Learning to Rotate: Quaternion Transformer for Complicated Periodical Time Series Forecasting

Weiqi Chen
jarvus.cwq@alibaba-inc.com
DAMO Academy, Alibaba Group
Hangzhou, China

Wenwei Wang
duoluo.www@alibaba-inc.com
DAMO Academy, Alibaba Group
Hangzhou, China

Bingqing Peng
pengbingqing.pbq@alibaba-inc.com
DAMO Academy, Alibaba Group
Hangzhou, China

Qingsong Wen
qingsong.wen@alibaba-inc.com
DAMO Academy, Alibaba Group
Bellevue, WA, USA

Tian Zhou
tian.zt@alibaba-inc.com
DAMO Academy, Alibaba Group
Hangzhou, China

Liang Sun
liang.sun@alibaba-inc.com
DAMO Academy, Alibaba Group
Bellevue, WA, USA

ABSTRACT

Time series forecasting is a critical and challenging problem in many real applications. Recently, Transformer-based models prevail in time series forecasting due to their advancement in long-range dependencies learning. Besides, some models introduce series decomposition to further unveil reliable yet plain temporal dependencies. Unfortunately, few models could handle complicated periodical patterns, such as multiple periods, variable periods, and phase shifts in real-world datasets. Meanwhile, the notorious quadratic complexity of dot-product attentions hampers long sequence modeling. To address these challenges, we design an innovative framework *Quaternion Transformer* (Quatformer), along with three major components: 1). *learning-to-rotate attention* (LRA) based on quaternions which introduces learnable period and phase information to depict intricate periodical patterns. 2). *trend normalization* to normalize the series representations in hidden layers of the model considering the slowly varying characteristic of trend. 3). *decoupling LRA* using global memory to achieve linear complexity without losing prediction accuracy. We evaluate our framework on multiple real-world time series datasets and observe an average 8.1% and up to 18.5% MSE improvement over the best state-of-the-art baseline.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Supervised learning*; • **Mathematics of computing** → **Time series analysis**.

KEYWORDS

time series forecasting, decomposition, transformer, periodicity

ACM Reference Format:
Weiqi Chen, Wenwei Wang, Bingqing Peng, Qingsong Wen, Tian Zhou, and Liang Sun. 2022. Learning to Rotate: Quaternion Transformer for Complicated Periodical Time Series Forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539234

KDD '22, August 14–18, 2022, Washington, DC, USA

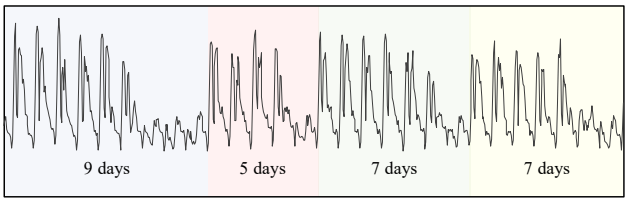
ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539234

1 INTRODUCTION

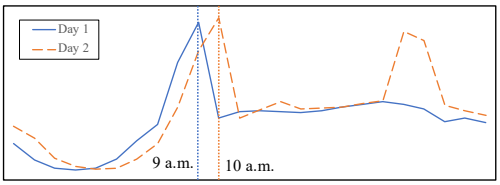
Time series forecasting plays a crucial role in many real-world scenarios, e.g., traffic forecasting [6], financial prediction [30], electric load forecasting [32], and business optimization [19]. Accurate forecasting can help people make better decisions. For instance, predicting the future status of electricity consumption is important for electricity generation, scheduling and management. In financial markets, forecasting currency exchange rates based on historical data helps in investment planning to maximize the profit.

Many time series are characterized by complex and diverse periodic components [26]. For example, over 60% of time series in Microsoft big-data clusters are reported to be periodic [11], and the percentage is even higher for those business-oriented time series data due to periodic human behaviors. Although periodicity generally refers to repeated patterns in time series, the periodic component can be dynamic in many real-world applications, such as variable periods and phase shifts. We use two examples in Fig. 1 to illustrate it. The electricity consumption time series in Fig. 1(a) exhibits both daily and weekly periodic components, and the weekly period varies in the long weekend. In Fig. 1(b), the traffic occupancy time series is characterized by phase shift: peak hour in the second day is be slightly later than the first day due to weather conditions. In addition, multiple periodic components may co-exist and interlace with each other. Other components, such as changing trend and noises, can also interfere the periodic components. Given the complex periodic component, modeling periodic time series is a fundamental problem in time series forecasting.

Recent deep forecasting models have achieved great success, especially the Transformer-based models [16, 28, 29, 31] due to its excellent ability to model long-term dependencies for sequential data. Despite the powerful capacity of Transformers, how to model the temporal dependencies still remains a challenge as the dependencies could be obscured by entangled trend and periodical patterns. A most recent work, namely Autoformer [29], incorporates series decomposition into Transformers to model the trend and periodical dependencies separately and achieves great progress. It detects a number of potential periods of seasonal component with auto-correlation and delays the series based on the estimated



(a) Multiple Periods and Variable Periods



(b) Phases Shifts

Figure 1: Examples of Complicated Periodical Patterns. (a) is the electricity consumption time series, which exhibits daily and weekly period, and the weekly period varies due to long weekend; (b) is the traffic occupancy time series of two days with morning peaks occurring at different times.

periods for aggregation.However, this time delay aggregation can not han- dle complicated periodical patterns, as auto-correlation can only estimate invariable periods and the time delay operation cannot account for phase shifts and might result in incorrect correlations. On the other hand, traditional forecasting methods, e.g., seasonal ARIMA [9] and Prophet [24], also utilize the decomposition with heuristic periodic priors, yet fail to model complicated periodical patterns automatically and intelligently.

In this paper, our main goal is to model the complicated periodical patterns for accurate time series forecasting. Specially, we first design a new kernel, namely *rotatory softmax-kernel*, which rotates the sequence of representations w.r.t. the frequency (or period) of the given time series by representing them in quaternion forms, and thus incorporates the period and phase information when measuring pair-wise similarities. This kernel can be directly plugged into dot-product attention. To further handle multiple periods, variable periods, and phase shifts, we propose to learn the latent frequencies and phases in a data-driven manner, and derive *learning-to-rotate attention*, which retains the capacity of modeling long-range dependencies of attention mechanism and makes use of periodical characteristics of time series. Besides, we propose *trend normalization* to better normalize the series representations in hidden layers by enforcing the slowly varying characteristic to the trend component.

Another challenge is the quadratic complexity of Transformer which hinders long time series modeling. Given the observation that similar temporal patterns occur repeatedly throughout the entire time series, it's likely to compress the long series into compact ones. We thus develop *decoupling attention* to decouple the attention mechanism into two attentions with linear complexity by introducing an additional latent series with a fixed length storing global memory. In a nutshell, input series is compressed into a fixed length series with the first attention, and then uncompressed with the second one. Our framework, namely *Quaternion Transformer*, achieves the superior performance on six benchmark datasets with linear complexity.

To summarize, our main contributions are as follows:

- To capture complicated periodical patterns (e.g., multiple periods, variable periods, and phase shifts) of time series, we propose *learning-to-rotate attention* (LRA) with quaternions to empower the canonical dot-product attention with the capacity of utilizing the learned period and phase information when calculating position-wise similarity.
- We propose *trend normalization* to normalize the series representations in hidden layers of the model rather than the widely used layer normalization, considering the inductive bias that the trend of time series is a slowly varying function.
- To break the bottleneck of the computational efficiency of long time series modeling, we propose *decoupling LRA* with global memory, which can discover the global temporal patterns occur repeatedly throughout the entire time series and reduce the computational complexity from quadratic to linear.
- The proposed framework *Quaternion Transformer* achieves superior performance on six real-world time series datasets, with an average 8.1% and up to 18.5% MSE improvement over the best state-of-the-art baseline.

