Accurate and Efficient Multivariate Time Series Forecasting via Offline Clustering

Yiming Niu*, Jinliang Deng†‡, Lulu Zhang*, Zimu Zhou§ Yongxin Tong*
*State Key Laboratory of Complex & Critical Software Environment,
Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, Beijing, China
†HKGAI, Hong Kong University of Science and Technology, Hong Kong SAR, China
‡Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen, China
§Department of Data Science, City University of Hong Kong, Hong Kong SAR, China
{yimingniu@, zhangluluzll@, yxtong@}buaa.edu.cn
dengjinliang@ust.hk, zimuzhou@cityu.edu.hk

Abstract—**Accurate and efficient multivariate time series (MTS) forecasting is essential for applications such as traffic management and weather prediction, which depend on capturing long-range temporal dependencies and interactions between entities. Existing methods, particularly those based on Transformer architectures, compute pairwise dependencies across all time steps, leading to a computational complexity that scales quadratically with the length of the input. To overcome these challenges, we introduce the Forecaster with Offline Clustering Using Segments (FOCUS), a novel approach to MTS forecasting that simplifies long-range dependency modeling through the use of prototypes extracted via offline clustering. These prototypes encapsulate high-level events in the real-world system underlying the data, summarizing the key characteristics of similar time segments. In the online phase, FOCUS dynamically adapts these patterns to the current input and captures dependencies between the input segment and high-level events, enabling both accurate and efficient forecasting. By identifying prototypes during the offline clustering phase, FOCUS reduces the computational complexity of modeling long-range dependencies in the online phase to linear scaling. Extensive experiments across diverse benchmarks demonstrate that FOCUS achieves state-of-the-art accuracy while significantly reducing computational costs.**

Index Terms—**multivariate time series, spatiotemporal data mining, forecasting**

I. Introduction

Accurate and efficient multivariate time series (MTS) forecasting is of great importance in various real-world applications [1]–[9]. MTS data, characterized by its complex structure encompassing both temporal and entity dimensions [5], [10]–[14], is crucial for enabling precise predictions across different application scenarios. For example, in traffic flow forecasting, accurate predictions can optimize resource allocation [4], [5], [15], while in weather forecasting, efficient predictions can provide timely warnings to the public [16], [17].

A key challenge in achieving accurate MTS forecasting lies in modeling long-range dependencies [18]–[21]. These dependencies capture the deep relationships across long time periods (*e.g.* seasonal variations of temperature) and multiple entities (*e.g.* traffic flow relationships between intersections) within the data [22]–[25]. Modeling these long-range dependencies involves two primary steps: First, identifying potential *events* within the data, such as peaks in traffic or fluctuations in climate. Each event corresponds to a cluster of similar time segments, where each cluster shares a representative segment pattern, referred to as a *prototype*. Next, these prototypes and their corresponding events are analyzed to further explore their temporal relationships and interactions between entities. This detailed modeling helps capture the long-range dependencies, thereby enabling accurate time series forecasting.

Modeling long-range dependencies in MTS poses a critical challenge. Canonical neural architectures, such as CNNs and RNNs, require increasingly more computational steps (hops) to connect distant time points as the time lag grows, making them ineffective for capturing long-range dependencies. In contrast, Transformer architectures [26]–[28], with their ability to directly model global correlations across sequences of arbitrary lengths, offer a promising solution to this problem. PatchTST [19] models the long-range dependencies between time segments, significantly improving forecasting accuracy compared with previous solutions. Although this approach implicitly identifies representative segment patterns during the learning process and models their dependencies, it does not utilize these patterns effectively. Instead, it still treats all segments as independent units, leaving the model confined to all-pairs dependency modeling framework, which results in $\mathcal{O}(L^2)$ computational complexity.

To enhance computational efficiency, Informer [18] introduces the ProbSparse self-attention mechanism, which performs sparse sampling on the attention matrix and computes attention only for critical key-value pairs, reducing complexity to $\mathcal{O}(L \log L)$. However, this sampling mechanism may discard critical information, resulting in performance degradation when capturing intricate patterns. Crossformer [29] leverages low-rank properties to compress input sequences and decompose the original large attention matrix into the product of two smaller matrices, reducing computational complexity to $\mathcal{O}(2kN)$, where N represents the number of entities. However, it heavily relies on dimensionality reduction. Although these methods reduce computational costs through sampling and dimensionality reduction, they fail to fundamentally eliminate the reliance on all-pairs dependency modeling on segments. As a result, the efficiency and accuracy trade-off remains, offering

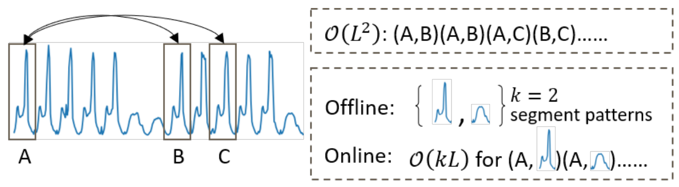

$\mathcal{O}(L^2)$: (A,B)(A,B)(A,C)(B,C)......
Offline:
$k = 2$
segment patterns
Online: $\mathcal{O}(kL)$ for (A,)(A,)......
A
B
C

Fig. 1: Modeling long-range dependency on Traffic dataset [27] with representative segment patterns (*i.e.* prototypes) discovered offline.

limited practical value in addressing the core issue.

A promising solution to address the complexity of long-range dependency modeling is to shift the identification of prototypes to an offline phase. Given specific applications, these prototypes are relatively universal and can be effective across various instances. Therefore, we can extract these prototypes offline from the dataset and apply them to different instances in the online phase, as shown in Fig. 1.

Our Solution. We propose the **F**orecaster with **O**ffline **C**lustering **U**sing **S**egments (FOCUS), a model that leverages prototypes from time series data offline. This approach simplifies modeling long-range dependencies in forecasting. The FOCUS operates in two distinct phases:

In the *offline clustering*, the focus is on extracting prototypes from the dataset. The process begins by dividing the entire training dataset's time series data into smaller segments. These segments are then clustered based on their similarities, with each cluster represented by a prototype that summarizes the overall pattern of the cluster. To ensure that these prototypes accurately reflect the features of their respective clusters, an iterative optimization process is employed. This process not only evaluates the distance-based similarity between the prototype and the segments but also considers the relational dependencies within the cluster. But there may be discrepancies between the prototypes and specific instances, suggesting a need to adapt the prototypes based on the instance.

In the *online adaptation*, the focus shifts to how the extracted prototypes can be adapted to the online input instances to achieve more accurate predictions. The online input is segmented in a manner similar to the offline phase, with each segment dynamically assigned to its closest prototype. This adaptation allows the prototypes to be tailored to current inputs. A deep learning model then establishes associations between the input segments and the adapted prototypes, effectively modeling the long-range dependencies. Based on these associations, the model performs precise forecasting.

Contributions. Our contributions are summarized as follows:

- We propose a novel forecasting model, FOCUS, that combines an offline clustering phase to identify representative segment patterns and an online adaptation phase to dynamically refine these patterns based on input data, ensuring accurate and efficient forecasting.
- We design an efficient mechanism to capture long-range dependencies with linear complexity by leveraging the correlations between a fixed set of representative segment patterns (*i.e.* prototypes) and the input sequence. This enables the model to handle long and high-dimensional sequences efficiently without compromising accuracy.
- Comprehensive experiments on diverse datasets demonstrate that FOCUS outperforms state-of-the-art methods in terms of computational efficiency, while ranking top-1 accuracy among the 8 models for comparison on 26 out of the 28 settings.

The remainder of this paper is organized as follows: In Sec. III, we discuss the motivation for addressing long-range dependency challenges. Sec. IV provides an overview of our method, followed by details on offline clustering in Sec. V and online adaptation in Sec. VI. We describe the dual-branch architecture for efficient feature fusion in Sec. VII. Finally, Sec. VIII presents experimental results demonstrating the efficiency and accuracy of FOCUS.

II. Problem Statement

TABLE I: Frequently used notations.

Notation	Description
d	the intrinsic dimension of features within one token
k	the number of prototypes for the offline and online process
p	Number of time steps within one segment
N	The total entities contains in the dataset
T	The total time steps contains in the dataset
L	Number of historical time steps (lookback)
L_f	Number of future time steps (horizon)
l	Number of segments
$\mathcal{D}$	The time series dataset
$\mathcal{X}$	Historical multivariate time series data
$\mathcal{X}_{e,t}$	Value of variable c at time step t
$\mathcal{Y}$	Actual future multivariate time series data
$\hat{\mathcal{Y}}$	Predicted future multivariate time series data
$\mathcal{P}$	A matrix of time series segments, each of length l.
$\mathcal{P}_{e,i}$	The i-th segment of entity e.
$\mathcal{M}$	Model used for forecasting
$\mathcal{H}$	The hidden state of multiple entities and multiple tokens

We first formally define the time series data and then define the multivariate time series forecasting problem. Tab. I shows the frequently used notations.

