MR-Transformer: Multi-Resolution Transformer for Multivariate Time Series Prediction

Siying Zhu, Jiawei Zheng, Qianli Ma†, *Member, IEEE*

***Abstract*—Multivariate Time Series (MTS) prediction has been studied broadly, which is widely applied in real-world applications. Recently, Transformer-based methods have shown the potential in this task for their strong sequence modeling ability. Despite progress, these methods pay little attention to extracting short-term information in the context, while short-term patterns play an essential role in reflecting local temporal dynamics. Moreover, we argue that there are both consistent and specific characteristics among multiple variables, which should be fully considered for multivariate time series modeling. To this end, we propose a Multi-Resolution Transformer (MR-Transformer) for MTS prediction, modeling multivariate time series from both the temporal and the variable resolution. Specifically, for the temporal resolution, we design a long short-term Transformer. We first split the sequence into non-overlapping segments in an adaptive way and then extract short-term patterns within segments, while long-term patterns are captured by the inherent attention mechanism. Both of them are aggregated together to capture the temporal dependencies. For the variable resolution, besides the variable-consistent features learned by long short-term Transformer, we also design a temporal convolution module to capture the specific features of each variable individually. MR-Transformer enhances the multivariate time series modeling ability by combining multi-resolution features between both time steps and variables. Extensive experiments conducted on real-world time series datasets show that MR-Transformer significantly outperforms the state-of-the-art MTS prediction models. The visualization analysis also demonstrates the effectiveness of the proposed model.**

***Index Terms*—Multivariate time series, Multi-step forecasting, Transformer-based model, Dynamic temporal wrapping (DTW).**

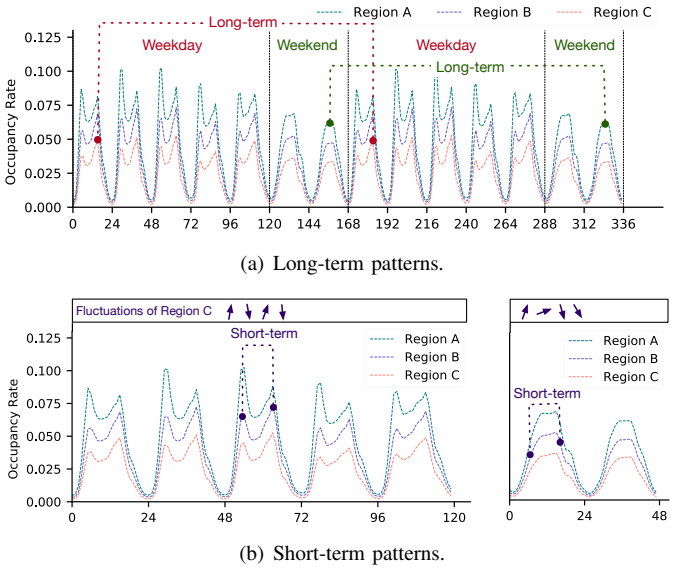
I. Introduction

MULTIVARIATE time series (MTS) are ubiquitous in the modern manufacturing industry, sensor networks and information services. MTS prediction helps decision-making with the estimated forecasts of metrics or events, which is widely applied in various real-world fields such as financial markets [46, 44, 2], energy management [55, 14], traffic analysis [12, 33] and climate modeling [41]. It is a challenging problem, as both intra-series temporal dependencies and inter-series correlations need to be modeled jointly. In other words, each variable depends not only on its historical values but also on other variables.

Qianli Ma is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China (e-mail: qianlima@scut.edu.cn, corresponding author).

Siying Zhu is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China (e-mail: siyingzhu99@163.com).

Jiawei Zheng is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510006, China (e-mail: csjwzheng@foxmail.com).



(a) Long-term patterns.

(b) Short-term patterns.

Fig. 1: The hourly occupancy rate of three regions in San Francisco Bay Area for 2 weeks. Examples of long-term and short-term patterns are marked with dotted lines.

Traditional time series forecasting methods including auto-regressive models like Auto-Regressive Integrated Moving Average (ARIMA) [5], latent state-based models like Kalman Filters (KF) [17] and others have been extensively studied. However, these methods assume a linear dependence over time, and they are not well-suited to model long-term dependencies. Due to the ability of modeling long-term and non-linear relationships for large-scale and complicated sequential data, deep learning models have achieved better performance than these traditional methods.