2 RELATED WORK

2.1 Deep Time Series Forecasting

Over the past few years, deep forecasting methods have become prevailing in time series forecasting. FC-LSTM [23] forecasts time series with LSTM and fully-connected layers. TCN [2] introduces causal and dilated convolutions to model temporal patterns. DeepAR [19] is a LSTM-based model where the parameters of the distributions of the future values are predicted by LSTM. Most recently, Transformer-based models take advantage of attention mechanism to capture long-term temporal dependencies and have dominated this landscape [28]. For example, LogTrans [16] introduces local/logsparse attention, Reformer [13] approximates attention with locality-sensitive hashing, Informer [31] approximates attention with sparsity queries, and Autoformer [29] introduces series decomposition and time delay aggregation to better model the periodicity. However, existing deep forecasting models cannot fully capture the complicated periodical patterns. In this work, the *learning-to-rotate attention* is designed to address this problem.

2.2 Time Series Decomposition

Decomposition is useful in time series analysis by decomposing the original time series into several components, including seasonality, trend, and residual or irregular part [3, 27]. Many statistical forecasting models often adopt decomposition-based schemes especially for time series with seasonal patterns, e.g., seasonal ARIMA [9], TBATS [5], and Prophet [24]. Though introducing period modeling explicitly, these methods only follow some arbitrary yet simple assumptions, such as additive or multiplicative seasonality, to capture certain plain periodic effects and fail to model complicated periodical dependencies beyond such simplified assumptions. For

deep forecasting models, decomposition is also adopted either as a preprocessing block or inner block to enhance performance. ES-RNN [21] is a hybrid forecasting model by combing the classical exponential smoothing (ES) and RNN networks, where ES decomposes the time series into level, trend and seasonality components. N-BEATS [18] proposes an interpretable forecasting architecture by designing the trend and seasonality stacks as learnable inner blocks. Autoformer [29] also makes series decomposition as a basic inner block in the proposed Transformer-based forecasting model. Though decomposition brings significant performance improvement in forecasting, how to model the complicated periodic patterns intelligently requires further investigation.

2.3 Positional Encodings of Transformers

Numerous positional encoding schemes have been proposed to introduce positional information in Transformers. The vallina positional encoding [25] is a map $f : \mathbb{N} \to \mathbb{R}^d$ from a position index to an embedding vector and added to value embeddings directly. It enables to distinguish the data points on different positions. To investigate the relative positional information in sequence, [4, 20] propose to encode the position difference into Transformers, namely relative positional encodings. Recently, [22] introduces a rotary position encoding scheme which is the most related work compared to ours. It encodes the relative distance of two positions with an orthogonal matrix. Unfortunately, existing schemes can not capture the basic characteristics of time series data and are globally shared by all input series. In this paper, we encode the period and phase information specifically to address this issue.

3 PRELIMINARIES

In this section we briefly introduce the canonical attention and quaternions as preliminaries for further discussion.

3.1 Canonical Attention

Given an input sequence of tokens $\mathcal{X}$ with length N, the canonical dot-product attention [25] computes

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right)\boldsymbol{V},$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N\times d}$ are representations of $\mathcal{X}$, each of which are interpreted as queries, keys and values, respectively, d is the hidden dimension, and $\sqrt{d}$ is a scaling factor. In practice, the multi-head variant of attention, which performs the attention mechanism h times in parallel, is commonly used. Throughout this paper, we omit h for simplicity.

In a more compact matrix form, dot-product attention can be written as

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V},\ \ \boldsymbol{A} = \exp(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d}),\ \ \boldsymbol{D} = \mathrm{diag}(\boldsymbol{A}\boldsymbol{1}_N),$$

where $\exp(\cdot)$ is applied element-wise, $\boldsymbol{1}_N$ is the all-ones vector of length N, and $\mathrm{diag}(\cdot)$ is a diagonal matrix with the input vector as the diagonal. Elements of the attention matrix $\boldsymbol{A}$ take the form $\boldsymbol{A}(m, n) = \mathrm{SM}(\boldsymbol{Q}_m, \boldsymbol{K}_n)$, where $\boldsymbol{Q}_m/\boldsymbol{K}_n$ stands for the $m-\mathrm{th}/n-\mathrm{th}$ row in $\boldsymbol{Q}/\boldsymbol{K}$, and SM is the *softmax-kernel* : $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ defined as

$$\mathrm{SM}(\boldsymbol{x}, \boldsymbol{y}) \overset{\mathrm{def}}{=} \exp(\boldsymbol{x}^\top \boldsymbol{y}),$$

which measures the similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$.

3.2 Quaternions

As our proposed attention mechanism using quaternions, we make a brief introduction of quaternion first. Quaternions are extension of complex numbers [7]. A quaternion is represented in the form $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where a, b, c, d are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are imaginary units. The set of quaternions has a basis $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$, and multiplication of basis elements is

$$\mathbf{i}1 = 1\mathbf{i} = \mathbf{i}, \qquad \mathbf{j}1 = 1\mathbf{j} = \mathbf{j}, \qquad \mathbf{k}1 = 1\mathbf{k} = \mathbf{k},$$
$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1,$$
$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \qquad \mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \qquad \mathbf{ki} = -\mathbf{ik} = \mathbf{j}.$$

Similar to complex numbers, a is called the real part of quaternion q and the conjugation of q is $q^* = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k}$, and its norm is defined as $\|q\| = \sqrt{qq^*} = \sqrt{a^2 + b^2 + c^2 + d^2}$. A quaternion with norm 1 is called a unit quaternion. Our learning-to-rotate attention utilizes two key properties of quaternions, i.e., inner product with quaternions and rotation with unit quaternions.

Inner Product with Quaternions. Given two real-valued vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, we have

$$\boldsymbol{x}^\top \boldsymbol{y} = \mathrm{Re}[\tilde{\boldsymbol{x}}^{\mathsf{H}}\tilde{\boldsymbol{y}}],$$

where $\mathrm{Re}[\cdot]$ is the real part of a quaternion, $^{\mathsf{H}}$ stands for conjugate transpose of a quaternion vector, and $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ are $d/4$-dimensional quaternion vectors. We call $\tilde{\boldsymbol{v}}$ the *quaternion form* of the real-valued vector $\boldsymbol{v}$, which is defined as

$$\tilde{\boldsymbol{v}} \overset{\mathrm{def}}{=} (\boldsymbol{v}_1; \boldsymbol{v}_2; \boldsymbol{v}_3; \boldsymbol{v}_4)(1; \mathbf{i}; \mathbf{j}; \mathbf{k})^\top, \qquad \boldsymbol{v} = \left[\boldsymbol{v}_1^\top; \boldsymbol{v}_2^\top; \boldsymbol{v}_3^\top; \boldsymbol{v}_4^\top\right]^\top.$$

Here, $\boldsymbol{v}$ is split into four equal-length parts $\boldsymbol{v}_{1\sim 4} \in \mathbb{R}^{\frac{d}{4}}$, and each part is treated as the coefficient of a quaternion basis of $\tilde{\boldsymbol{v}}$. Thus, rewriting the inner product into a quaternion form, *softmax-kernel* can be computed as

$$\mathrm{SM}(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\boldsymbol{x}^\top \boldsymbol{y}\right) = \exp\left(\mathrm{Re}[\tilde{\boldsymbol{x}}^{\mathsf{H}}\tilde{\boldsymbol{y}}]\right).$$

Rotation with Unit Quaternions. For a 3-dimensional unit vector $\boldsymbol{u} = (u_\mathrm{x}, u_\mathrm{y}, u_\mathrm{z})$, the extended Euler's formula $q = e^{\theta(u_\mathrm{x}\mathbf{i} + u_\mathrm{y}\mathbf{j} + u_\mathrm{z}\mathbf{k})}$ $= \cos\theta + \sin\theta(u_\mathrm{x}\mathbf{i} + u_\mathrm{y}\mathbf{j} + u_\mathrm{z}\mathbf{k})$ induces a unit quaternion that describes a rotation in 4-dimensional Euclidean space. In learning-to-rotate attention, we utilize the rotation nature of unit quaternions to endow series with period and phase information.

4 METHODOLOGIES

The goal of time series forecasting is to make next O-length predictions given previous I-length observations. In this section, we introduce our proposed framework, *Quaternion Transformer*, which succeeds in handling complicated periodical patterns and breaking the bottleneck of computational efficiency in long time series forecasting. The framework is illustrated in Fig. 2, wherein the learning-to-rotate attention (LRA) with quaternion is designed to model intricate periodical dependencies of time series, and the trend normalization is similar to layer normalization, but has an emphasis on slowly varying trends of series representations in hidden layers. In addition, we propose a decoupling attention to reduce the quadratic complexity of LRA to linear.