Definition 1: (Time Series) A time series represents a collection of attributes over time for a single variable. Formally, it is defined as:

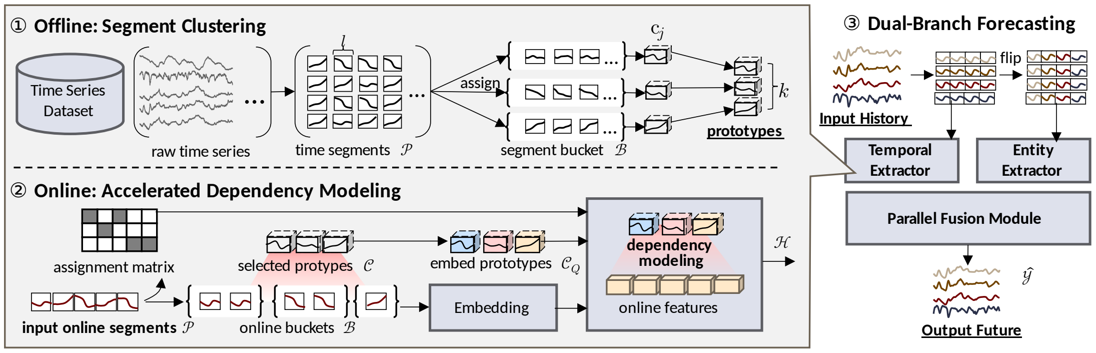
$$\mathcal{D}_e = \{x_{e,1}, x_{e,2}, \ldots, x_{e,T}\}, \tag{1}$$

where $\mathcal{D}_e$ denotes the data of variable e over T time steps, and $x_{e,t}$ is the value of variable e at time step t.

Definition 2: (Multivariate Time Series) A multivariate time series is a collection of time series for multiple variables or entities. For N variables or entities, it is defined as:

$$\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_D\}, \tag{2}$$

where each $\mathcal{D}_e = \{x_{e,1}, x_{e,2}, \ldots, x_{e,T}\}$ represents the sequence of values for variable e over T time steps. Additionally, for any two variables e_1 and e_2 ($1 \leq e_1 \neq e_2 \leq N$), the time index t is consistent across all variables, meaning that the time step corresponding to $x_{e_1,t}$ is the same as the time step



① Offline: Segment Clustering
Time Series Dataset
raw time series
time segments $\mathcal{P}$
l
assign
segment bucket $\mathcal{B}$
c_j
k
prototypes
② Online: Accelerated Dependency Modeling
assignment matrix
selected protypes $\mathcal{C}$
embed prototypes $\mathcal{C}_Q$
dependency modeling
online features
$\mathcal{H}$
input online segments $\mathcal{P}$
online buckets $\mathcal{B}$
Embedding
③ Dual-Branch Forecasting
Input History
flip
Temporal Extractor
Entity Extractor
Parallel Fusion Module
$\hat{y}$
Output Future

Fig. 2: Overview of the proposed FOCUS.

corresponding to $x_{e_2,t}$ for $t = 1, 2, \ldots, T$. Relationships or dependencies may exist between the variables.

Definition 3: (Multivariate Time Series Forecasting) The goal of multivariate time series forecasting is to predict future segments of the series based on observed historical segments. Given the historical data $\mathcal{X}$, which is a subset of the multivariate time series dataset $\mathcal{D}$ over a specific time period of length L, it is defined as:

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{D\times L}, \tag{3}$$

where $\mathbf{x}_t \in \mathbb{R}^N$ represents the values of D variables at time step t. The goal is to predict the future time series $\mathcal{Y}$:

$$\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{L_f}\} \in \mathbb{R}^{N\times L_f}, \tag{4}$$

where L_f is the number of future time steps, and $\mathbf{y}_t \in \mathbb{R}^N$ represents the values of N variables at future time step t. The forecasting task aims to find a model $\mathcal{M}$ such that:

$$\hat{\mathcal{Y}} = \mathcal{M}(\mathcal{X}), \tag{5}$$

where $\hat{\mathcal{Y}}$ denotes the predicted future time series.

III. Motivations

Modeling dependencies among all segments in multivariate time series theoretically captures all long-range dependencies, but its high computational complexity makes it impractical. This approach has quadratic computational complexity, resulting in excessive resource and time consumption when processing long and high-dimensional data. Thus, alternative methods are needed to reduce computational complexity.

To address this, we can explore representative segment patterns within the data as a more efficient solution. Representative segment patterns are representations of high-level system events, effectively capturing temporal and spatial repetition. These patterns exhibit stable recurrence over time and space, serving as key features of high-level system events.

Example 1: Consider a 7-day traffic flow time series as shown in Fig. 3, which can be divided into fixed-length intervals (e.g., hourly windows). These intervals typically exhibit simple, predictable patterns, such as increased congestion during rush hours and decreased flow at night. While these patterns may vary slightly depending on specific conditions, their recurrence remains stable across different days and locations. For instance, the heavy traffic during the 7-8 AM rush hour consistently appears across days (A and B), and intersections with similar road structures often show comparable patterns(A and C).

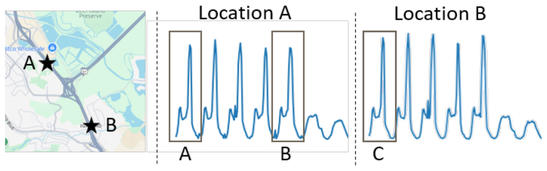


Fig. 3: Example from Traffic dataset [27] of a 7-day period.

Leveraging these patterns enables efficient dependency modeling across temporal and spatial dimensions. By modeling the dependencies between segments and high-level events, representative patterns significantly reduce computational complexity while retaining long-range dependency information. Additionally, combinations of representative patterns enable description of diverse high-level system events, endowing the model with the potential to generalize to unknown ones. This approach greatly lowers computational overhead. It preserves the ability to capture long-range dependencies while improving efficiency.

More importantly, the number of representative patterns is determined by the high-level system events, ensuring that computational costs grow linearly with sequence length, effectively addressing computational bottlenecks. Therefore, this approach demonstrates strong scalability for long-range dependency modeling in long and high-dimensional data.

IV. Overview of FOCUS

To leverage the representative segment patterns in time series data, we propose a two-phase method. In the offline phase, representative segment patterns are extracted from the dataset. In the online phase, these patterns are used to accelerate long-range dependency modeling, enabling efficient forecasting with minimal computational overhead when handling large-scale time series data.

As shown in Fig. 2, the method operates in two phases: offline clustering and online adaptation.

- In the *offline* phase, the time series is first divided into smaller segments. Next, these segments are assigned to buckets, where each bucket is associated with a specific prototype representing the common characteristics of the segments it contains. Subsequently, these prototypes are iteratively refined to better capture the most representative local structures in the data.
- In the *online* phase, the process starts by segmenting the incoming data and assigning each segment to the nearest prototype, resulting in the assignment matrix. Next, another matrix is computed to capture the relationships between the input segments and the prototypes through a deep learning approach. By combining these two matrices, the method efficiently models long-range dependencies. Importantly, this approach achieves linear computational complexity with respect to the input size, as the number of prototypes remains fixed regardless of the length of the online inputs.
- Furthermore, this approach is integrated into a dual-branch forecasting network, where long-range dependencies in the temporal and entity dimensions are modeled separately, features from both dimensions are extracted, and then fused to generate future predictions.

V. Offline: Segment Clustering

Understanding representative segment patterns in long and high-dimensional time series is crucial, as they provide a concise representation of complex dynamics. Those patterns and their corresponding high-level events often capture the system's underlying structure, allowing for significant simplification during preprocessing. By leveraging these patterns, we can potentially reduce the need for exhaustive pairwise computations for segments in downstream global dependency modeling, alleviating computational overhead.

We obtain representative segment patterns from the dataset by means of clustering. Unlike the clustering approaches used at the channel-level [30]–[32] in existing forecasting methods, this clustering method at the segment-level places more emphasis on the analysis of local similarity. However, although the clustering method optimized with Euclidean distance is effective in finding simple patterns [33], [34], it struggles to identify the higher-order correlations that are crucial for dependency modeling. This necessitates the use of more refined similarity metrics to account for the correlations between segments and guide the discovery of representative segment patterns.

Example 2: Suppose the traffic flow at intersection A changes as $\{9, 10, 11\}$*, at intersection B as* $\{7, 10, 13\}$*, and at intersection C as* $\{11, 10, 9\}$*. While the Euclidean distance between A and C is the same as that between A and B, the flow changes at intersection A are significantly more correlated with those at intersection B than with intersection C. This highlights the need to better differentiate between intersections B and C for modeling dependencies.*

We address this problem by augmenting distance-based clustering objectives with correlation-based optimization. Let the input dataset $\mathcal{D} = \{\mathcal{P}_e \mid e = 1, 2, \ldots, D\}$, where the sequence for each entity e is segmented into a sequence of segments $\mathcal{P}_{e,i} \in \mathbb{R}^p$. Here, p represents the length of each segment, and each entity contributes T/p segments, assuming T is divisible by p.

