	0.781	0.767	0.650	0.838	0.719	1.153	0.810
Weather	24	**0.307**	**0.357**	0.455	0.489	0.348	0.401	0.370	0.426	0.385	0.427	0.326	0.378	0.492	0.500
	48	**0.381**	**0.422**	0.544	0.542	0.488	0.505	0.443	0.475	0.498	0.505	0.447	0.664	0.497	0.504
	96	**0.456**	**0.481**	0.554	0.546	0.603	0.574	0.511	0.520	0.562	0.550	0.548	0.532	0.500	0.506
	192	**0.508**	**0.517**	0.585	0.560	0.700	0.632	0.537	0.541	0.591	0.570	0.620	0.575	0.503	0.517
	384	**0.511**	**0.514**	-		0.681	0.621	0.536	0.535	0.622	0.585	0.630	0.579	0.513	0.524

1: A lower MSE or MAE indicates a better prediction, and we use black numbers to indicate the best performance.
2: Due to the limitation of memory, the batch size of some models is changed to 16, which is indicated by underlined numbers.
3: The '-' indicates that there is still not enough memory after the batch size is changed to 16.

4 EXPERIMENT

4.1 Datasets

ETT(Electricity Transformer Temperature):[1] The ETT is a key indicator for long-term deployment of the electric power, which collects data from two counties in China from July 2016 to July 2018 for a total of two years.

ECL (Electricity Consuming Load):[2] It collects the electricity consumption (Kwh) of 321 customers. Due to the lack of data [26], we follow the settings in Informer [52] to convert the dataset to hourly consumption for 2 years.

Weather:[3] This dataset contains the local climatologicale data of nearly 1600 locations in the United States. The data are collected by once an hour from 2010 to 2013.

[1]ETT dataset was acquired at [52].
[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/.

4.2 Experimental Details

Baselines: We selected six methods as comparison, including Transformer [45], four latest state-of-the-art Transformer-based models: Reformer [23], LogTrans [26], Informer [52], Autoformer [46], and one RNN-based models: LSTM (Long Short-Term Memory networks) [18].

Experiment Setting: Our experiment was implemented in Pytoch [35], and all the experiments are conducted on a single Nvidia RTX 3090 GPU (24GB memory). The input of each dataset is zero-mean normalized. We use two evaluation metrics, including mean square error (MSE) and mean absolute error (MAE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\frac{(y-\hat{y})^2}{d} \tag{16}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\frac{|y-\hat{y}|}{d} \tag{17}$$

where n is the length of the sequence and d is the dimension of data at each time point. We use these two evaluation metrics on each prediction window to calculate the average of forecasts and roll the whole set with $stride = 1$.

Table 2: Different input lengths for two prediction lengths in $ETTh_1$

Predicition length		336						480				
Encoder's input		336	480	720	960	1200	1440	480	720	960	1200	1440
Informer	MSE	1.474	1.576	1.644	1.607	1.580	1.766	1.441	1.531	1.508	1.428	1.520
	MAE	0.999	1.024	1.045	1.043	1.037	1.124	0.971	0.998	0.997	0.967	1.017
Reformer	MSE	1.000	1.043	1.200	1.159	-	-	1.148	1.259	-	-	-
	MAE	0.766	0.790	0.842	0.829			0.827	0.870			
Transformer	MSE	1.085	1.161	1.630	1.922	-	-	1.083	1.410	-	-	-
	MAE	0.830	0.858	1.065	1.159			0.836	0.969			
LogTrans	MSE	1.136	1.127	1.059	1.075	1.060	1.067	0.888	0.940	0.892	0.885	0.914
	MAE	0.851	0.849	0.817	0.823	0.818	0.816	0.733	0.764	0.736	0.732	0.745
Autoformer	MSE	0.581	0.560	0.748	-	-	-	0.573	-	-	-	-
	MAE	0.547	0.547	0.650				0.558				
Infomaxformer	MSE	**0.567**	**0.556**	**0.544**	**0.566**	**0.568**	**0.624**	**0.537**	**0.583**	**0.597**	**0.599**	**0.709**
	MAE	**0.545**	**0.547**	**0.543**	**0.563**	**0.564**	**0.602**	**0.551**	**0.581**	**0.587**	**0.585**	**0.653**

1: The '-' indicates failure for the out-of-memory.

Our implementation details follows common practice of Informer [52] training, and all experiments are repeated five times. We use Adam [22] optimizer for optimization with a learning rate starts from $1e^{-4}$, decaying two times smaller every epoch, and the batch size is 32. The number of encoder layers is 3 and the number of decoder layers is 2. There is no limit to the total number of epochs, with appropriate early stopping, i.e., when the loss of the validation set does not decrease on three epochs, the training will be stopped. More detailed settings can be found in Appendix 4.2.

4.3 Multivariate Time-series Forecasting

To compare the performance of different prediction lengths, we fixed the input length L_x to 784 and gradually extended the prediction length L_y, i.e., {24, 48, 96, 192, 384}, representing {6h, 12h, 24h, 48h, 96h} in ETTm, {1d, 2d, 4d, 8d, 16d} in {ETTh, ECL, Weather}, and we set the length of the label to double L_y.