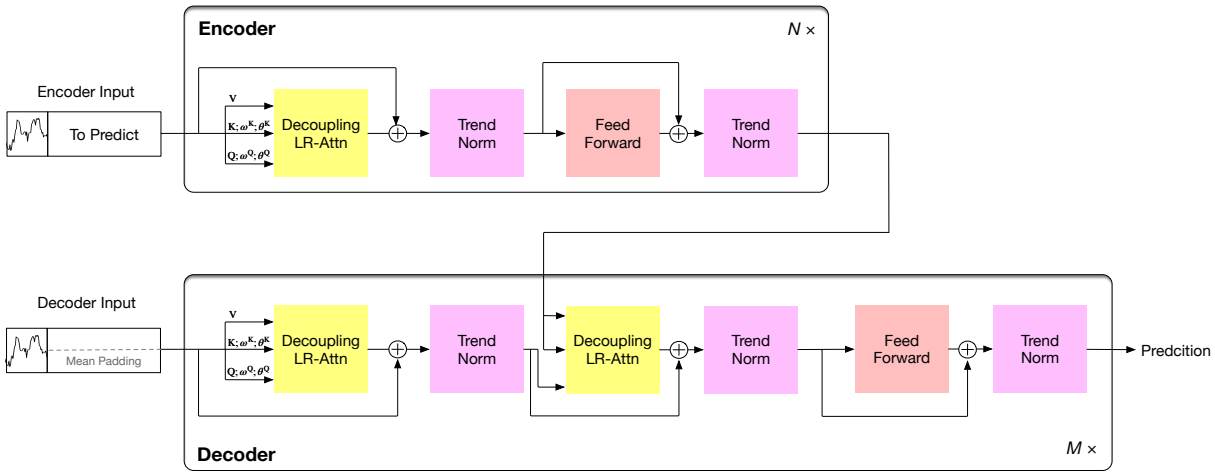


Figure 2: Quaternion Transformer Architecture. Quaternion Transformer follows the encoder-decoder architecture of canonical Transformer. The observed time series is input to the encoder, and the latter part of the input series concatenated with mean value of input series (which is a placeholder of target series) are input into the decoder. The encoder has N identical layers composed of two sublayers: decoupling learning-to-rotate self-attention, which can capture complicated periodical temporal patterns of time series, and position-wise feed-forward network. Both of them are followed by trend normalization. The decoder has M identical layers, which inserts a third sub-layer, performing learning-to-rotate cross-attention over the output representations of the encoder.

4.1 Learning-to-Rotate Attention

The canonical attention calculates the similarity between two tokens based on their inner product, and numerous positional embedding schemes have been proposed to incorporate the position information. However, existing positional embedding methods focus on absolute positions of the tokens and neglect the periodic patterns. We aim to design a novel mechanism to measure the similarity on time series that could capture complicated temporal patterns such as multiple periods, variable periods, and phase shifts.

Let us start from the simplest periodic time series and show the ideal properties of a desirable kernel function to preserve superiority of the canonical attention and describe periodical patterns. Given a d-dimensional time series $TS = [\cdots, \boldsymbol{x}, \cdots, \boldsymbol{y}, \cdots]$ with a single fixed period $T \in \mathbb{N}^+$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ are located at the m-th and n-th time step, and $\omega = 1/T$ is the corresponding frequency, we seek to find an enhanced softmax-kernel $\mathrm{SM}'(\phi(\boldsymbol{x}, m), \psi(\boldsymbol{y}, n))$ to measure the similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$, where *(quaternion) functions* $\phi(\cdot, m)$ and $\psi(\cdot, m)$ involve period and phase information of positions m and n specific to TS. Formally, we wish this new kernel to meet three properties: boundedness, initial-value invariance, and period-independent and phase-dependent translation.

Property 1. ***Boundedness:*** *$\forall\, \boldsymbol{x}, \boldsymbol{y}, m, n$, $\frac{\|\phi(\boldsymbol{x},m)\|}{\|\boldsymbol{x}\|} \le C$, $\frac{\|\psi(\boldsymbol{y},n)\|}{\|\boldsymbol{y}\|} \le C$, where $C \in \mathbb{R}^+$ is a constant. Further, we call ϕ and ψ* ***Norm-invariant*** *functions if $\|\phi(\boldsymbol{x}, m)\| = \|\boldsymbol{x}\|$ and $\|\psi(\boldsymbol{y}, n)\| = \|\boldsymbol{y}\|$.*

The *Boundedness* property ensures that $|\phi(\boldsymbol{x}, m)^{\mathsf{H}}\psi(\boldsymbol{y}, n)| \le \|\phi(\boldsymbol{x}, m)\|\|\psi(\boldsymbol{y}, n)\| \le C^2\|\boldsymbol{x}\|\|\boldsymbol{y}\|$, and thus the stability of the inner product can be guaranteed.

Property 2. ***Initial-value invariance:*** *$\forall\, \boldsymbol{x}, \boldsymbol{y}$, given $m, n = 0, 0$, there exist $\phi(\boldsymbol{x}, 0) = \boldsymbol{x}$, $\psi(\boldsymbol{y}, 0) = \boldsymbol{y}$, and $\mathrm{SM}'(\phi(\boldsymbol{x}, 0), \psi(\boldsymbol{y}, 0)) = \mathrm{SM}(\boldsymbol{x}, \boldsymbol{y})$.*

The *initial-value invariance* property makes the new kernel consistent with the original softmax-kernel at initial position, and thus preserves the the superiority of softmax-kernel.

Next, we are going to incorporate the periodical nature of time series into the designed kernel. Translate $\boldsymbol{x}, \boldsymbol{y}$ at time steps m, n to $m + O_1, n + O_2$, i.e., $(\boldsymbol{x}, m) \xrightarrow{\text{offset}} (\boldsymbol{x}, m + O_1)$, and $(\boldsymbol{y}, n) \xrightarrow{\text{offset}} (\boldsymbol{y}, n + O_2)$, and write O_1, O_2 into $O_1 = k_1T + s_1, O_2 = k_2T + s_2$, where T is the period of the given time series, $k_{1,2}, s_{1,2} \in \mathbb{N}$, and $s_{1,2} \in [0, T)$ indicating phase offsets. We define a set of translation functions $\{\mathrm{Trans}_{O_1,O_2} | O_1, O_2 \in \mathbb{N}\}$ that

$$\mathrm{Trans}_{O_1,O_2}[\mathrm{SM}'(\phi(\boldsymbol{x}, m), \psi(\boldsymbol{y}, n))] = \mathrm{SM}'(\phi(\boldsymbol{x}, m{+}O_1), \psi(\boldsymbol{y}, n{+}O_2)).$$

Property 3. ***Period-independent and phase-dependent translation:*** *$\mathrm{SM}'(\cdot)$ is said to be a period translation-independent and phase translation-dependent function, if there exists a set of translation functions Trans_{O_1,O_2} such that for any $O_1, O_2 \in \mathbb{N}$, $\mathrm{Trans}_{O_1,O_2} = \mathrm{Trans}_{s_1,s_2}$, where Trans_{O_1,O_2}, s_1, and s_2 are defined above.*

Property 3 indicates that given $\boldsymbol{x}, \boldsymbol{y}$, their similarity measured by $\mathrm{SM}'(\cdot)$ only depends on their phases of positions.

We claim that a kernel satisfying Properties 1, 2 and 3 is a better choice to describe periodical time series, as it combines the powerful expressive capacity of dot-product attention and the periodical characteristic of time series. And we derive such a kernel named *rotatory softmax-kernel* with quaternions as

$$\begin{aligned} \mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x}, m), \psi(\boldsymbol{y}, n)) &= \exp(\mathrm{Re}[\phi(\boldsymbol{x}, m)^{\mathsf{H}}\psi(\boldsymbol{y}, n)]), \\ \phi(\boldsymbol{x}, m) = \tilde{\boldsymbol{x}}e^{\mathbf{i}2\pi\omega m}, &\quad \psi(\boldsymbol{y}, n) = \tilde{\boldsymbol{y}}e^{\mathbf{j}2\pi\omega n}, \end{aligned} \tag{1}$$

where $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ are the *quaterion forms* (defined in Section 3.2) of $\boldsymbol{x}, \boldsymbol{y}$, $\omega = 1/T$ is the frequency of the time series, $e^{\mathbf{i}2\pi\omega m}, e^{\mathbf{j}2\pi\omega n}$ are unit quaternions representing rotations with angular frequency ω. Notice that ϕ and ψ are two different rotation functions implying rotations on two different planes, as $\boldsymbol{x}$ and $\boldsymbol{y}$ are asymmetric in dot-product attention.