A prototype set $\mathcal{C} = \{\mathbf{c}_j \mid j = 1, 2, \ldots, k\}$ is defined, where each prototype $\mathbf{c}_j \in \mathbb{R}^p$ encapsulates a representative segment pattern. The assignment of each segment $\mathcal{P}_{e,i}$ to its closest prototype $\mathbf{c}_{q_{e,i}}$ is determined by minimizing a composite loss that combines numerical similarity and correlation alignment:

$$q_{e,i} = \text{argmin}_j \left(\|\mathcal{P}_{e,i} - \mathbf{c}_j\|^2 + \alpha \cdot (1 - \text{corr}(\mathcal{P}_{e,i}, \mathbf{c}_j)) \right), \tag{6}$$

where $\text{corr}(\mathcal{P}_{e,i}, \mathbf{c}_j)$ denotes the Pearson correlation coefficient between $\mathcal{P}_{e,i}$ and $\mathbf{c}_j$, and α is a fixed parameter balancing numerical similarity with correlation alignment. Each segment $\mathcal{P}_{e,i}$ is then assigned to the corresponding bucket:

$$\mathcal{B}_j = \{\mathcal{P}_{e,i} \mid q_{e,i} = j\}. \tag{7}$$

The prototype $\mathbf{c}_j$ of each bucket is optimized based on the reconstruction loss and correlation loss. The reconstruction loss $\mathcal{L}_{\text{rec}}$ enforces numerical consistency between each prototype and the mean characteristics of its assigned segments:

$$\mathcal{L}_{\text{rec}} = \sum_{j=1}^{k} \|\mathbf{c}_j - \text{mean}(\mathcal{B}_j)\|^2, \tag{8}$$

where $\text{mean}(\cdot)$ computes the mean segment across all $\mathcal{P}_{e,i}$ assigned to prototype $\mathbf{c}_j$. To capture temporal dynamics, the correlation loss $\mathcal{L}_{\text{corr}}$ maximizes the alignment between prototypes and their assigned segments, using the Pearson correlation coefficient:

$$\mathcal{L}_{\text{corr}} = -\sum_{j=1}^{k} \frac{1}{|\mathcal{B}_j|} \sum_{\mathcal{P}_{e,i} \in \mathcal{B}_j} \text{corr}(\mathcal{P}_{e,i}, \mathbf{c}_j). \tag{9}$$

For a dataset consisting of L time steps, the computational complexity of both types of loss is on the order of $O(L)$. The combined optimization objective is a weighted sum of the reconstruction and correlation losses:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \alpha \cdot \mathcal{L}_{\text{corr}}, \tag{10}$$

where α controls the weight of the correlation loss. The global objective integrates these levels of optimization:

$$\min_{\mathcal{C}} \mathcal{L}(\mathcal{C}, \mathbf{q}) \quad \text{subject to} \quad \mathbf{q} \in \text{argmin}_{\mathbf{q}'} \mathcal{L}_{\text{assign}}(\mathbf{q}', \mathcal{C}), \tag{11}$$

$$\mathcal{L}_{\text{assign}}(\mathbf{q}', \mathcal{C}) = \sum_{e,i} \text{Dis}(\mathcal{P}_{e,i}, \mathbf{c}_{q'_{e,i}}), \tag{12}$$

and $\text{Dis}(\mathcal{P}_{e,i}, \mathbf{c}_{q'_{e,i}})$ is defined as:

$$\text{Dis}(\mathcal{P}_{e,i}, \mathbf{c}_{q'_{e,i}}) = \left\|\mathcal{P}_{e,i} - \mathbf{c}_{q'_{e,i}}\right\|^2 + \alpha \cdot (1 - \text{corr}(\mathcal{P}_{e,i}, \mathbf{c}_{q'_{e,i}})). \tag{13}$$

To solve this, we employ the AdamW optimizer [35], iteratively updating the prototype set $\mathcal{C}$. This approach ensures that each prototype effectively balances numerical fidelity and morphological expressiveness.

Algorithm 1 Segment Clustering

```
Input: Time series data $\mathcal{D}$, Segment length $p$, Number of prototypes $k$
Output: Updated set of prototypes $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$
 1: Initialize the prototypes $\mathcal{C}$ with $k$ random prototypes
 2: for each entity $e$ in $\mathcal{D}$ do
 3:     Segment the time series $\mathcal{D}_e$ into segments of length $p$
 4:     Store all segments in $\mathcal{P}_e$
 5: end for
 6: Combine all segments: $\mathcal{P} = \bigcup_{e=1}^{D} \mathcal{P}_e$
 7: while not converged do
 8:     for each local pattern $\mathcal{P}_{e,i} \in \mathcal{P}$ do
 9:         Assign $\mathcal{P}_{e,i}$ to the nearest prototype $\mathbf{c}_j$
10:         Place $\mathcal{P}_{e,i}$ into the corresponding bucket $b_j$
11:     end for
12:     for each bucket $b_j \in \mathcal{B}$ do
13:         Compute total loss $\mathcal{L}_j$ for segments in $b_j$
14:         Update prototype $\mathbf{c}_j$ using gradient descent on $\mathcal{L}_j$
15:     end for
16: end while
17: return the updated prototypes $\mathcal{C}$
```

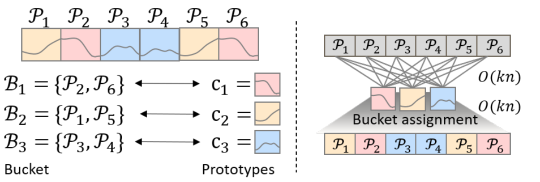

$\mathcal{P}_1$ $\mathcal{P}_2$ $\mathcal{P}_3$ $\mathcal{P}_4$ $\mathcal{P}_5$ $\mathcal{P}_6$
$\mathcal{B}_1 = \{\mathcal{P}_2, \mathcal{P}_6\}$
$\mathrm{c}_1 =$
$\mathcal{B}_2 = \{\mathcal{P}_1, \mathcal{P}_5\}$
$\mathrm{c}_2 =$
$\mathcal{B}_3 = \{\mathcal{P}_3, \mathcal{P}_4\}$
$\mathrm{c}_3 =$
Bucket
Prototypes
$O(kn)$
$O(kn)$
Bucket assignment

Fig. 4: Online bucket assignment and the computation for dependency modeling.

The resulting prototypes serve as a compact yet expressive representation of the time series. This approach aligns well with applications requiring scalable analysis by embedding temporal relationships into the prototype learning process.

VI. Online: Accelerated Dependency Modeling

In the online phase, our goal is to model long-range dependency between the temporal and entity dimensions of the input using observations over a specific time window containing N entities. To efficiently model long-range dependencies in multivariate time-series data, we propose Prototypes Attentive Modeling (ProtoAttn). As shown in Fig. 4, ProtoAttn leverages prototypes derived in the offline phase to group input sequences into segment buckets and models interactions between these prototypes and the original segments. This approach avoids the quadratic complexity of traditional self-attention mechanisms while preserving critical temporal relationships.

A. Implementation of ProtoAttn

The input sequence undergoes the same segmentation process as in the offline phase, resulting in $l \times N$ input segments, where $l = L/p$ represents the number of temporal segments per entity, and N is the total number of entities. Without loss of generality, we assume that p divides L evenly. For simplicity, we focus on modeling n segments which is denoted as $\mathcal{P}$, irrespective of whether they originate from different timestamps of the same entity or from different entities at the same timestamp. Let k denote the number of prototypes derived from the offline phase. ProtoAttn starts by computing a mapping matrix $A \in \mathbb{R}^{l \times k}$ that assigns input segments to the pattern buckets established offline, linking the input data to the prototypes. This mapping allows the transformation of input sequences into Query, Key, and Value:

$$\mathcal{C}_Q = \mathcal{C}W_E, \quad K = \mathcal{P}W_K, \quad V = \mathcal{P}W_V \tag{14}$$

where $\mathcal{P} \in \mathbb{R}^{l \times p}$ represents the input time segments, $\mathcal{C} \in \mathbb{R}^{k \times p}$ represents the prototypes, and $W_E, W_K, W_V \in \mathbb{R}^{p \times d}$ are the learnable projection matrices for feature mapping.

Next, the query matrix Q is constructed, where each query vector $q_i \in \mathbb{R}^d$ is represented by its corresponding cluster assignment. This allows us to define the approximate representation of q_i as:

$$\hat{q}_i = A_i \mathcal{C}_Q \tag{15}$$

where $\mathcal{C}_Q \in \mathbb{R}^{k \times d}$ represents a matrix of embedded prototypes, and $A_i \in \mathbb{R}^{1 \times k}$ is a one-hot vector denoting the cluster assignment for q_i. To compute the attention weights for each centroid c_{q_i} with respect to the keys K, we perform a dot product followed by a softmax normalization:

$$\alpha_i = \text{softmax}\left(\frac{c_{q_i}^\top K^\top}{\sqrt{d_k}}\right) \tag{16}$$

where $\alpha_i \in \mathbb{R}^n$ represents the attention distribution for the centroid c_{q_i}. The final output of the ProtoAttn mechanism is computed as a weighted sum of the value matrix V using the attention weights α_i:

$$\text{ProtoAttn}(q, K, V) = \alpha_i V \tag{17}$$

expanding this across all query centroids yields will get us:

$$\text{ProtoAttn}(\mathcal{C}_Q, K, V) = A\left(\text{softmax}\left(\frac{\mathcal{C}_Q K^\top}{\sqrt{d_k}}\right)\right) V \tag{18}$$

This formulation aggregates attention outputs for each centroid in the query space, ensuring that queries sharing the same centroid c_{q_i} will yield identical attention weights:

$$A_i = A_j \implies \alpha_i = \alpha_j \tag{19}$$

B. Analysis of the Online Dependency Modeling

The pseudocode in Algorithm 2 demonstrates the matrix computation form of the above process, which can better leverage the capabilities of parallel processors.

Complexity Analysis. The complexity analysis for the ProtoAttn begins with the computation of the assignment matrix A, where each segment $\mathcal{P}_i$ is assigned to the nearest prototype among k candidates, with a complexity of $\mathcal{O}(l \cdot k \cdot d)$. Constructing the Query, Key, and Value matrices involves projecting the

Algorithm 2 Online Dependency Modeling

```
Input: Input segment matrix $\mathcal{P} \in \mathbb{R}^{l\times d}$, Prototypes Set $C$
Output: Attention output ProtoAttn$(Q, K, V)$
 1: Initialize $A \in \mathbb{R}^{l\times k}$ as a zero matrix.
 2: for each segment $\mathcal{P}_i$ in $\mathcal{P}$ do
 3:     Assign $A[i, j] \leftarrow 1$, where $j$ is determined as the closest prototype to $\mathcal{P}_i$ by (6).
 4: end for
 5: $K \leftarrow \mathcal{P}W_K \in \mathbb{R}^{l\times d}$
 6: $V \leftarrow \mathcal{P}W_Q \in \mathbb{R}^{l\times d}$
 7: $\alpha \leftarrow \text{softmax}\left(\frac{\mathcal{C}_Q K^\top}{\sqrt{d_k}}\right)$
 8: ProtoAttn$(\mathcal{C}_Q, K, V) \leftarrow \alpha V$
 9: Output $\leftarrow A \times$ ProtoAttn$(Q, K, V)$
10: return Output
```

prototypes $\mathcal{C}$ and segments $\mathcal{P}$, resulting in a total complexity of $\mathcal{O}(k \cdot d^2 + l \cdot d^2)$. Computing attention weights α, applying them to the Value matrix V, and mapping outputs back to segments together have a complexity of $\mathcal{O}(k \cdot l \cdot d + l \cdot k \cdot d)$. Summing these, the total complexity is $\mathcal{O}(l\cdot(k\cdot d+d^2)+k\cdot d^2)$. Since d, and p are constants, this simplifies to $\mathcal{O}(kl)$, making ProtoAttn highly efficient with linear complexity relative to the number of segments l.