Existing deep learning models for time series forecasting can be divided into three categories: Recurrent Neural Network (RNN)-based models [35, 42, 21, 39], Temporal Convolutional Network (TCN)-based models [4, 40] and Transformer-based models [45, 23, 49, 56, 52]. The first two categories face challenges in capturing long-term dependencies in time series data. RNN-based methods easily suffer from gradient vanishing and exploding problems, while TCN-based methods require deeper layers to achieve larger local receptive fields. For both of them, information must pass through a long path between two distant temporal locations.

Recently, Transformer-based methods have shown great potential in time series forecasting. The self-attention mechanism in Transformer allows for modeling relationships between

any pair of elements, regardless of their distance, making it more suitable for capturing long-term dependencies [26, 23]. As a result, researchers have explored the value of Transformers in time series analysis, achieving promising results on certain tasks, particularly on long sequence time series forecasting (LSTF) [56, 52]. However, these Transformer-based methods pay little attention to extracting short-term information in the context, while short-term patterns play an important role in reflecting local temporal dynamics. For example, Fig. 1 displays the evolution of the occupancy rate over three selected regions in San Francisco Bay Area. For each region, i.e., each individual time series, there are two time scale patterns, weekly and daily. The former long-term patterns (Fig. 1(a)) reflect the workday and weekend modes that repeat over an extended period, while the latter short-term patterns (Fig. 1(b)) portray the characteristics within short temporal slices. The fluctuations vary across different temporal slices, with more intense dynamics during the weekdays, with morning and evening peaks, and smoother changes on the weekends. Taking both kinds of temporal patterns into account, we aim to design a long short-term Transformer for more accurate time series forecasting.

Moreover, as manifested in Fig 1, the MTS of different regions exhibit some similar trends, but each has its own unique characteristics such as differences in scales and dynamic patterns. In this work, we view variable-consistent features as the impact imposes a similar effect on all variables, while variable-specific features as the impact affects individual variable specifically. Fully incorporating both the variable-consistent and variable-specific characteristics can help capture the complementary features of MTS. However, existing Transformer-based models mainly focus on modeling variable-consistent features while neglecting the specific features of each variable.

To address the aforementioned limitations, we propose Multi-Resolution Transformer (MR-Transformer) for MTS prediction, modeling multivariate time series from a more comprehensive perspective. For the temporal resolution, we design a Long Short-Term Transformer that utilizes the classic Transformer encoder-decoder architecture to capture long-term temporal dependencies via the vanilla attention mechanism. To capture the short-term temporal patterns, we first split the sequence into non-overlapping segments in an adaptive manner and then extract the features within each segment. Note that the segmenting approach is adaptive rather than fixed or pre-defined. Specifically, we employ Dynamic Time Warping (DTW) [38] to perform semantic segmentation automatically by aligning the representation of the sequence and the prototypical features of the segments, which are trainable and optimized simultaneously with the network parameters. For the variable resolution, variable-consistent and variable-specific features reflect disentangled effects on multiple variables. Besides the variable-consistent features learned by the Long Short-Term Transformer, we design a Temporal Convolution module to learn a set of variable-specific dynamics individually. In this way, MR-Transformer is able to produce a more comprehensive representation for MTS prediction by combining both variable-consistent and variable-specific features. The major contributions of this paper are summarized as follows:

- We take both the long-term patterns and short-term patterns into account and design a Long Short-Term Transformer for more accurate time series forecasting.
- Apart from the temporal resolution, we also consider the variable resolution. We fully consider both variable-consistent and variable-specific features and thus enhance the multivariate time series modeling ability by combining multi-resolution features between both time steps and variables.
- We verify the model effectiveness via extensive experimental evaluation on real-world MTS datasets, achieving competing performance compared with existing state-of-the-art methods. The visualization analysis shows our model can effectively capture different time series dependencies. Furthermore, we demonstrate MR-Transformer is also generally applicable to univariate time series datasets.

The remainder of this paper is organized as follows. Section II reviews related works on MTS prediction and Transformer-based forecasting models. Section III introduces our MR-Transformer model in detail. In Section IV, we demonstrate the effectiveness of MR-Transformer empirically. Finally, Section VI summarizes the paper.

II. Related Work

A. Time Series Forecasting

Time series forecasting is a long-standing problem in the literature, which has received significant attention over the years. Early works are based on classical models such as Vector Auto-Regression (VAR) [28] and Auto-Regressive Integrated Moving Average (ARIMA) [5]. However, these classical methods assume a linear dependence over time, and are not well-suited to model long-term dependencies.