Here we just give an intuitive proof that $\mathrm{SM}^{\mathrm{rot}}$ satisfies Properties 1, 2 and 3, and the formal proof is in Appendix A.2.First, $e^{\mathbf{i}2\pi\omega\cdot}, e^{\mathbf{j}2\pi\omega\cdot}$ are unit quaternions that do not change the norms of $\boldsymbol{x}, \boldsymbol{y}$ (Property 1). Second, $e^{\mathbf{i}2\pi\omega 0} = e^{\mathbf{j}2\pi\omega 0} = 1$ (Property 2). Finally, $\phi(\cdot, m)$ is a periodic function w.r.t. m, i.e., $\phi(\boldsymbol{x}, m+T) = \tilde{\boldsymbol{x}}e^{\mathbf{i}2\pi\frac{1}{T}(m+T)} = \tilde{\boldsymbol{x}}e^{\mathbf{i}2\pi\omega m} = \phi(\boldsymbol{x}, m)$, and so does $\psi(\cdot, n)$. Thus, $\mathrm{SM}^{\mathrm{rot}}$ is not affected by period-translation (Property 3). Though the rotations of *rotatory softmax-kernel* are performed in the 4D Euclidean space, a simplified 3D illustration is in Fig. 3.

Learning-to-Rotate Attention. Our learning-to-rotate attention (LRA) relies on $\mathrm{SM}^{\mathrm{rot}}$ as the kernel function. Next we show LRA is realized. Real-world time series usually have more complicated periodicity patterns than a single fixed period. We learn several frequencies and phases in a data-driven manner with convolutional neural networks.

Assume we have a N-length query series $\mathcal{X}$ and a M-length key-value series $\mathcal{Y}$. Firstly the original $\mathcal{X}$ and $\mathcal{Y}$ are projected onto the representation space, i.e., $\boldsymbol{Q} = \mathcal{X}W^{\mathrm{Q}} \in \mathbb{R}^{N\times d}$, $\boldsymbol{K} = \mathcal{Y}W^{\mathrm{K}} \in \mathbb{R}^{M\times d}$ and $\boldsymbol{V} = \mathcal{Y}W^{\mathrm{V}} \in \mathbb{R}^{M\times d}$. Then learning-to-rotate attention $\mathcal{H} = \mathrm{LR\text{-}Attn}(\mathcal{X}, \mathcal{Y})$ maps query series to outputs $\mathcal{H}$ using key-value series. For the single head case, the process is as follows:

Frequency/phase-Generation:

$$\begin{pmatrix} \omega_1^{\mathrm{Q}} \\ \cdots \\ \omega_P^{\mathrm{Q}} \end{pmatrix}, \begin{pmatrix} \theta_1^{\mathrm{Q}} \\ \cdots \\ \theta_P^{\mathrm{Q}} \end{pmatrix} = \mathrm{Conv}(\boldsymbol{Q}; W_\omega^{\mathrm{Q}}), \mathrm{Conv}(\boldsymbol{Q}; W_\theta^{\mathrm{Q}}),$$

$$\begin{pmatrix} \omega_1^{\mathrm{K}} \\ \cdots \\ \omega_P^{\mathrm{K}} \end{pmatrix}, \begin{pmatrix} \theta_1^{\mathrm{K}} \\ \cdots \\ \theta_P^{\mathrm{K}} \end{pmatrix} = \mathrm{Conv}(\boldsymbol{K}; W_\omega^{\mathrm{K}}), \mathrm{Conv}(\boldsymbol{K}; W_\theta^{\mathrm{K}}),$$

Series-Rotation:

$$\begin{aligned} \Phi_p(\boldsymbol{Q}, \mathrm{pos}^{\mathrm{Q}}) &= \tilde{\boldsymbol{Q}}e^{\mathbf{i}(2\pi\omega_p^{\mathrm{Q}}\mathrm{pos}^{\mathrm{Q}} + \theta_p^{\mathrm{Q}})}, \quad p = 1, 2, \cdots, P \\ \Psi_p(\boldsymbol{K}, \mathrm{pos}^{\mathrm{K}}) &= \tilde{\boldsymbol{K}}e^{\mathbf{j}(2\pi\omega_p^{\mathrm{K}}\mathrm{pos}^{\mathrm{K}} + \theta_p^{\mathrm{K}})}, \quad p = 1, 2, \cdots, P \end{aligned} \tag{2}$$

Series-Attention with rotatory softmax-kernel :

$$\boldsymbol{S} = \mathrm{softmax}\left(\frac{1}{P\sqrt{d}}\sum_{p=1}^{P}\mathrm{Re}[\Phi_p(\boldsymbol{Q}, \mathrm{pos}^{\mathrm{Q}})\Psi_p(\boldsymbol{K}, \mathrm{pos}^{\mathrm{K}})^{\mathsf{H}}]\right),$$

Series-Aggregation:

$$\mathcal{H} = \boldsymbol{S}\boldsymbol{V}.$$

Here, we hypothesize the series have P periods, and P is a hyperparameter (P is small, e.g., 2 and 3 to meet the characteristics of real-world time series). In *frequency/phase-generation* step, we utilize 1D convolutions with activation ReLU to generate P latent frequencies $\omega_{1\sim P}^{\mathrm{Q}} \in [0, +\infty)^{N\times 1}$ ($\omega_{1\sim P}^{\mathrm{K}}$ is similar). Convolutions can effectively capture local contexts of each time step to generate reliable latent frequencies, and these latent frequencies are not identical at each time step implying variable periods. Moreover, to account for phase shifts, we additionally generate P latent phases $\theta_{1\sim P}^{\mathrm{Q}} \in (-\pi, \pi)^{N\times 1}$ using 1D convolutions with activation $\pi \cdot \tanh$ ($\theta_{1\sim P}^{\mathrm{K}}$ is similar). In *series-rotation* step, we rotate the representations at each time step according to the learned latent frequencies and phases in the previous step. Each row vector of $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{K}}$ is in the *quaternion form* (defined in section 3.2) of the corresponding row vector of $\boldsymbol{Q}, \boldsymbol{K}$, and $\mathrm{pos}^{\mathrm{Q}} = [0, 1, 2, \cdots, N-1]^\top/N, \mathrm{pos}^{\mathrm{K}} = [0, 1, 2, \cdots, M-1]^\top/M$ are position vectors of series Q and K, respectively. In *series-attention* step, to integratedly capture position-wise simialrity under multiple periods, the unnormalized similarity is the mean of quaternion dot-product under multiple rotations. Finally, in *series-aggregation* step, the outputs $\mathcal{H} \in \mathbb{R}^{N\times d}$ is generated using the softmax-normalized similarity. In practice, we use the *multi-head* variant of LRA, and will not go into details here, as it can be derived quite directly. Notice that, LRA is more expressive than canonical dot-product attention. When $P = 1$, $\omega = 0$ and $\theta = 0$, LRA degenerates into canonical attention.

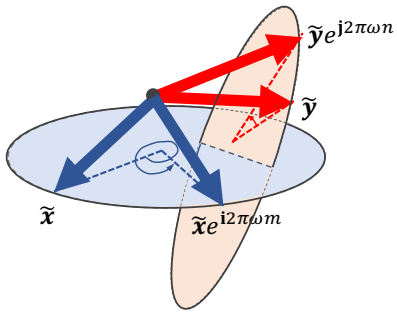


Figure 3: A 3D illustration of *rotatory softmax-kernel*. *Rotatory softmax-kernel* represents embeddings in quaternion form, and rotate them with angular frequency ω. Thus, embeddings at different phase can be discriminated. Finally, the similarity is measured by the exponential dot-product of embeddings after rotations.