Approximation Analysis of ProtoAttn. An important property in the above process is that the number of fixed patterns discovered from the entire dataset is identified during the offline stage, and it is independent of the length of historical data input during the online stage or the length of the predicted future. Formally, when the input data is divided into segments, its *rank* should be smaller than the number of fixed patterns discovered from the entire dataset, *i.e.* the low-rank nature of the input data. We provide a theoretical proof that, when the input data indeed exhibits low-rank properties, the aforementioned process can be approximated as the long-range dependencies in self-attention process.

Theorem 1: Let $\mathcal{P} \in \mathbb{R}^{l\times p}$ be an input sequence matrix composed of segments, with rank$(\mathcal{P}) \le r$, r is the number of representative segment patterns found offline in the whole dataset, and let $W^Q, W^K \in \mathbb{R}^{d\times d}$ be weight matrices. Let w be a vector drawn from any column of the product $(W_Q W_K^T$. For any $\epsilon \in (0, 1)$, there exists a low-rank matrix approximation $\tilde{\mathcal{P}} = AC$, where $A \in \mathbb{R}^{l\times k}$ and $C \in \mathbb{R}^{k\times d}$, k is the number of patterns in online input, which satisfies $k \le r$, such that the following inequality holds:

$$\Pr\left(\|\tilde{\mathcal{P}}w^T - \mathcal{P}w^T\| \le \epsilon\|\mathcal{P}w^T\|\right) > 1 - o(1), \tag{20}$$

where the dimension k satisfies:

$$k = O\left(\frac{\log r}{\epsilon^2}\right). \tag{21}$$

Proof 1: Since $\mathcal{P}$ has rank at most r, we can decompose it as $\mathcal{P} = UV$, where $U \in \mathbb{R}^{l\times r}$ is a matrix with orthonormal columns and $V \in \mathbb{R}^{r\times p}$. We aim to show that for a matrix $\tilde{\mathcal{P}} = AC$, the low-rank approximation error with respect to w^T is bounded with high probability.

The first step is to show that we can approximate the matrix V using a matrix of rank k. To achieve this, we define $A' \in \mathbb{R}^{r\times k}$ and $C \in \mathbb{R}^{k\times d}$, and let $\tilde{V} = A'C$.. This ensures that rank$(\tilde{V}) \le$ rank$(A') = k \le r$, so $\tilde{V}$ is a low-rank approximation of V. Our goal is to show that $\tilde{V}$ approximates V well, i.e., we want to prove that the approximation error of V with respect to $\tilde{V}$ is small with high probability.

$$\Pr\left(\|\tilde{V}w^T - Vw^T\| \le \epsilon\|Vw^T\|\right) > 1 - o(1), \tag{22}$$

Here we provide a constructive proof by making $A' = VC^T$. We aim to show:

$$\Pr\left(\|VC^TCw^T - Vw^T\| \le \epsilon\|Vw^T\|\right) > 1 - o(1). \tag{23}$$

Using the Johnson-Lindenstrauss (JL) lemma [36], for any matrix $R \in \mathbb{R}^{k\times d}$ and vector $x \in \mathbb{R}^k$, we have the following inequality for any $y \in \mathbb{R}^k$:

$$\Pr\left(\|xR^TRy^T - xy^T\| \le \epsilon\|xy^T\|\right) > 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}, \tag{24}$$

where k satisfies:

$$k = \frac{5\log r}{\epsilon^2 - \epsilon^3}. \tag{25}$$

To apply this bound to the original matrix V, we define:

$$M_1 = VC^TCw^T - Vw^T. \tag{26}$$

$$M_2 = xC^TCw^T - xw^T, \tag{27}$$

$$\begin{aligned}\Pr\left(\|M_1\| \le \epsilon\|Vw^T\|\right) &\ge 1 - \sum_{x\in V}\Pr\left(\|M_2\| \le \epsilon\|xw^T\|\right)\\ &\ge 1 - 2ne^{-(\epsilon^2-\epsilon^3)k/4}\end{aligned} \tag{28}$$

Now we finished proving (23). Next, we can observe that the error between $\tilde{\mathcal{P}}w^T$ and $\mathcal{P}w^T$ can be written as:

$$\|\tilde{\mathcal{P}}w^T - \mathcal{P}w^T\| = \|U(A'Cw^T - Vw^T)\|. \tag{29}$$

Using the fact that U is a matrix with orthonormal columns, we know that:

$$\|U(A'Cw^T - Vw^T)\| = \|A'Cw^T - Vw^T\|. \tag{30}$$

Since $\mathcal{P}w^T = UVw^T$, we can conclude that:

$$\Pr\left(\|\tilde{\mathcal{P}}w^T - \mathcal{P}w^T\| \le \epsilon\|\mathcal{P}w^T\|\right) > 1 - o(1), \tag{31}$$

Thus, let $A = UA'$, we have shown that the low-rank approximation ACw^T of $\mathcal{P}w^T$ has a small error with high probability, completing the proof.

□

This theorem demonstrates that the input sequence matrix can be decomposed into a low-rank form that effectively approximates the original attention matrix without introducing bias. Notably, the rank of the approximation is independent of the input size and is determined by the intrinsic properties of the time series itself, as discussed in Sec. III.

VII. Dual-Branch Forecasting

In this section, we present the design of the dual-branch fusion forecasting network, FOCUS, which utilizes the former two-phase process to model temporal and entity correlations for accurate foreacasting. In Sec. VII-A, we shows how we use the online operation to construct the dual-branch feature extractor. In Sec. VII-B, we introduce a readout mechanism to capture useful information from extracted features efficiently to enhance forecasting accuracy.

A. Efficient Long-range Dependency Extractor

Accurate time series forecasting requires capturing both temporal dynamics and inter-entity relationships. The temporal dimension represents sequential dependencies that evolve over time, while the entity dimension models the interactions between different variables, which is particularly crucial in multivariate forecasting scenarios.

Unlike traditional approaches that rely on multiple stacked layers to model dependencies, we propose a dual-branch design that simultaneously captures temporal and entity correlations within a unified framework. This design integrates seamlessly with our accelerated correlation modeling in the online phase, significantly enhancing both the expressiveness and efficiency of forecasting models.

To capture temporal dependencies, we construct long-range relationships along the time dimension by processing input data sequentially. This is achieved by modeling the interactions and dependencies between segments at different positions within the same entity, which effectively encapsulates the sequence's intrinsic temporal characteristics. To capture the significant impact of entity interactions on system behavior, we shift the focus to the entity dimension, enabling the model to learn entity-level relationships at each time point. The procedure for feature extraction in both dimensions is described in Algorithm 3, where the temporal and entity feature matrices, $\mathcal{H}_t$ and $\mathcal{H}_e$, are computed using parallel operations on the input time series data.

Algorithm 3 Dual-Branch Feature Extraction Network

```
Input: Input time series data $\mathcal{X}$ with dimensions $[L \times N]$, segment length $p$
Output: Temporal features $\mathcal{H}_t$ and entity features $\mathcal{H}_e$
1: Divide $\mathcal{X}$ along the time dimension into overlapping/non-overlapping segments of length $p$, forming $\mathcal{P}$
2: for each entity $i = 1$ to $N$ do
3:     Select the time segments $\mathcal{P}_t^{(i)}$ for entity $i$
4:     Compute temporal features:
           $\mathcal{H}_t^{(i)} \leftarrow \text{LayerNorm}\left(\text{OnlineModeling}(\mathcal{P}_t^{(i)}) + \mathcal{P}_t^{(i)}\right)$
5: end for
6: Concatenate $\mathcal{H}_t^{(i)}$ across all entities along the entity dimension to form $\mathcal{H}_t$
7: for each time segment $j = 1$ to $L/p$ do
8:     Select the entity segments $\mathcal{P}_e^{(j)}$ for time segment $j$
9:     Compute entity features:
           $\mathcal{H}_e^{(j)} \leftarrow \text{LayerNorm}\left(\text{OnlineModeling}(\mathcal{P}_e^{(j)}) + \mathcal{P}_e^{(j)}\right)$
10: end for
11: Concatenate $\mathcal{H}_e^{(j)}$ across all time segments along the time dimension to form $\mathcal{H}_e$
12: return $\mathcal{H}_t, \mathcal{H}_e$
```

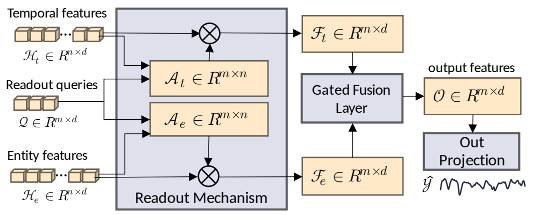

Temporal features
$\mathcal{H}_t \in R^{n\times d}$
Readout queries
$\mathcal{Q} \in R^{m\times d}$
Entity features
$\mathcal{H}_e \in R^{n\times d}$
$\mathcal{A}_t \in R^{m\times n}$
$\mathcal{A}_e \in R^{m\times n}$
Readout Mechanism
$\mathcal{F}_t \in R^{m\times d}$
Gated Fusion Layer
$\mathcal{F}_e \in R^{m\times d}$
output features
$\mathcal{O} \in R^{m\times d}$
Out Projection
$\hat{y}$

Fig. 5: The information flow of Parallel Fusion Module.