As shown in Table 1, our proposed Infomaxformer model achieves the best performance in all benchmarks and all predicted length settings. Although the performance of Autoformer is closest to our model, it can only set the batch size to 32 when the prediction length is 24. When the prediction length is 384, the batch size to 16 will also lead to out-of-memory. This shows that our proposed Infomaxformer model can increase the prediction ability, while greatly reducing the use of memory. In addition, we also found that with the increase of prediction length, the prediction performance of Infomaxformer is more stable, and there is no sudden drop in performance, which means that Infomaxformer maintains good long-term robustness. Low memory usage, good robustness, high-performance prediction, etc., which is very meaningful for practical applications, and our model has the above advantages.

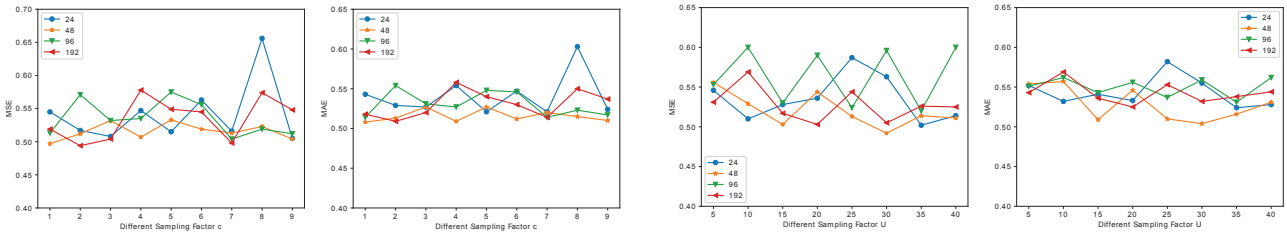
4.4 Parameter Sensitivity

We perform the sensitivity analysis of the proposed Infomaxformer model on $ETTh_1$.

Input Length: As shown in the Table 2, we gradually extended the size of the input sequence L_x, i.e., {336, 480, 720, 960, 1200, 1440}, while keeping the predicted length L_y unchanged, and the length L_{label} of the label sequence is consistent with L_y. Our L_y selected two values, 336 and 480. In Table 2, it can be seen that increasing the input length will lead to the decrease of MSE and MAE, because long input will bring repeated short-term patterns. However, as the input sequence increases, there may be more dependencies between the inputs, and the influence of noise in the input data will also increase. Some models can not effectively eliminate the influence of these noises and can not better grasp the dependence of long time-series, so MSE and MAE may increase. In this experiment, the batch size of Autoformer is set to 16, because too large batch size will directly lead to out-of-memory. It can be seen that there is still a certain gap between the memory usage of many model theories and the actual application. It seems that the Autoformer with memory usage of $O(L \log L)$ is not better than the original Transformer with memory usage of $O(L^2)$.

Sampling Factor c: The sampling factor c controls the information bandwidth of MEA in Equation (5). We start with small factors (=1) and gradually increase to large factors (=9). As can be seen in Figure 5(a), the performance of our Infomaxformer does not change much, and it is not similar to the case that the performance of Informer is slightly improved with the change of sampling factor. This is because our initial sampling is $\sqrt{L}$, not $\log L$ in Informer [52], so enough dominant queries is selected to calculate the dot-product to prevent the loss of important features.

Sampling Factor U: The sampling factor U controls how many keys are selected to calculate the variance. Although the variance



(a) Sampling Factor c (b) Sampling Factor U

Figure 5: The parameter sensitivity of two components in Infomaxformer

of data subject to the same distribution is the same, too few data samples will cause the calculated variance to be not the actual variance. It can be seen from Figure 5(b) that when U is too low, the performance of the Infomaxformer model will indeed be affected. However, when U gradually increases, the performance of the model tends to be stable.

Table 3: Ablation study of the Infomaxformer

Encoder's input		720	960	1200	1440
Infomaxformer	MSE	**0.566**	**0.588**	**0.633**	**0.573**
	MAE	**0.579**	**0.585**	**0.617**	**0.585**
Infomaxformer^1	MSE	1.326	1.237	0.880	1.290
	MAE	0.911	0.891	0.736	0.882
Infomaxformer^2	MSE	0.906	0.681	0.839	0.831
	MAE	0.763	0.638	0.727	0.720
Infomaxformer^3	MSE	1.074	1.018	0.964	1.121
	MAE	0.822	0.823	0.789	0.846

4.5 Ablation Study

We also performed some additional experiments for ablation analysis on $ETTh_1$. Infomaxformer^1 indicates that Keys/Values Distilling is replaced with the original projection operation, Infomaxformer^2 indicates that MEA is replaced with the canonical self-attention mechanism, and Infomaxformer^3 indicates that we have not taken our proposed Time-Series Decomposition. In this experiment, we set the predicted length L_y to 720 and select four ultra long input lengths. As shown in the Table 3, without any part of the Infomaxformer model, the performance will be degraded. Only the complete Infomaxformer can achieve the best performance. The impact of MEA mechanism on the performance of Infomaxformer is not as obvious as the other two, because the mechanism focuses on sparing self-attention and reducing time complexity.