With the development of deep learning, neural networks have shown stronger modeling ability than traditional models, especially in capturing non-linear data features. Recurrent Neural Networks (RNNs) [7] are particularly suitable for time series forecasting due to their advantages in sequence learning. Among them, DeepAR [39] proposes an RNN-based auto-regressive model to predict the probabilistic future distribution. LSTNet [21] introduces CNN with recurrent-skip connections to capture both long-term and short-term dependencies. There are also some works [42, 35, 43, 15] that introduce temporal attention to RNN to achieve long-range dependencies for better prediction performance. Despite progress, RNN-based methods easily suffer from gradient vanishing and exploding problems and are hard to work in parallel. Recently, Temporal Convolution Networks (TCNs) [32, 3] have been proposed with convolutional architecture for sequence modeling tasks. Benefiting from the well-designed architecture with a parallel mechanism, TCN-based models [40, 4] are effective in capturing local temporal features and have achieved good results. However, for both RNN-based and TCN-based models, signals must pass through a long path between two far-away temporal

locations, which can make it difficult to capture long-term dependencies in time series data.

In recent years, hybrid models have emerged as a promising approach for improving time series forecasting performance by incorporating multiple modules. For instance, DReF [31] combines a statistical model (such as ARIMA or SARIMA) for performing linear forecasts and a selected machine learning model for forecasting the residual errors. Self-supervised LSTM-DeepFM [36] proposes a hybrid model that combines LSTM, deep, and FM components to extract temporal, high-dimensional, and low-dimensional features, respectively. HyDCNN [24] utilizes position-aware dilated CNNs and an autoregressive model to capture the sequential non-linear and linear dependencies. Additionally, frequency information has been found to be effective in improving forecasting performance. Within the framework of the ATFN [53], the time-domain block learns the trend feature, while the frequency-domain block learns the complicated periodic features of time series. Hybrid models can overcome the limitations of individual established methods and achieve superior forecasting results.

An MTS is a collection of univariate time series, where there are temporal dependencies within each series and correlations across multiple variables. Various methods have been proposed to tackle MTS prediction. TRMF [54] assumes that MTS has a low-rank structure, while some studies [25, 35] leverage attention mechanisms to learn correlations among multiple variables. ST-Norm [9] introduces temporal normalization (TN) and spatial normalization (SN) modules to refine high-frequency and local components separately. CATN [13] proposes a tree-embedding approach to learn inter-series correlations and multi-level dependency learning to capture temporal patterns for intra-series data. MCTAN [37] employs a channel attention mechanism to weigh contributions from different channels and extracts temporal relations through an attention mechanism. Furthermore, recent studies [40, 15, 11] have adopted a hybrid perspective that combines global and local modules to respectively extract shared patterns among all variables and individual variable patterns. Inspired by these works, we disentangle variable-consistent and variable-specific characteristics and integrate them for variable resolution modeling.

B. Transformer-based Models

Transformer [45] was originally proposed as a sequence-to-sequence model in natural language processing (NLP) to deal with machine translation. Due to its powerful and flexible modeling capabilities, it has been widely applied in processing non-sequential data such as images in computer vision (CV) tasks [6, 27]. In addition to its significant success in NLP and CV, researchers have explored the adoption of Transformer to solve time series forecasting problems [23, 49, 56, 52, 48].

With the self-attention mechanism, Transformer directly models the relationships between any element pairs regardless of distance, making it potentially more suitable for capturing long-term dependencies. However, applying attention to long-term time series forecasting can be computationally prohibitive due to the quadratic increase in time and memory requirements with sequence length. To address this challengeLogTrans [23] introduces the local convolution to Transformer and proposes the LogSparse attention to select time steps following the exponentially increasing intervals. Informer [56] defines a sparsity measurement for queries and uses ProbSparse attention to select dominant queries. Reformer [20] introduces local-sensitive hashing to approximate attention by allocating similar queries, leading to significant complexity reduction. Moreover, subsequent Transformer-based models such as AutoFormer [52] and FEDformer [57] further improve the performance of Autoformer by incorporating series decomposition and frequency transform to extract relevant features. Nonetheless, these Transformer-based methods tend to overlook the importance of extracting multi-resolution information from time series, including variable correlations and short-term information within the context.