B. Efficient Parallel Fusion Module

Efficiently integrating diverse feature sets is crucial for enhancing predictive performance in sequential data tasks. As shown in Fig. 5, our proposed *Parallel Fusion Module* merges temporal and entity features in a computationally efficient manner, achieving linear scalability with respect to input sequence length.

The process begins by generating a fixed number of *readout queries* from the input features, with their length m corresponding to the desired forecast horizon. Using these queries, we compute their correlations with extracted features to capture their dependencies, producing prediction representations enhanced by each dimension. The gating mechanism dynamically adjusts the contributions of temporal and entity features, prioritizing the most relevant information for accurate forecasting. Finally, the output is directly projected to produce the future prediction.

The pseudo-code of the Parallel Fusion Module is presented in Algorithm 4. The size of correlation matrices are linear to the input length as the number of readout queries are fixed.

Complexity Analysis. The overall complexity of FOCUS is mainly determined by online dependency modeling. For temporal feature extraction, processing the sequence along the temporal dimension divides it into $l = L/p$ segments, with a complexity of $O(kL/p)$ (k is the number of prototypes, L is the temporal length). For entity feature extraction, modeling dependencies between N entities at each time step gives N segments and a complexity of $O(kN)$.

In the fusion processing, projecting input features $\mathcal{H}_t \in \mathbb{R}^{l\times d}$ and $\mathcal{H}_e \in \mathbb{R}^{N\times d}$ into m fixed readout queries has a complexity of $\mathcal{O}(m \cdot l \cdot d + m \cdot N \cdot d)$. Computing correlation matrices $\mathcal{A}_t$ and $\mathcal{A}_e$ also costs $\mathcal{O}(m \cdot l \cdot d + m \cdot N \cdot d)$, and aggregating features via matrix multiplications is $\mathcal{O}(m \cdot l \cdot d + m \cdot N \cdot d)$, while the fusion projection and gating mechanism

Algorithm 4 Parallel Fusion Module

```
Input: Temporal features $\mathcal{H}_t$, Entity features $\mathcal{H}_e$, Number of readout queries $m$
Output: Forecasting result $\hat{\mathcal{Y}}$
1: Project input features into $m$ fixed readout queries $\mathcal{Q}$ to represent essential information
2: Compute $\mathcal{A}_t = \text{softmax}\left(\frac{\mathcal{H}_t \mathcal{Q}^\top}{\sqrt{d}}\right)$ for temporal features
3: Compute $\mathcal{A}_e = \text{softmax}\left(\frac{\mathcal{H}_e \mathcal{Q}^\top}{\sqrt{d}}\right)$ for entity features
4: Calculate $\mathcal{F}_t = \mathcal{A}_t \mathcal{H}_t$ and $\mathcal{F}_e = \mathcal{A}_e \mathcal{H}_e$
5: Concatenate $\mathcal{F}_t$ and $\mathcal{F}_e$ to obtain $\mathcal{F}_{\text{proj}}$
6: Use a gating mechanism $g(\mathcal{F}_{\text{proj}})$ to adjust the contributions of each features:
7: $\mathcal{O} \leftarrow g(\mathcal{F}_{\text{proj}}) \odot \mathcal{F}_t + (1 - g(\mathcal{F}_{\text{proj}})) \odot \mathcal{F}_e$
8: Map the final output $\mathcal{O}$ to the forecasted result $\hat{\mathcal{Y}}$
9: return $\hat{\mathcal{Y}}$
```

add $\mathcal{O}(m \cdot d^2)$. Summing these, the total complexity is $\mathcal{O}(m \cdot l \cdot d + m \cdot N \cdot d + m \cdot d^2)$. With m fixed, it simplifies to $\mathcal{O}(l \cdot d + N \cdot d + d^2)$, and for large-scale inputs, further to $\mathcal{O}(l \cdot d + N \cdot d)$. Since $l = (L/p)$, the complexity is $\mathcal{O}((L/p) \cdot d + N \cdot d)$, showing linear scalability with L and N.

VIII. Experiments

In this section, we evaluate FOCUS on the long-term multivariate forecasting task, with a primary focus on accuracy and efficiency comparisons. We include many widely-used datasets from various real-world scenarios, including traffic, weather, and power systems. Our code is now publicly available[1].

A. Long-range Forecasting Experiment

Datasets. We evaluate our proposed FOCUS on multiple real-world multivariate time series (MTS) datasets, as summarized in Tab. II. Following standard data split configurations, we use a 7/1/2 train/validation/test ratio for Weather, Electricity, and Traffic datasets, and a 6/2/2 ratio for ETT, PEMS04, and PEMS08 datasets. These datasets are normalized using statistical information derived from the training set, consistent with prior studies [11], [19]. Widely recognized as benchmarks in MTS forecasting research [19], [27], [29], [37], [38], these datasets span a variety of sizes and entity counts, representing diverse real-world applications and effectively capturing the multifaceted characteristics of multivariate time series data.

TABLE II: Statistics of multivariate time series datasets.

Dataset	Domain	Frequency	Lengths	Dim	Split
PEMS04	Traffic	5 mins	16,992	307	6:2:2
PEMS08	Traffic	5 mins	17,856	170	6:2:2
ETTh1	Electricity	1 hour	14,400	7	6:2:2
ETTm1	Electricity	15 mins	57,600	7	6:2:2
Traffic	Traffic	1 hour	17,544	862	7:1:2
Electricity	Electricity	1 hour	26,304	321	7:1:2
Weather	Environment	10 mins	52,696	21	7:1:2

[1]https://anonymous.4open.science/r/ICDE25-ScalableForecasting-0AC5/

Metrics. In terms of evaluation metrics, we focus primarily on the predictive accuracy and computational efficiency of the model. For accuracy, we utilize Mean Absolute Error (MAE) and Mean Squared Error (MSE), where lower values indicate better forecasting performance. These metrics are widely recognized as benchmarks for evaluating the accuracy of time series forecasting, as both quantify the numerical differences between the predicted and actual sequences. For efficiency, in alignment with existing conventions [11] and to minimize the impact of varying deep learning platforms and operating system conditions (*e.g.* concurrent program execution), we adopt three key metrics: the number of floating-point operations (FLOPs) involved during inference, peak memory usage during inference (Peak Memory), and the model's parameter count. FLOPs reflect the total computational workload of the model, peak memory usage represents the space required for intermediate computation results, and parameter count indicates the storage capacity demands. These metrics collectively provide a comprehensive assessment of the computational bottlenecks that time series forecasting models might face in real-world scenarios.

Baselines. We compare FOCUS with state-of-the-art models for multivariate time series forecasting, encompassing various architectures as follows:

- PatchTST [19]: A single-channel forecasting model based on a transformer architecture, widely used for long-term forecasting benchmarks and known for its robust and consistent accuracy across diverse datasets.
- Crossformer [29]: A sophisticated model that employs Transformer structures along both the temporal and entity dimensions for enhanced performance.
- MTGNN [39]: A widely adopted model that integrates adaptive graph convolutional networks with temporal convolutional networks in an optimized architecture.
- Graph Wavenet [40]: An efficient forecasting model using adaptive graph modeling and dilated causal convolution.
- TimesNet [37]: A model that introduces Temporal 2D-Variation Modeling to achieve strong performance.
- LightCTS [11]: A recent model that employs a refined structural design for efficient time series forecasting.
- DLinear [38]: A simple yet effective model based on linear layers for multi-step prediction.

Implementation Details. All experiments are conducted on a cloud server equipped with two NVIDIA Tesla V100 GPUs, each with 32GB of memory. To ensure fairness, we use the original configurations for all baseline models. To align with existing works in long-term forecasting [19], [29], a lookback window of 512 steps is adopted, with forecasting horizons set to 96 and 336 steps. The correlation loss weight is set to $\alpha = 0.2$ during the offline phase. For FOCUS, we use a single-layer structure for both the Temporal Extractor and the Entity Extractor, while the feature mixing layer employs readout tokens. The number of readout tokens (m) is set to 6 for a forecasting horizon of 96 steps and 21 for 336 steps. The embedding size (d) is configured as $d = 128$ for the