Frequency and Phase Regularization. We propose to regularize the latent frequencies and phases to mitigate the overfitting problem, and the regularization terms of a single layer of LRA is defined as

$$\begin{aligned} \mathcal{L}_\omega &= \frac{1}{P(N-1)}\sum_{p=1}^{P}\sum_{n=0}^{N-2}\left(\omega_p^{(n+1)} - \omega_p^{(n)}\right)^2, \\ \mathcal{L}_\theta &= \frac{1}{PN}\sum_{p=1}^{P}\sum_{n=0}^{N-1}\left|\theta_p^{(n)}\right|. \end{aligned} \tag{3}$$

We omit the superscripts Q, K here for simplicity, and $\omega_p^{(n)}, \theta_p^{(n)}$ are the n-th entry of ω_p, θ_P, respectively. Specifically, $\mathcal{L}_\omega$ is the l_2-norm of the difference of ω, which indicates that the frequency (or period) of series varies slowly, and $\mathcal{L}_\theta$ is the l_1-norm of θ, which indicates that the phase shifts are sparse in time series. Thus, the final objective of Quaternion Transformer is[1]

$$\mathcal{L} = \mathcal{L}_{\mathrm{pred}} + \lambda_1\mathcal{L}_\omega + \lambda_2\mathcal{L}_\theta, \tag{4}$$

[1]We slightly abuse the notations here, and $\mathcal{L}_\omega$, $\mathcal{L}_\theta$ indicate the sum of regularizations of all layers in the model rather than a single attention layer.

where $\mathcal{L}_{\text{pred}}$ is the prediction loss, e.g., mean absolute error (MAE) and mean squared error (MSE), and $\lambda_1, \lambda_2 > 0$ are hyperparameters.

4.2 Trend Normalization

Normalization techniques [1, 10] are widely used in deep neural networks to normalize the activations of the neurons in order to ensure stable gradients during the training procedure. These methods usually work in the form of $\frac{\boldsymbol{\gamma}}{\sigma} \odot (\mathcal{X} - \mu) + \boldsymbol{\beta}$, re-centering and re-scaling the activations to a desired distribution, where $\mathcal{X}$ is the original activations, μ and σ are the mean and standard deviation of $\mathcal{X}$ over some dimension(s), and $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are learnable parameters to shift and scale the zero-score normalized activations.

Generic normalization techniques do not rely on any time series-specific knowledges. A typical inductive bias is that the trend of time series is a slowly varying function at the most of time, while the high-dimensional series of the hidden layers in neural networks might fluctuate violently due to the unstable training process. To mimic this slowly varying behaviour in hidden layers, we introduce *trend normalization* inspired by series decomposition techniques, which performs re-trending and re-scaling on the series representations with polynomial basis. For length-N series $\mathcal{X} \in \mathbb{R}^{N \times d}$, the form of trend normalization is:

$$\begin{aligned} &\frac{\boldsymbol{\gamma}}{\sigma} \odot (\mathcal{X} - \text{MovingAvg}(\mathcal{X})) + \mathcal{T}, \\ &\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathcal{X}_i - \mu)^2}, \quad \mu = \frac{1}{N}\sum_{i=1}^{N}\mathcal{X}_i, \\ &\mathcal{T} = \sum_{i=0}^{p} \boldsymbol{\beta}_i \text{pos}^i, \end{aligned} \tag{5}$$

where $\mathcal{X}_i \in \mathbb{R}^d$ is the i-th time step of the series, $\mu, \sigma \in \mathbb{R}^d$ are the mean and standard deviation of $\mathcal{X}$ over the time dimension, $\boldsymbol{\gamma} \in \mathbb{R}^d$ is the learnable scaling factors, $\text{MovingAvg}(\cdot)$ calculates the mean in a temporal window at each time step (necessary padding is performed to make sure invariant sequence length after moving average), $\mathcal{X} - \text{MovingAvg}(\mathcal{X})$ can detrend the original series, $\mathcal{T} \in \mathbb{R}^{N \times d}$ is the learnable trend using polynomial functions, $\text{pos} = [0, 1, 2, \cdots, N-1]^\top / N$ is the position vector of the series, pos^i is the power of pos, p is the maximum degree of the polynomial function, $\boldsymbol{\beta} \in \mathbb{R}^{p \times d}$ is the learnable coefficients of the polynomial function, and $\boldsymbol{\beta}_i$ is the i-th row, correspondence to the i-th degree coefficient.

The re-scaling step in trend normalization is similar to other normalization techniques, while the re-trending step is introduced in place of re-centering. It first detrends the original series and then assigns learnable trends approximated by slowly varying polynomial functions to the detrended series.

4.3 Decoupling Attention with Global Memory

A potential problem when using LRA for modeling extremely long time series is the quadratic complexity $O(N^2)$. Notice the fact that similar temporal patterns occur repeatedly throughout the entire time series, which provides the potential to compress the long time series into compact representations. Thus, we introduce a momentum-updated c-length latent series $\mathcal{M} \in \mathbb{R}^{c \times d}$, which is used to encode the global temporal patterns throughout mini-batches, and decouple learning-to-rotate attention $\mathcal{H} = \text{LR-Attn}(\mathcal{X}, \mathcal{Y})$ into:

$$\begin{aligned} \mathcal{H} &= \text{LR-Attn}(\mathcal{X}, \mathcal{M}') \in \mathbb{R}^{N \times d}, \\ \mathcal{M}' &= \text{LR-Attn}(\mathcal{M}, \mathcal{Y}) \in \mathbb{R}^{c \times d}. \end{aligned} \tag{6}$$

The decoupling attention first transforms $\mathcal{M}$ to $\mathcal{M}'$ by attending to $\mathcal{Y}$, and then $\mathcal{M}'$ is attended to by $\mathcal{X}$ to produce the output $\mathcal{H}$. The underlying thought is that the long time series can be compressed to compact ones guided by the global patterns. Note that, Eq. (6) computes attention between two series of length c and L, and thus the computation complexity of decoupling attention is $O(2cN)$, where c is a hyperparameter relying on the characteristics of the time series dataset, but independent of the input sequence length N. Therefore, decoupling attention could be considered as linear complexity.

The next problem is how to obtain the latent series $\mathcal{M}$. Two straightforward ways are taking $\mathcal{M}$ as fixed embeddings or learnable parameters. However, fixed $\mathcal{M}$ is less expressive, and learnable $\mathcal{M}$ jointly updated with the model parameters is overly restricted by the final objective and might lose the capacity of encoding global temporal patterns in the entire dataset. To address this issue, we propose a momentum-updating mechanism in a mini-batch fashion:

$$\mathcal{M} \leftarrow \alpha \mathcal{M} + (1 - \alpha)\frac{1}{B}\sum_{i=1}^{B}\mathcal{M}'_i, \tag{7}$$

where $\alpha \in [0, 1)$ is the momentum coefficient (a large α, e.g., 0.999, makes $\mathcal{M}$ evolve slowly such that it can maintain long-term memory throughout mini-batches), B is the number of samples in a mini-batch, and $\mathcal{M}'_i$ is the intermediate representation of the i-th sample in the batch, which encodes the information of individual series, and is then merged into the global memory. In Section 5.2, we empirically verify the superiority of our proposed momentum-updating mechanism. Besides, we illustrate the architectures of different attentions in Appendix A.1 to highlight their difference.

Relations to Related Models. Set Transformer [15] and Luna [17] also introduce the extra memory and perform dual attentions to reduce the computation complexity of attention. The main difference is the learning procedure of the extra introduced memory. Specifically, the *inducing points* in Set Transformer are learnable parameters, which is suboptimal when modeling time series as we discussed above, and the *extra input sequence* in Luna is designed to capture the contextual information of the input sequence rather than the global information of the entire dataset. Differently, our proposed decoupling attention uses a momentum-updating mechanism to learn the latent series to maintain the global memory of temporal patterns. Moreover, it utilizes LRA as basic blocks, which is specially designed for modeling periodical time series.

5 EXPERIMENTS

In this section, we compare the performances of the proposed framework with other state-of-the-art time series forecasting models. We also perform ablation studies and analyze how each component in the framework contributes to the final accurate results, and discuss the insights of our model.

Datasets. We perform empirical studies on six widely used benchmark datasets as follows: (1) *ETT-small* [31] collects 2-year load and oil temperature data from 2 separate electricity transformers. *ETTh*

Table 1: Multivariate time series forecasting results on six benchmark datasets with input length $I = 96$ and prediction length $O \in \{96, 192, 336, 720\}$. A lower MSE and MAE indicates better performance. The best results are highlighted in bold.