In this work, we propose an adaptive segment attention mechanism to explore the short-term information within segments. Our method differs from the above method in explicitly discovering different segment-wise patterns. Moreover, we design a Temporal Convolution module to extract variable-specific features, which is combined with the variable-consistent features captured by the Long Short-Term Transformer.

III. Method

The basic idea of the proposed method Multi-Resolution Transformer (MR-Transformer) is to model multivariate time series from a comprehensive perspective, for both temporal and variable resolutions.

The framework of MR-Transformer is illustrated in Fig. 2. It mainly consists of two modules for learning variable-consistent and variable-specific patterns, respectively. In the variable-consistent module, multivariate time series are processed by the **Long Short-Term Transformer** to extract both long-term and short-term temporal features. To discover various short-term patterns, we carefully design an adaptive segment attention mechanism, which first splits the sequence into non-overlapping segments in an adaptive way and then extracts short-term patterns within segments. In the variable-specific module, each univariate time series is fed independently into the **Temporal Convolution**, producing the unique temporal patterns for each variable. Finally, we aggregate the representations from both variant-consistent and variant-specific modules for final predictions.

A. Problem Statement

In this paper, we focus on multivariate time series forecasting. Under the rolling forecasting setting with a fixed size window, we have a sequence of historical T time steps of observations, denoted as $\boldsymbol{X}_{1:T} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$, where $\boldsymbol{x}_t \in \mathbb{R}^N$ and N is the variable dimension. Our goal is to predict a sequence of future values $\{\boldsymbol{x}_{T+1}, \boldsymbol{x}_{T+2}, \ldots, \boldsymbol{x}_{T+h}\}$ where h is the desirable horizon. Let $\boldsymbol{Z}_{T+h} = \{z_1, z_2, ..., z_{T+h}\}$ be associated time-dependent covariate vectors that are assumed

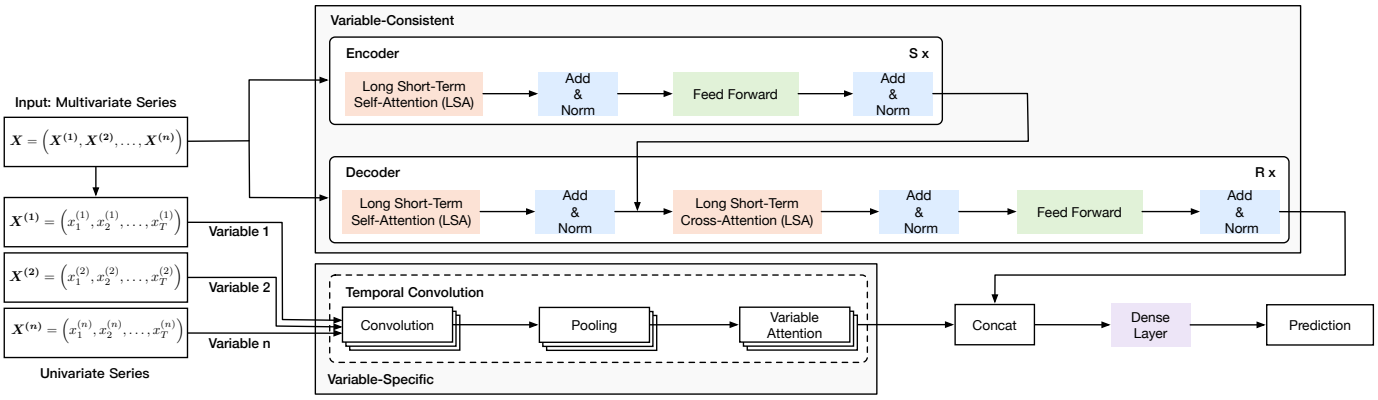


Fig. 2: The overall framework of Multi-Resolution Transformer (MR-Transformer), composed of variable-consistent and variable-specific modules. The multivariate time series are fed into **Long Short-Term Transformer** to extract both long-term and short-term temporal features, while each univariate time series is independently fed into the **Temporal Convolution** to extract unique temporal features of a single variable. The features from the two modules are aggregated to produce multi-resolution representations for prediction.

to be known over the entire time. The MR-Transformer takes the following form:

$$\widehat{\boldsymbol{X}}_{T+1:T+h} = f(\boldsymbol{X}_{1:T}, \boldsymbol{Z}_{T+h}; \Phi), \tag{1}$$

where f is the forecasting model and Φ denotes the learnable parameters of the model.