TABLE III: Comparison of long-range forecasting accuracy with baselines

Models		FOCUS		PatchTST		Crossformer		MTGNN		Graph Wavenet		TimesNet		LightCTS		DLinear	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PEMS04	96	**0.0758**	**0.170**	0.102	0.228	0.103	0.209	0.0835	0.189	0.0838	0.186	0.0950	0.203	0.115	0.229	0.129	0.219
	336	**0.0936**	**0.190**	0.120	0.240	0.125	0.234	0.108	0.218	0.105	0.209	0.101	0.208	0.123	0.230	0.161	0.245
PEMS08	96	**0.0504**	**0.139**	0.0775	0.200	0.0773	0.190	0.0581	0.158	0.0615	0.157	0.66	0.172	0.0835	0.196	0.115	0.207
	336	**0.066**	**0.161**	0.0833	0.201	0.0929	0.207	0.0757	0.184	0.0793	0.183	0.07	0.177	0.0900	0.200	0.140	0.233
ETTh1	96	**0.372**	**0.402**	0.391	0.422	0.418	0.449	0.454	0.472	0.458	0.472	0.428	0.452	0.401	0.429	0.401	0.424
	336	**0.391**	**0.423**	0.434	0.463	0.751	0.681	0.457	0.474	0.543	0.531	0.454	0.425	0.506	0.495	0.424	0.446
ETTm1	96	0.304	**0.352**	**0.297**	0.354	0.324	0.370	0.366	0.427	0.358	0.402	0.316	0.369	0.312	0.363	0.307	0.358
	336	**0.366**	**0.394**	0.368	0.395	0.418	0.442	0.523	0.525	0.449	0.459	0.381	0.410	0.392	0.417	0.371	0.399
Traffic	96	**0.387**	**0.275**	0.388	0.299	0.520	0.287	0.492	0.291	0.523	0.292	0.623	0.337	0.596	0.415	0.395	0.276
	336	**0.414**	**0.285**	0.418	0.313	0.525	0.286	0.535	0.321	0.570	0.325	0.652	0.374	0.623	0.383	0.421	0.331
Electricity	96	**0.132**	**0.227**	0.155	0.274	0.136	0.236	0.137	0.238	0.146	0.247	0.190	0.296	0.177	0.279	0.135	0.233
	336	**0.164**	**0.259**	0.192	0.305	0.166	0.266	0.176	0.281	0.191	0.292	0.206	0.309	0.208	0.307	0.166	0.267
Weather	96	**0.148**	**0.199**	0.152	0.205	0.154	0.220	0.1613	0.227	0.161	0.221	0.170	0.229	0.149	0.203	0.155	0.223
	336	0.236	**0.280**	0.240	0.284	0.264	0.333	0.260	0.330	0.259	0.323	0.260	0.305	**0.234**	0.281	0.239	0.302

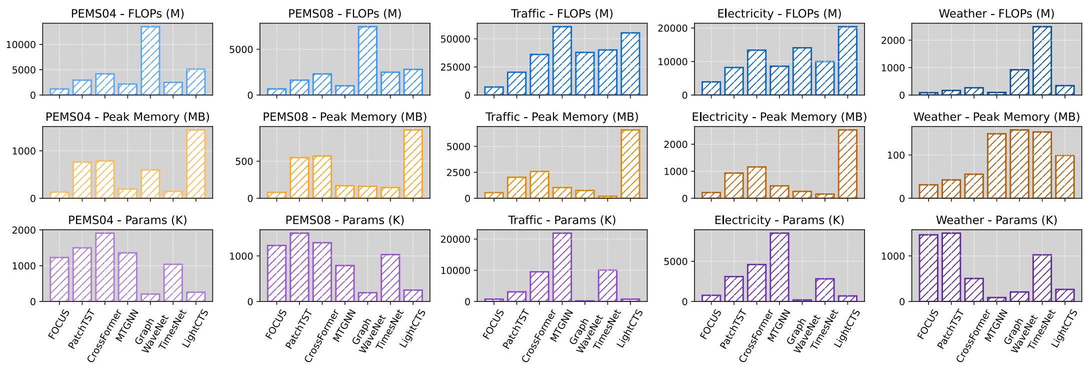


Fig. 6: Comparison of FLOPs, Peak Memory Occupation, Number of Parameters with baselines.

PEMS04 and PEMS08 datasets, and $d = 64$ for all other datasets. Other hyperparameters employed in the experiment, including the segment length p and the number of prototypes k, were obtained through the grid-search method.

Conclusion. The results in Tab. III highlight FOCUS's outstanding accuracy in long-term forecasting across seven diverse datasets. The best results are in **bold**, and the second-best results are underlined. FOCUS consistently outperforms nearly all baseline models in both MAE and MSE metrics, with only a marginal MSE difference in favor of LightCTS on the Weather dataset and a marginal difference with PatchTST on ETTm1. In most datasets and scenarios, FOCUS demonstrates a significant advantage, surpassing models like PatchTST and Crossformer with superior accuracy. On more challenging datasets, such as PEMS04, PEMS08, and Traffic, FOCUS exhibits both high precision and an efficient approach.

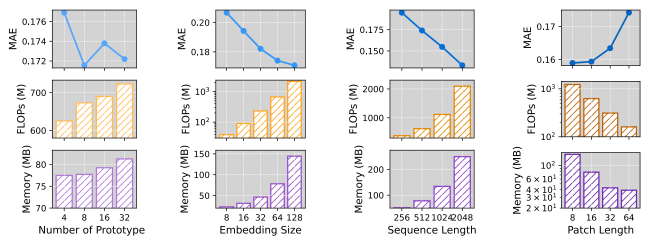
As illustrated in Fig. 6, FOCUS demonstrates significantly lower FLOPs and peak memory consumption compared to other baseline models when processing long input sequences. This efficiency ensures that FOCUS remains practical even on resource-constrained devices, enabling robust performance without overburdening hardware. While FOCUS may exhibit slightly higher parameter counts in specific scenarios, it effectively mitigates peak memory usage by minimizing intermediate variable storage, maintaining a balance between computational cost and accuracy.

B. Parameter Study

We systematically examine the influence of key hyperparameters on the performance of FOCUS, focusing on the input length, embedding size, and the number of prototypes. These parameters are chosen due to their adjustable nature and their substantial impact on the model's architecture and performance. The analysis is conducted on the PEMS08 dataset, with results visualized in Fig. 7. This evaluation provides insights into the sensitivity of FOCUS to these critical hyperparameters, enabling a deeper understanding of its performance and guiding parameter tuning for practical use.

1) Impact of Number of Prototypes k: Fig. 7a depicts the impact of varying the number of prototypes k in FOCUS on the PEMS08 dataset. Increasing k enlarges the prototype correlation matrix we need to calculate in the online phase, driving a predictable and steady increase in both FLOPs and peak memory consumption. At the same time, more prototypes can capture more meaningful correlations, enhancing model performance to a degree. Notably, once k exceeds a certain threshold, further gains in prediction accuracy plateau, aligning with our motivation in Sec. III, which highlights that the segment patterns in the data are not overly complex.

2) Impact of Embedding Size d: Fig. 7b illustrates how the embedding size d in FOCUS affects performance on



(a) (b) (c) (d)

Fig. 7: Impact of (a) number of prototypes k, (b) embedding size d, (c) length of the input sequence L and (d) patch length p in FOCUS for long-term forecasting on PEMS08 dataset.

the PEMS08 dataset. As d increases, both FLOPs and peak memory usage rises while prediction accuracy gradually improves. Specifically, as d grows, we observe a steady decline and eventual convergence of prediction error. However, this improvement comes at a steep cost: the marginal gains in accuracy diminish rapidly as the computational overhead continues to escalate. This insight suggests that the selection of an appropriate embedding size should be guided by a balance between performance needs and resource constraints, tailored to the specific application context.

3) Impact of Input Window Size L: Fig. 7c shows how the input window size L influences performance in FOCUS on the PEMS08 dataset. Extending the input window increases the information available to the model, consistently leading to a reduction in prediction error. However, unlike the trend observed with embedding size scaling, we see a stable improvement in predictive capability. This enhancement comes at a cost: processing longer input sequences inflates both FLOPs and peak memory requirements. This behavior supports a potential scaling law in time series prediction, as suggested in [41], which posits that longer input sequences can yield substantial performance gains. FOCUS adheres well to this principle, demonstrating its strong potential for scaling effectively to exploit ultra long input sequences.

4) Impact of Patch Length p: Fig. 7d shows the impact of patch length on model accuracy and performance. Shorter patches boost performance by generalizing to diverse temporal patterns as we mentioned in Sec. III, but increase processing overhead as more patches are needed for the prediction. This finding underscores the importance of striking a balance between the model's generalization ability and efficiency when selecting the patch length.

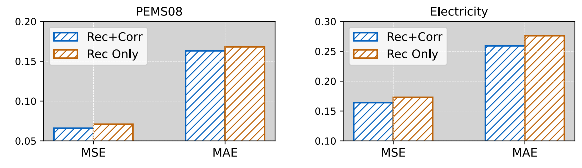
C. Ablation Study

We conducted an ablation study on the PEMS04 dataset to quantify the impact of each component in FOCUS, assessing both accuracy and efficiency through several model variants:

- *FOCUS-Attn*:The feature extractors are replaced with Self-Attention layers.
- *FOCUS-LnrFusion*: The Parallel Fusion Module is replaced by a gated Linear layer.

TABLE IV: Ablation Study

Dataset	Model	MSE	MAE	FLOPs(M)	Mem(MB)	Param(K)
PEMS08	FOCUS	**0.0711**	**0.168**	673	79.23	1227
	FOCUS-Attn	0.0864	0.178	1235	96.13	
	FOCUS-LnrFusion	0.0875	0.191	559	54.93	1438
	FOCUS-AllLnr	0.0897	0.193	**519**	**50.96**	1429
Electricity	FOCUS	**0.162**	**0.258**	3929	266	2617
	FOCUS-Attn	0.163	0.258	4434	305	2650
	FOCUS-LnrFusion	0.167	0.264	3134	177	2989
	FOCUS-AllLnr	0.173	0.281	**2966**	**168**	2956



(a) PEMS08 (b) Electricity

Fig. 8: Comparison of accuracy between prototypes optimized only based on Euclidean distance reconstruction error (*Rec Only*) and those incorporating correlation error (*Rec+Corr*).

- *FOCUS-AllLnr*: Both feature extractors and The Parallel Fusion Modules are replaced by Linear layers.