Models		Quatformer		Quatformer†		Autoformer		Informer		LogTrans		Reformer		LSTM		TCN	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh	96	**0.403**	**0.434**	0.426	0.450	0.442	0.451	1.105	0.845	0.786	0.780	0.648	0.619	1.154	0.853	0.713	0.650
	192	**0.444**	**0.463**	0.453	0.466	0.500	0.482	1.014	0.780	0.974	0.744	1.078	0.827	1.219	0.886	0.960	0.781
	336	**0.452**	**0.455**	0.464	0.474	0.512	0.492	1.289	0.928	1.250	0.882	1.299	0.841	1.213	0.884	0.832	0.714
	720	0.477	0.490	**0.474**	**0.487**	0.514	0.512	1.185	0.872	1.406	0.968	2.415	1.520	1.223	0.895	0.965	0.787
ETTm	96	0.220	0.301	**0.217**	**0.297**	0.247	0.325	0.403	0.494	0.495	0.524	0.312	0.402	1.605	0.967	0.812	0.662
	192	0.279	0.333	**0.269**	**0.329**	0.278	0.335	0.807	0.696	1.260	0.886	0.348	0.433	2.227	1.115	0.776	0.676
	336	0.331	**0.354**	**0.330**	0.366	0.336	0.370	1.375	0.912	1.571	1.004	0.350	0.433	2.465	1.172	1.083	0.790
	720	**0.422**	**0.413**	0.433	0.428	0.439	0.435	3.752	1.475	3.318	1.399	2.631	1.242	2.761	1.262	2.342	1.220
Weather	96	**0.211**	**0.279**	0.213	0.287	0.259	0.332	0.315	0.380	0.369	0.441	0.689	0.596	0.289	0.348	0.248	0.329
	192	**0.263**	**0.325**	0.265	0.326	0.300	0.359	0.395	0.431	0.671	0.595	0.752	0.638	0.344	0.390	0.294	0.371
	336	**0.310**	**0.344**	0.315	0.354	0.364	0.401	0.457	0.454	0.876	0.689	0.639	0.596	0.435	0.451	0.428	0.459
	720	**0.381**	**0.374**	0.382	0.378	0.439	0.440	0.706	0.607	0.992	0.744	1.130	0.792	0.475	0.481	0.440	0.455
Exchange	96	**0.147**	**0.274**	0.148	0.276	0.154	0.284	0.948	0.797	0.615	0.625	1.065	0.829	1.104	0.906	2.037	1.110
	192	**0.254**	**0.364**	0.255	0.365	0.272	0.381	1.237	0.894	1.360	0.983	1.188	0.906	1.555	1.074	1.912	1.141
	336	0.427	0.481	**0.425**	**0.480**	0.461	0.509	1.568	1.095	1.304	0.982	1.357	0.976	1.016	1.156	1.862	1.125
	720	**0.974**	**0.751**	1.095	0.800	1.100	0.813	2.696	1.357	0.974	0.804	1.510	1.016	2.422	1.308	3.339	1.523
Traffic	96	0.618	**0.384**	**0.617**	0.387	0.636	0.397	0.633	0.510	0.653	0.353	0.732	0.423	0.855	0.467	0.815	0.456
	192	0.619	0.384	**0.600**	**0.367**	0.618	0.381	0.780	0.426	0.658	0.355	0.733	0.420	1.010	0.545	0.831	0.471
	336	0.622	**0.384**	**0.618**	0.385	0.626	0.388	0.779	0.429	0.669	0.359	0.742	0.420	1.346	0.743	0.846	0.474
	720	0.629	0.383	**0.616**	**0.379**	0.653	0.400	0.942	0.521	0.674	0.361	0.755	0.423	1.495	0.801	0.815	0.493
Electricity	96	**0.197**	**0.308**	0.200	0.311	0.203	0.318	0.350	0.424	0.296	0.379	0.312	0.402	0.419	0.463	0.356	0.424
	192	**0.205**	**0.302**	0.216	0.330	0.233	0.338	0.340	0.419	0.309	0.385	0.348	0.433	0.461	0.489	0.361	0.431
	336	**0.220**	**0.329**	0.228	0.343	0.259	0.359	0.374	0.446	0.322	0.392	0.350	0.433	0.591	0.572	0.373	0.442
	720	0.245	0.350	**0.242**	**0.349**	0.255	0.361	0.356	0.425	0.319	0.382	0.340	0.420	0.892	0.760	0.367	0.430

(a) Learning-to-Rotate Attention **(b) Canonical Attention** **(c) Auto-Correlation**

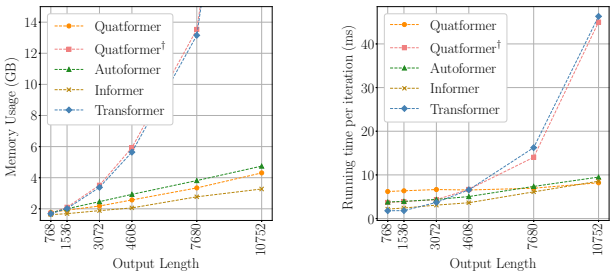
Figure 4: Visualization of Learned Dependencies. We select the top-6 similar points (marked in green points) w.r.t. the query (marked in orange stars) at the last point of the time series.

and *ETTm* were sampled 1-hourly and 15 minutely respectively. (2) *Weather*[2] records 21 meteorological measurements, e.g. air temperature, wind velocity, and etc. for year 2020 at 10 minutes granularity. (3) *Exchange Rate* [14] contains the daily exchange rates of eight foreign countries including Australia, British, Canada, Switzerland, China, Japan, New Zealand and Singapore ranging from 1990 to 2016. (4) *Traffic*[3] curates 2-year road occupancy rates measured by 862 sensors deployed in San Francisco Bay Area freeways. (5) *Electricity*[4] includes hourly electricity consumption data for 321 customers from 2012 to 2014.

[2]https://www.bgc-jena.mpg.de/wetter/
[3]https://pems.dot.ca.gov/
[4]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

Baselines. The baselines including: four state-of-the-art transformer based models: Autoformer [29], Informer [31], LogTrans [16], Reformer [13], one RNN based seq2seq model: LSTM [8, 23] and one CNN based model: TCN [2].

Implementation Details. Quatformer stands for our proposed Quaternion Transformer, and Quatformer† for Quaternion Transformer without Decoupling Attention. The regularization coefficients λ_1, λ_2 are chosen from $\{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, the momentum coefficient α is chosen from $\{0.99, 0.999, 0.9999\}$, and the number of latent periods P is chosen from $\{1, 2, 4\}$ using grid search. The length of latent series in decoupling attention $c = 96$. For all the models, we fix the input length to be 96 and select output length from $\{96, 192, 336, 720\}$. Models are trained using L_2 loss with ADAM [12] optimizer, accompanied with initial learning rate



(a) Memory Efficiency **(b) Running Time Efficiency**

Figure 5: Computational Efficiency Analysis. It is clear that Quatformer **has linear complexity, while** Quatformer† **still subjects to quadratic complexity.**

10^{-4}. All experiments are iterated 5 times. We use mean squared error (MSE) and mean absolute error (MAE) as *evaluation metrics*, defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\text{Y} - \hat{\text{Y}}|, \quad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{Y} - \hat{\text{Y}})^2, \tag{8}$$

where $\text{Y}, \hat{\text{Y}}$ are ground truth and predictions, respectively.

5.1 Performance Comparisons

As shown in Table 1, for multivariate time series forecasting, Quaternion Transformer achieves the best performance in all benchmarks consistently, and it yields an overall 8.1% MSE improvement and up to 18.5% MSE improvement compared with Autoformer, the second best model, among all datasets and all settings. Besides, Quatformer achieves a comparable or even better performance with Quatformer† implying that decoupling attention not only reduces the computational complexity (discussed in Section 5.2), but also retains the advantages of prediction accuracy. Furthermore, we can also find that the performance of Quatformer changes quite steadily as the prediction length O increases, implying its scalability in long-term time series forecasting. In addition, we report results of univariate time series forecasting in Appendix A.3

5.2 Ablation Studies and Analysis

We conduct ablation studies on *ETTh* and *Weather*, both of which show complicated periodical patterns, to investigate the components of our proposed framework.