Different Time-Series Decomposition: In order to better compare the TSD proposed by us with the Series decomposition block (SDB) proposed by Autoformer [46], we freely combined MEA, Auto-Correlation (AC) [46] with TSD, SDB. As shown in the Table 4, we found an interesting phenomenon. No matter Infomaxformer (TSD+MEA) or Autoformer (SDB+AC), only the complete state

Table 4: Comparative experiment of different Decomposition Block in Infomaxformer and Autoformer

Predicition length		96	192	384	720
TSD+MEA	MSE	**0.554**	**0.496**	**0.567**	**0.551**
	MAE	**0.540**	**0.513**	**0.555**	**0.559**
SDB+MEA	MSE	0.709	0.770	0.683	0.677
	MAE	0.624	0.662	0.626	0.626
TSD+AC	MSE	0.706	0.677	-	-
	MAE	0.627	0.605	-	-
SDB+AC	MSE	0.623	0.699	-	-
	MAE	0.577	0.619	-	-

model has the best performance. Our TSD is inferior to SDB in short sequence prediction ($L_y = 96$), but superior to SDB in long sequence prediction ($L_y = 192$). The sparsity of MEA results in the model being able to output longer sequences, but it is precisely because of this sparsity that the performance of MEA is inferior to AC when memory allows. However, the perfect combination of TSD and MEA proposed by us can output longer sequences and maintain better prediction performance.

5 CONCLUSION

In this paper, we studied the long time-series forecasting problem (LTFP) and proposed Infomaxformer to predict time-series. Specifically, we designed the Maximum Enterprise Self-attention mechanism and Keys/Values Distilling operation to deal with the challenges of quadratic time complexity and quadratic memory usage in vanilla Transformers. Finally, we reduce the time complexity and memory usage to $O(L)$. In addition, the well-designed time-series decomposition and the perfect combination with Transformer architecture can effectively deal with the complex time-series patterns of time-series, thus alleviating the limitations of the traditional decomposition architecture. The experiments on real-word data have proved the effectiveness of Infomaxformer in improving the prediction ability of LTFP.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

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

[4] Hangbo Bao, Li Dong, and Furu Wei. 2021. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254* (2021).

[5] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[6] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. 2017. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* (2017).

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509* (2019).

[8] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition. *J. Off. Stat* 6, 1 (1990), 3–73.

[9] Jonathan D Cryer. 1986. *Time series analysis*. Vol. 286. Springer.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Luca Di Persio and Oleksandr Honchar. 2016. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing* 10, 2016 (2016), 403–413.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, and Sylvain and Gelly. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[13] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. 2019. Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*. 2527–2535.

[14] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. In *Proceedings of NAACL-HLT*. 1315–1325.

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.

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

[17] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

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

[19] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.

[20] Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical review* 106, 4 (1957), 620.

[21] Hao Jiang, Mingyao Cui, Derrick Wing Kwan Ng, and Linglong Dai. 2022. Accurate Channel Prediction Based on Transformer: Making Mobility Negligible. *IEEE Journal on Selected Areas in Communications* (2022).

[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6980

[23] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The Efficient Transformer. In *ICLR*.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[25] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 95–104.

[26] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems* 32 (2019).

[27] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (2021), 1748–1764.

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

[29] Bryan Lim, Stefan Zohren, and Stephen Roberts. 2020. Recurrent neural filters: Learning independent bayesian filtering steps for time series prediction. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[30] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. *arXiv preprint arXiv:2106.04554* (2021).

[31] Luchen Liu, Jianhao Shen, Ming Zhang, Zichang Wang, and Jian Tang. 2018. Learning the joint representation of heterogeneous temporal events for clinical endpoint prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

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

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10012–10022.

[34] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International conference on machine learning*. PMLR, 4055–4064.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[37] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[38] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. *Advances in neural information processing systems* 31 (2018).

[39] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics* 9 (2021), 53–68.

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

[41] Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[42] Peiwang Tang and Xianchao Zhang. 2022. Features Fusion Framework for Multimodal Irregular Time-series Events. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 366–379.

[43] Peiwang Tang and Xianchao Zhang. 2022. MTSMAE: Masked Autoencoders for Multivariate Time-Series Forecasting. *arXiv preprint arXiv:2210.02199* (2022).

[44] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 4344–4353.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[46] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[47] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. 2020. Adversarial sparse transformer for time series forecasting. *Advances in neural information processing systems* 33 (2020), 17105–17115.

[48] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. 2019. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070* (2019).

[49] Yi Yin and Pengjian Shang. 2016. Multivariate multiscale sample entropy of traffic time series. *Nonlinear Dynamics* 86, 1 (2016), 479–488.

[50] Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan L Yuille, and Wei Shen. 2021. Glance-and-gaze vision transformer. *Advances in Neural Information Processing Systems* 34 (2021), 12992–13003.

[51] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems* 33 (2020), 17283–17297.

[52] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11106–11115.