As shown in Tab. IV, substituting the online modeling process in the extractors with Self-Attention (FOCUS-Attn) increases complexity, resulting in higher FLOPs and peak memory usage, but provides negligible improvements in MSE and MAE. Replacing the Parallel Fusion Module with gated linear layers (FOCUS-LnrFusion) reduces accuracy, highlighting the effectiveness of our feature fusion strategy. The fully linear variant (FOCUS-AllLnr), while the most efficient in terms of FLOPs and peak memory usage, shows the poorest accuracy. A comparison between FOCUS-AllLnr and FOCUS-LnrFusion reveals that the online modeling process introduces only minimal additional resource demands than linear layer. These findings demonstrate that our design achieves an effective balance between performance and resource efficiency.

D. Study towards generalization on test set

The non-stationary property of time series data often causes new segment patterns to emerge in the test set. This challenges the model's generalization ability.

To test its impact on model performance, we used the t-SNE [42] method to compare segment distributions in the training and test sets of the Electricity dataset. We then identified test-set instances containing unseen segments to test the model's forecasting accuracy, comparing with PatchTST (also segmentation-based), as shown in Fig. 9. The results indicate that the input sequences contain unseen segments, mainly characterized by steeper intra-segment trends. FOCUS predicts larger data-change magnitudes than PatchTST, better following ground truth trends. In terms of the method, because the clustering process of FOCUS assists the model in associating new segments with known segments, it can reduce the difficulty for FOCUS to understand new segment patterns.

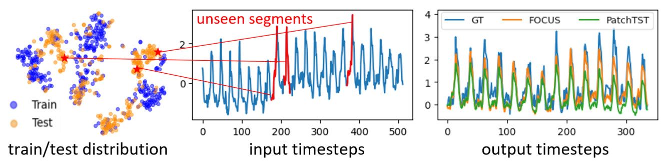


Fig. 9: A test instance in the Electricity dataset and the prediction results of FOCUS and PatchTST on it.

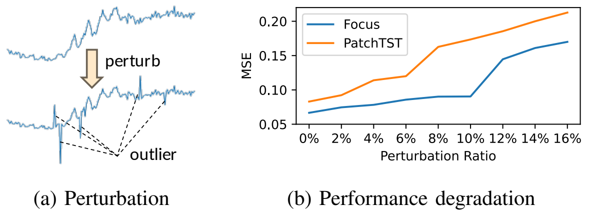


Fig. 10: The forecasting accuracy under different perturbation ratios.

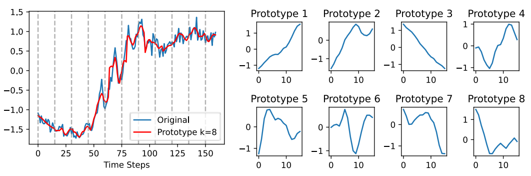


Fig. 11: Series approximation of the rise at morning with the number of prototypes $k = 8$.

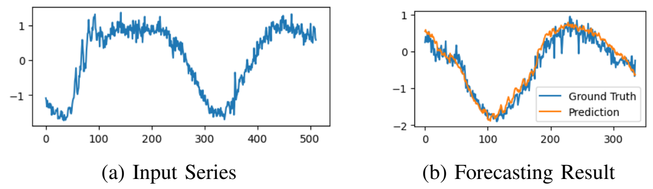


Fig. 12: Visualization of the input series for case study and corresponding forecasting result.

E. Study towards the effect of outliers

When collecting time series data, outliers often occur due to collection device reliability issues. These outliers can harm the model's modeling ability, so it's crucial to test the model's outlier resistance. We simulate different ratio by replacing training data points with outliers (sampled from a distribution over three-times the real data's standard deviation), as shown in Fig. 10a and then test the model's forecasting accuracy.

As shown in Fig. 10b, the prediction accuracy of FOCUS stays relatively stable under a certain degree of perturbation. There is a notable increase when the perturbation ratio is around 10%. A similar phenomenon is observed in PatchTST, where its accuracy spikes when the perturbation ratio is about 6%. By comparison, FOCUS has stronger ability to resist outliers. In terms of the method, FOCUS assigns data segments to the nearest clustering centers. This operation helps to minimize the impact of outliers, ensuring the performance remains stable even as the perturbation ratio improves.

F. Experiment on Offline Processing

In the offline clustering phase, we utilize reconstruction error based on Euclidean distance and correlation error based on Pearson correlation to identify and optimize typical segment patterns in the dataset, generating prototypes for use in the online phase. To evaluate the impact of clustering objectives, we compare two approaches for obtaining prototypes: clustering optimized solely with reconstruction error (Rec Only) and clustering optimized with both reconstruction and correlation errors (Rec+Corr) on PEMS08 and Electricity datasets, while keeping all other settings consistent. Since the primary goal of the prototypes is to enhance prediction accuracy during the online phase, we use the prediction accuracy of the final model trained with each set of prototypes as the evaluation metric.

Fig. 8 illustrates the effects of correlation loss. Models utilizing prototypes obtained with correlation error demonstrate improved prediction accuracy in terms of both MSE and MAE. Meanwhile, we observe that the additional running time is indistinguishable from noise, which means that for the proposed method, the improvement is almost cost-free. This indicates that incorporating correlation error during the offline phase yields prototypes that better support long-sequence modeling and prediction in the online phase.

G. Case Study

We conducted an in-depth case study on a randomly sampled sequence from the PEMS08 dataset, which is shown in Fig. 12a, evaluating the quality of the model's predictions, how the model interprets the sequence dynamics, and analyzing how prototypes effectively approximate long sequences.

Forecasting Result. Fig. 12b presents the model's forecasting results. The predictions closely match the ground truth, even for subtle patterns, such as the slight rise before the traffic decline and multiple spikes during the morning rise. The ability to predict these intricate details indicates the model's proficiency in understanding long-term dependencies and the sequence's nuanced behaviors.

Learned Long-range Dependency. We further examined the temporal dependencies learned by FOCUS, obtained by directly multiplying the assignment matrix with the online correlation matrix, as shown in Fig. 13. Notably, the analysis highlights a strong dependency between the rise of traffic flow in the morning and the decline in the night. This insight confirms that our model captures significant long-range dependencies and understands the underlying temporal dynamics of traffic patterns.

Approximation via Prototypes. To approximate the original time series effectively, we decomposed the sequence into $k = 8$ prototypes, with each prototype adjusted to maintain the original mean and standard deviation. As illustrated in Fig. 11, these prototypes capture critical features of the sequence, such

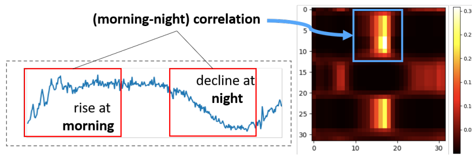


Fig. 13: An example of long-range dependency extracted by FOCUS.

as the distinct spikes observed between the 16th and 64th time steps. This demonstrates that a limited set of prototypes, when combined with local statistical details, can capture and approximate the essential patterns of complex time series data.

H. Summary of Major Experimental Findings

The major experimental findings are summarized as follows:

- FOCUS achieves outstanding accuracy with minimal FLOPs compared to a diverse and comprehensive set of baselines across multiple datasets, while still maintaining lower peak memory usage.
- Parameter study reveal that FOCUS maintains feasible computational overhead and consistently improves prediction accuracy when scaling to longer time series.
- The case study highlights the effectiveness of FOCUS's approximation for long sequence representation and its ability to capture meaningful long-range dependencies.

IX. Related Work

Multivariate Time Series Forecasting. Multivariate Time Series (MTS) forecasting, essential for modeling the temporal dynamics of multiple variables, has garnered significant attention [43] due to its applications in various domains [2]–[4], [15]–[17], [44]–[47].

Statistical models like ARIMA [1] and VAR [48] assume linear dependencies, limiting their effectiveness for complex, high-dimensional data. Deep learning approaches, including LSTNet [49] and Wavenet [50], leverage CNNs or RNNs to model nonlinear spatial and temporal patterns. However, these methods struggle with long-term dependencies and scalability.Spatial-Temporal Graph Neural Networks (STGNNs) like AGCRN [51], METRO [52], Graph Wavenet [40], and MTGNN [39] integrate graph convolutions for spatial dependencies. While effective, their limited receptive fields hinder long-term forecasting.

Transformers overcome these limitations with self-attention mechanisms, excelling at modeling long-term dependencies. Informer [18] improves efficiency with ProbSparse attention, Autoformer [27] enhances interpretability via decomposition, and PatchTST [19] scales to high dimensions with patch-based representations. Crossformer [29] integrates temporal and inter-variable attention, setting new benchmarks.

Efficient Long-Range Dependency Modeling. Handling long-range dependencies is a fundamental challenge in multivariate time series, as these dependencies capture essential relationships across time and variables. Transformers revolutionized sequence modeling by capturing long-range dependencies. However, their quadratic complexity ($O(L^2)$) limits the scalability for time series data [18].

Recent work aims to optimize processing performance with different inductive biases. Informer [18] prunes low-value info using the attention mechanism's low-rank property, which cuts costs but risks losing information. Pyraformer [53] tries to lower complexity in a hierarchical way, though this introduces propagation errors. FedFormer [28] uses time-series data's frequency-domain features, but frequency-domain sampling may lose important info. PatchTST [19] and Crossformer [29] discretize time-series data by patching. This improves prediction efficiency and reaches top-notch accuracy. However, patching can't reduce the time-series dimension complexity.

Based on discretization, FOCUS starts from the time series data itself. It attains the discretized representation by clustering time-series segments based on their similarity, enables time-series modeling and prediction with linear complexity.