Learning-to-Rotate Attention vs. Canonical Attention. In this study, we compare our proposed *learning-to-rotate attention* with canonical attention by changing the attention mechanism in Quaternion Transformer. As illustrated in Table 2, *learning-to-rotate attention* shows better performance.

Fig. 4 provides an visualization of the learned dependencies with learning-to-rotate attention, canonical attention, and auto-correlation (used in Autoformer). This time series has a period fluctuated around 24 and some time steps may have phase shifts. Our proposed *learning-to-rotate Attention* captures reliable and robust periodical dependencies, while canonical attention finds less precise similar points, and Autoformer seeks periods too stably and is less flexible in such complicated case.

Table 2: Comparisons of Learning-to-Rotate Attention and Canonical Attention.

Modules		Learning-to-Rotate		Canonical Attention	
		MSE	MAE	MSE	MAE
ETTh	96	**0.403**	**0.434**	0.533	0.526
	192	**0.444**	**0.463**	0.647	0.596
	336	**0.452**	**0.455**	0.713	0.607
	720	**0.477**	**0.490**	0.833	0.696
Weather	96	**0.211**	**0.279**	0.344	0.411
	192	**0.263**	**0.325**	0.416	0.456
	336	**0.310**	**0.344**	0.487	0.502
	720	**0.381**	**0.374**	0.646	0.598

Table 3: Comparisons of Trend Normalization and Layer Normalization.

Modules		TrendNorm		LayerNorm	
		MSE	MAE	MSE	MAE
ETTh	96	**0.403**	**0.434**	1.100	0.776
	192	**0.444**	**0.463**	1.104	0.769
	336	**0.452**	**0.455**	1.118	0.831
	720	**0.477**	**0.490**	1.128	0.857
Weather	96	**0.211**	**0.279**	0.343	0.399
	192	**0.263**	**0.325**	0.403	0.462
	336	**0.310**	**0.344**	0.626	0.514
	720	**0.381**	**0.374**	0.792	0.667

Table 4: Comparisons of Difference Learning Methods of Latent Series in Decoupling Attention.

Methods		Momentum-Update		Fixed-Embeddings		Learnable-Params	
		MSE	MAE	MSE	MAE	MSE	MAE
ETTh	96	**0.403**	**0.434**	0.410	0.436	0.424	0.433
	192	**0.444**	**0.463**	0.455	0.472	0.449	0.470
	336	**0.452**	**0.455**	0.455	0.486	0.469	0.466
	720	**0.477**	**0.490**	0.487	0.513	0.492	0.501
Weather	96	**0.211**	**0.279**	0.240	0.295	0.217	0.309
	192	**0.263**	**0.325**	0.299	0.333	0.303	0.338
	336	**0.310**	**0.344**	0.324	0.369	0.317	0.346
	720	**0.381**	**0.374**	0.402	0.388	0.393	0.380

Trend Normalization vs. Layer Normalization. In this study, we compare our proposed *trend normalization* with layer normalization by changing the normalization layer in Quaternion Transformer. As illustrated in Table 3, *trend normalization* shows better performance, which is because *trend normalization* can restrict the trend of series in hidden layers to vary slowly and avoid erratic fluctuations.

Analysis of Decoupling Attention. In this study, we evaluate the computational efficiency of decoupling attention and discuss how the momentum-updating mechanism work well.

Computational Efficiency. We evaluate the memory efficiency and running time efficiency for Quatformer, Quatformer†, Autoformer, Informer and Transformer (Fig. 5). The output length increases exponentially while the input length is intact as 96, and batch size is set to 1 in order to support extremely long time series. Besides, we run the models 10^3 times to get the mean running time per iteration. All experiment are conducted on on a single

Tesla V100 SXM2 16GB GPU. We can see that decoupling attention (Quatformer) achieves highly promotion of computation efficiency in modeling long time series compared with attentions with quadratic complexity (Quatformer† and Transformer).

Advantage of Momentum-Updating Mechanism. We evaluate different learning methods of the latent series (i.e., global memory), including Momentum-updating mechanism, fixed embeddings, and learnable parameters. For fixed embeddings, we choose the better results of random generated embeddings and fixed positional embeddings proposed in [25]. The results are curated in Table 4, which verifies the effectiveness of momentum-updating mechanism. We conclude that this mechanism help to maintain effective global memory throughout the dataset.

6 CONCLUSION AND FUTURE WORKS

In this paper, we explore how to model the complicated periodical patterns (e.g., multiple periods, variable periods, and phase shifts) commonly exhibited in real-world time series and propose learning-to-rotate attention (LRA), which incorporates period and phase information when performing dot-product attention, accompanied with trend normalization to normalize the series representations in hidden layers with the slowly varying trend. Moreover, to scale LRA to modeling extremely long time series, we design a decoupling attention with global memory which could both reduce the computational complexity and retain the advantages of prediction accuracy. Extensive experiments on real-world data demonstrate that our framework achieve superior forecasting performance. In the current framework, we concentrate on how to leverage the insights of trend and seasonal components of time series. In the future, we plan to explore how to incorporate the residual component (noise) into the model which might help to estimate the uncertainty of forecasting.

REFERENCES

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

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271

[3] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* 6, 1 (1990), 3–73.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* (2019).

[5] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. 2011. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association* 106, 496 (2011), 1513–1527.

[6] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 364–373.

[7] William Rowan Hamilton. 1848. Xi. on quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 33, 219 (1848), 58–60.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (11 1997), 1735–1780.

[9] Robin John Hyndman and George Athanasopoulos. 2018. *Forecasting: Principles and Practice* (2nd ed.). OTexts, Australia.

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[11] Sangeetha Abdu Jyothi, Carlo Curino, Ishai Menache, Shravan Matthur Narayanamurthy, Alexey Tumanov, Jonathan Yaniv, Ruslan Mavlyutov, Inigo Goiri, Subru Krishnan, Janardhan Kulkarni, et al. 2016. Morpheus: Towards Automated {SLOs} for Enterprise Clusters. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 117–134.

[12] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[13] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[14] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. arXiv:1703.07015

[15] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*. 3744–3753.

[16] 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. In *Advances in Neural Information Processing Systems*, Vol. 32.

[17] Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear Unified Nested Attention. *arXiv preprint arXiv:2106.01540* (2021).

[18] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[19] 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.

[20] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155* (2018).

[21] Slawek Smyl. 2020. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting* 36, 1 (2020), 75–85.

[22] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864* (2021).

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. arXiv:1409.3215

[24] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[26] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. 2021. RobustPeriod: Robust Time-Frequency Mining for Multiple Periodicity Detection. In *Proceedings of the 2021 International Conference on Management of Data(SIGMOD)*. 2328–2337.

[27] Qingsong Wen, Zhe Zhang, Yan Li, and Liang Sun. 2020. Fast RobustSTL: Efficient and Robust Seasonal-Trend Decomposition for Time Series with Complex Patterns. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2203–2213.

[28] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in Time Series: A Survey. *arXiv preprint arXiv:2202.07125* (2022).

[29] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. 101–112.

[30] Dawei Zhou, Lecheng Zheng, Yada Zhu, Jianbo Li, and Jingrui He. 2020. Domain adaptive multi-modality neural attention network for financial forecasting. In *Proceedings of The Web Conference 2020*. 2230–2240.

[31] 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 *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 35. 11106–11115.

[32] Yihong Zhou, Zhaohao Ding, Qingsong Wen, and Yi Wang. 2022. Robust Load Forecasting towards Adversarial Attacks via Bayesian Learning. *IEEE Transactions on Power Systems* (2022).

A APPENDIX

A.1 Comparisons of Different Attentions

We illustrate the architectures of different attentions in Fig. 6.

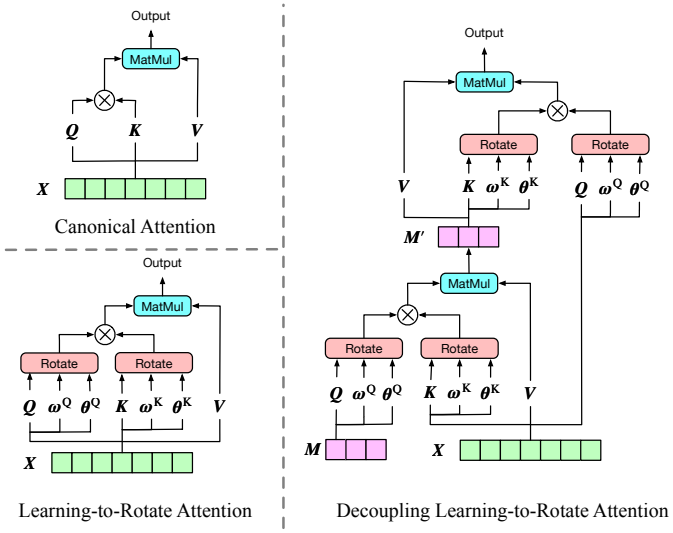


Figure 6: Illustrations of Canonical Attention, Learning-to-Rotate Attention, and Decoupling Learning-to-Rotate Attention. ⊗ stands for measuring similarity with softmax-kernel.

A.2 Proofs in Section 4.1

In this section we will prove that the rotatory softmax-kernel with quaternions

$$\mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x},m),\psi(\boldsymbol{y},n)) = \exp(\mathrm{Re}[\phi(\boldsymbol{x},m)^{\mathsf{H}}\psi(\boldsymbol{y},n)]),$$
$$\phi(\boldsymbol{x},m) = \tilde{\boldsymbol{x}}e^{\mathbf{i}2\pi\omega m}, \quad \psi(\boldsymbol{y},n) = \tilde{\boldsymbol{y}}e^{\mathbf{j}2\pi\omega n},$$

satisfies Properties 1, 2 and 3.

Proof. (*boundedness*) By definition,

$$\|\phi(\boldsymbol{x},m)\| = \|\tilde{\boldsymbol{x}}\| = \|\boldsymbol{x}\| \quad \text{and} \quad \|\psi(\boldsymbol{y},m)\| = \|\tilde{\boldsymbol{y}}\| = \|\boldsymbol{y}\|.$$

(*Initial-value invariance*) It is easy to observe that $\phi(\boldsymbol{x},0) = \tilde{\boldsymbol{x}}$ and $\psi(\boldsymbol{y},0) = \tilde{\boldsymbol{y}}$. Further,

$$\mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x},0),\psi(\boldsymbol{y},0)) = \mathrm{SM}^{\mathrm{rot}}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}}) = \exp(\mathrm{Re}[\tilde{\boldsymbol{x}}^{\mathsf{H}}\tilde{\boldsymbol{y}}]) = \mathrm{SM}(\boldsymbol{x},\boldsymbol{y}).$$

(*Period-independent and phase-dependent translation*)
Since $O_1 = k_1T + s_1$,

$$\begin{aligned}
&e^{\mathbf{i}2\pi\omega(m+O_1)}\\
=&e^{\mathbf{i}(2\pi\omega(m+s_1)+2\pi k_1)}\\
=&\cos(2\pi\omega(m+s_1)+2\pi k_1)+\mathbf{i}\sin(2\pi\omega(m+s_1)+2\pi k_1)\\
=&\cos(2\pi\omega(m+s_1))+\mathbf{i}\sin(2\pi\omega(m+s_1))\\
=&e^{\mathbf{i}2\pi\omega(m+s_1)},
\end{aligned}$$

which implies $\phi(\boldsymbol{x},m+O_1) = \phi(\boldsymbol{x},m+s_1)$ and similarly we have $\psi(\boldsymbol{y},n+O_2) = \phi(\boldsymbol{y},n+s_2)$. Thus for any $\boldsymbol{x}$, $\boldsymbol{y}$, m, n,

$$\begin{aligned}
&\mathrm{Trans}_{O_1,O_2}[\mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x},m),\psi(\boldsymbol{y},n))]\\
=&\mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x},m+O_1),\psi(\boldsymbol{y},n+O_2))\\
=&\mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x},m+s_1),\psi(\boldsymbol{y},n+s_2))\\
=&\mathrm{Trans}_{s_1,s_2}[\mathrm{SM}^{\mathrm{rot}}(\phi(\boldsymbol{x},m),\psi(\boldsymbol{y},n))].
\end{aligned}$$

□

A.3 Univariate Time Series Forecasting

The results of univariate time series forecasting is in Table 5.

Table 5: Univariate time series forecasting results with input length $I = 96$ and prediction length $O \in \{96, 192, 336, 720\}$. A lower MSE and MAE indicates better performance. The best results are highlighted in bold.

Models		Quatformer		Quatformer†		Autoformer		Informer		LogTrans		Reformer		LSTM		TCN	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh	96	**0.064**	**0.195**	0.065	0.195	0.090	0.233	0.193	0.377	0.283	0.468	0.532	0.569	0.230	0.402	0.325	0.509
	192	**0.081**	**0.222**	0.081	0.223	0.103	0.249	0.217	0.395	0.234	0.409	0.568	0.575	0.384	0.547	0.435	0.595
	336	**0.094**	**0.243**	0.096	0.245	0.119	0.275	0.202	0.381	0.386	0.546	0.635	0.589	0.604	0.708	0.396	0.565
	720	**0.091**	**0.240**	0.095	0.245	0.126	0.281	0.183	0.355	0.475	0.628	0.762	0.666	0.666	0.756	0.445	0.611
ETTm	96	**0.075**	**0.206**	0.081	0.214	0.120	0.266	0.080	0.217	0.075	0.208	0.077	0.214	0.174	0.335	0.110	0.262
	192	0.109	0.250	**0.108**	**0.249**	0.143	0.294	0.112	0.259	0.129	0.275	0.138	0.290	0.407	0.516	0.198	0.357
	336	0.134	0.283	**0.133**	**0.281**	0.177	0.326	0.166	0.314	0.154	0.302	0.160	0.313	0.273	0.421	0.230	0.390
	720	**0.173**	**0.325**	0.175	0.328	0.196	0.342	0.228	0.380	0.160	0.322	0.168	0.334	0.392	0.510	0.297	0.439
Weather	96	0.0088	**0.0318**	**0.0036**	0.0320	0.0076	0.0679	0.004	0.044	0.0046	0.052	0.012	0.087	0.005	0.060	0.006	0.066
	192	0.0016	0.0300	**0.0015**	**0.0291**	0.0073	0.0644	0.002	0.040	0.006	0.060	0.0098	0.044	0.005	0.056	0.006	0.067
	336	0.0030	0.0321	**0.0016**	**0.0299**	0.0054	0.0565	0.004	0.049	0.006	0.054	0.013	0.100	0.006	0.065	0.007	0.069
	720	0.0025	0.0343	**0.0020**	**0.0338**	0.0058	0.0588	0.003	0.042	0.007	0.059	0.011	0.083	0.005	0.060	0.007	0.071
Exchange	96	**0.158**	**0.303**	0.164	0.309	0.162	0.311	1.327	0.944	0.237	0.377	0.298	0.444	0.372	0.478	1.563	0.974
	192	**0.269**	**0.403**	0.272	0.404	0.316	0.430	1.258	0.924	0.738	0.619	0.777	0.719	1.280	0.967	0.899	0.771
	336	**0.484**	**0.537**	0.497	0.538	0.565	0.582	2.179	1.296	2.018	1.070	1.833	1.128	1.750	1.108	2.952	1.520
	720	**1.140**	**0.811**	1.167	0.825	1.319	0.888	1.280	0.953	2.405	1.175	1.203	0.956	2.389	1.356	3.163	1.611
Traffic	96	0.218	0.307	**0.216**	**0.302**	0.256	0.362	0.257	0.353	0.226	0.317	0.313	0.383	0.542	0.531	0.497	0.509
	192	**0.192**	**0.292**	0.208	0.301	0.245	0.355	0.299	0.376	0.314	0.408	0.386	0.453	0.551	0.535	0.503	0.516
	336	0.190	0.284	**0.183**	**0.278**	0.266	0.388	0.312	0.387	0.387	0.453	0.423	0.468	0.555	0.536	0.505	0.518
	720	**0.205**	0.303	0.206	**0.295**	0.273	0.372	0.366	0.436	0.437	0.491	0.378	0.433	0.989	0.801	0.515	0.517