X. Conclusion

In this paper, we introduce FOCUS, a novel two-phase multivariate time series forecasting model. It aims to balance accuracy and computational efficiency when modeling long-range dependencies. FOCUS combines an offline clustering phase with an online adaptation phase. The offline phase identifies representative segment patterns from the whole dataset, capturing key time-series features. These patterns are then dynamically adjusted in the online phase to adapt to trends and capture long-range dependencies in the input data. This integration enables FOCUS to efficiently capture complex long-range dependencies while reducing computational overhead. Experiments on various datasets show that FOCUS has significant advantages in forecasting accuracy and computational efficiency compared to SOTA models. Its robustness and adaptability make it an attractive choice for practical applications, especially in resource-constrained environments.

XI. Acknowledgements

This work was partially supported by National Key Research and Development Program of China under Grant No. 2023YFF0725103, National Science Foundation of China (NSFC) (Grant Nos. 62425202, U21A20516, 62336003), the Beijing Natural Science Foundation (Z230001), the Fundamental Research Funds for the Central Universities No. JK2024-03, the Didi Collaborative Research Program and the State Key Laboratory of Complex & Critical Software Environment (SKLCCSE). Zimu Zhou's research is supported by Chow Sang Sang Group Research Fund No. 9229139. Jinliang Deng's research is supported by Theme-based Research Scheme (T45-205/21-N) from Hong Kong RGC, Generative AI Research and Development Centre from InnoHK, and the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No.VRLAB2024A02). Corresponding authors are Yongxin Tong and Jinliang Deng.

References

[1] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*. IEEE, 2014, pp. 106–112.

[2] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 95–104.

[3] X. Cheng, F. Shi, X. Liu, M. Zhao, and S. Chen, "A novel deep class-imbalanced semisupervised model for wind turbine blade icing detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2558–2570, 2021.

[4] R.-G. Cirstea, T. Kieu, C. Guo, B. Yang, and S. J. Pan, "Enhancenet: Plugin neural networks for enhancing correlated time series forecasting," in *2021 IEEE 37th International Conference on Data Engineering*. IEEE, 2021, pp. 1739–1750.

[5] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, "Autocts: Automated correlated time series forecasting," *Proceedings of the VLDB Endowment*, vol. 15, no. 4, pp. 971–983, 2021.

[6] S. Yan, B. Ding, W. Guo, J. Zhou, Z. Wei, X. Jiang, and S. Xu, "Flashp: an analytical pipeline for real-time forecasting of time-series relational data," *Proceedings of the VLDB Endowment*, vol. 14, no. 5, pp. 721–729, 2021.

[7] X. Wu, D. Zhang, M. Zhang, C. Guo, B. Yang, and C. S. Jensen, "Autocts+: Joint neural architecture and hyperparameter search for correlated time series forecasting," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–26, 2023.

[8] Y. Yao, D. Li, H. Jie, H. Jie, T. Li, J. Chen, J. Wang, F. Li, and Y. Gao, "Simplets: An efficient and universal model selection framework for time series forecasting," *Proceedings of the VLDB Endowment*, vol. 16, no. 12, pp. 3741–3753, 2023.

[9] Z. Shao, F. Wang, Y. Xu, W. Wei, C. Yu, Z. Zhang, D. Yao, T. Sun, G. Jin, X. Cao *et al.*, "Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[10] J. Deng, X. Chen, R. Jiang, X. Song, and I. W. Tsang, "A multi-view multi-task learning framework for multi-variate time series forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7665–7680, 2022.

[11] Z. Lai, D. Zhang, H. Li, C. S. Jensen, H. Lu, and Y. Zhao, "Lightcts: A lightweight framework for correlated time series forecasting," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–26, 2023.

[12] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong, "Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2296–2306.

[13] J. Deng, X. Chen, R. Jiang, D. Yin, Y. Yang, X. Song, and I. W. Tsang, "Disentangling structured components: Towards adaptive, interpretable and scalable time series forecasting," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[14] J. Deng, F. Ye, D. Yin, X. Song, I. Tsang, and H. Xiong, "Parsimony or capability? decomposition delivers both in long-term time series forecasting," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[15] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan, "Towards spatio-temporal aware traffic time series forecasting," in *2022 IEEE 38th International Conference on Data Engineering*. IEEE, 2022, pp. 2900–2913.

[16] B. Wang, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng, and G. Zhang, "Deep uncertainty quantification: A machine learning approach for weather forecasting," in *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2087–2095.

[17] S. He, X. Li, L. Trenary, B. A. Cash, T. DelSole, and A. Banerjee, "Learning and dynamical models for sub-seasonal climate forecasting: Comparison and collaboration," in *Proceedings of the AAAI conference on artificial intelligence*, 2022, pp. 4495–4503.

[18] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[19] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations*, 2023.

[20] S. Guo, Y. Lin, L. Gong, C. Wang, Z. Zhou, Z. Shen, Y. Huang, and H. Wan, "Self-supervised spatial-temporal bottleneck attentive network for efficient long-term traffic forecasting," in *2023 IEEE 39th International Conference on Data Engineering*. IEEE, 2023, pp. 1585–1596.

[21] Y. Li, X. Lu, H. Xiong, J. Tang, J. Su, B. Jin, and D. Dou, "Towards long-term time-series forecasting: Feature, pattern, and distribution," in *2023 IEEE 39th International Conference on Data Engineering*. IEEE, 2023, pp. 1611–1624.

[22] R. Zha, L. Zhang, S. Li, J. Zhou, T. Xu, H. Xiong, and E. Chen, "Scaling up multivariate time series pre-training with decoupled spatial-temporal representations," in *2024 IEEE 40th International Conference on Data Engineering*. IEEE, 2024, pp. 667–678.

[23] Y. Jiang, X. Li, Y. Chen, S. Liu, W. Kong, A. F. Lentzakis, and G. Cong, "SAGDFN: A scalable adaptive graph diffusion forecasting network for multivariate time series forecasting," in *International Conference on Data Engineering*. IEEE, 2024, pp. 1255–1268.

[24] X. He, Y. Li, J. Tan, B. Wu, and F. Li, "Oneshotstl: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting," *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1399–1412, 2023.

[25] Y. Cheng, P. Chen, C. Guo, K. Zhao, Q. Wen, B. Yang, and C. S. Jensen, "Weakly guided adaptation for robust time series forecasting," *Proceedings of the VLDB Endowment*, vol. 17, no. 4, pp. 766–779, 2023.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.

[28] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning*. PMLR, 2022, pp. 27 268–27 286.

[29] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *International Conference on Learning Representations*, 2023.

[30] J. Chen, J. E. Lenssen, A. Feng, W. Hu, M. Fey, L. Tassiulas, J. Leskovec, and R. Ying, "From similarity to superiority: Channel clustering for time series forecasting," *Advances in Neural Information Processing Systems*, vol. 37, pp. 130 635–130 663, 2025.

[31] Z. Dong, R. Jiang, H. Gao, H. Liu, J. Deng, Q. Wen, and X. Song, "Heterogeneity-informed meta-parameter learning for spatiotemporal time series forecasting," in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, 2024, pp. 631–641.

[32] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert systems with applications*, vol. 140, p. 112896, 2020.

[33] R. Zuo, G. Li, B. Choi, S. S. Bhowmick, D. N.-y. Mah, and G. L. Wong, "Svp-t: a shape-level variable-position transformer for multivariate time series classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 11 497–11 505.

[34] R. Zuo, G. Li, R. Cao, B. Choi, J. Xu, and S. S. Bhowmick, "Darker: Efficient transformer with data-driven attention mechanism for time series," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3229–3242, 2024.

[35] I. Loshchilov, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[36] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[37] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in *International Conference on Learning Representations*, 2023.

[38] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 11 121–11 128.

[39] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 753–763.

[40] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 1907–1913.

[41] J. Shi, Q. Ma, H. Ma, and L. Li, "Scaling law for time series forecasting," *arXiv preprint arXiv:2405.15124*, 2024.

[42] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[43] X. Qiu, J. Hu, L. Zhou, X. Wu, J. Du, B. Zhang, C. Guo, A. Zhou, C. S. Jensen, Z. Sheng *et al.*, "Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods," *Proceedings of the VLDB Endowment*, vol. 17, no. 9, pp. 2363–2377, 2024.

[44] J. Deng, X. Chen, R. Jiang, X. Song, and I. W. Tsang, "St-norm: Spatial and temporal normalization for multi-variate time series forecasting," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 269–278.

[45] J. Hu, B. Yang, C. Guo, C. S. Jensen, and H. Xiong, "Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks," in *2020 IEEE 36th International Conference on Data Engineering*. IEEE, 2020, pp. 1417–1428.

[46] X. Chen, J.-Y. Jiang, K. Jin, Y. Zhou, M. Liu, P. J. Brantingham, and W. Wang, "Reliable: Offline reinforcement learning for tactical strategies in professional basketball games," in *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022, pp. 3023–3032.

[47] X. Chen, W.-Y. Wang, Z. Hu, D. Reynoso, K. Jin, M. Liu, P. J. Brantingham, and W. Wang, "Playbest: Professional basketball player behavior synthesis via planning with diffusion," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 4406–4413.

[48] L. Kilian and H. Lütkepohl, *Structural vector autoregressive analysis*. Cambridge University Press, 2017.

[49] L. Li, K. Wang, S. Li, X. Feng, and L. Zhang, "Lst-net: Learning a convolutional neural network with a learnable sparse transform," in *European Conference on Computer Vision*. Springer, 2020, pp. 562–579.

[50] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, vol. 12, 2016.

[51] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 804–17 815, 2020.

[52] Y. Cui, K. Zheng, D. Cui, J. Xie, L. Deng, F. Huang, and X. Zhou, "Metro: a generic graph neural network framework for multivariate time series forecasting," *Proceedings of the VLDB Endowment*, vol. 15, no. 2, pp. 224–236, 2021.

[53] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2021.