B. Long Short-Term Attention (LSA)

Long Short-Term Attention (LSA) plays a crucial role in the MR-Transformer model by capturing both long-term and short-term temporal dependencies in time series. We refer to both Long Short-Term Self-Attention and Long Short-Term Cross-Attention as Long Short-Term Attention (LSA) and use the term LSA directly in the paper for brevity and clarity.

The process begins by mapping the time series embedding $\boldsymbol{X}_{emb}$ into different subspaces:

$$\boldsymbol{Q} = \boldsymbol{X}_{emb}\boldsymbol{W}_q, \boldsymbol{K} = \boldsymbol{X}_{emb}\boldsymbol{W}_k, \boldsymbol{V} = \boldsymbol{X}_{emb}\boldsymbol{W}_v, \tag{2}$$

where $\boldsymbol{Q} \in \mathbb{R}^{T\times d_q}, \boldsymbol{K} \in \mathbb{R}^{T\times d_k}, \boldsymbol{V} \in \mathbb{R}^{T\times d_v}$ represent queries, keys and values in attention mechanism [45]. $\boldsymbol{W}_q \in \mathbb{R}^{d_{model}\times d_q}, \boldsymbol{W}_k \in \mathbb{R}^{d_{model}\times d_k}, \boldsymbol{W}_v \in \mathbb{R}^{d_{model}\times d_v}$ denote learnable projection matrices. Typically, we set the dimensions d_q, d_k, and d_v to be the same as d_{model}. Throughout the rest of the paper, we will use d_{model} to represent all three dimensions.

As illustrated in Fig. 3, the long-term representation of time series is extracted through the vanilla attention mechanism, which allows the model to access any part of the input sequence, regardless of the distance, making it potentially suitable for capturing recurring patterns with long-term dependencies [23, 10, 26]. In addition, the short-term representation is extracted through the adaptive segment attention mechanism.

1) Long-term Attention: The long-term attention leverages vanilla attention mechanism [45] to learn the long-term representation $\boldsymbol{M}_{long}$. For each attention layer, the calculation process is similar. Hence we omit the layer index for simplicity. Specifically, we calculate temporal correlations between time steps by scaled dot-product, as follows:

$$\boldsymbol{e} = \frac{\boldsymbol{Q}\,(\boldsymbol{K})^T}{\sqrt{d_k}}, \tag{3}$$

where $\boldsymbol{e} \in \mathbb{R}^{T\times T}$ is the attention score matrix with element e_{ij} representing the attention score between $\boldsymbol{Q}_i$ and $\boldsymbol{K}_j$, and d_k is the dimension of keys $\boldsymbol{K}$.

Subsequently, softmax$(\cdot)$ is applied to every element of $\boldsymbol{e}$ to obtain normalized scores as attention weights, as follows:

$$\boldsymbol{\alpha}_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T}\exp\left(e_{ik}\right)}, \boldsymbol{\alpha}_{ij} \in \boldsymbol{A}_{long}, \tag{4}$$

where $\boldsymbol{\alpha}_{ij}$ represents the attention weight between i-th time step and j-th time step and $\boldsymbol{A}_{long} \in \mathbb{R}^{T\times T}$ is the attention weight matrix. Finally, the output of the attention layer is calculated as:

$$\boldsymbol{M}_{long} = \boldsymbol{A}_{long}\boldsymbol{V}, \tag{5}$$

where $\boldsymbol{M}_{long} \in \mathbb{R}^{T\times d_{model}}$ can be viewed as the long-term representation of the input sequence.

2) Short-term Attention: As for the short-term patterns extraction, we design the an adaptive segment attention mechanism, which first splits the input sequence into multiple segments and then performs short-term attention within segments. The adaptive segment attention layer aims to learn short-term representation $\boldsymbol{M}_{short}$.

A straightforward strategy for segmentation is to evenly partition a whole time series into shorter ones of the same fixed length in a static manner, but it has several limitations. First, the segmentation aims to discover distinct short-term patterns which are internally homogeneous. Thus the length of each optimal time series segment is not the same as the others' in most cases. Furthermore, time points from different sample instances are not generally temporally aligned, making it difficult to find absolute temporal locations for segmentation. To tackle the challenges mentioned above, we propose an adaptive way to divide an input sequence into disjoint segments.

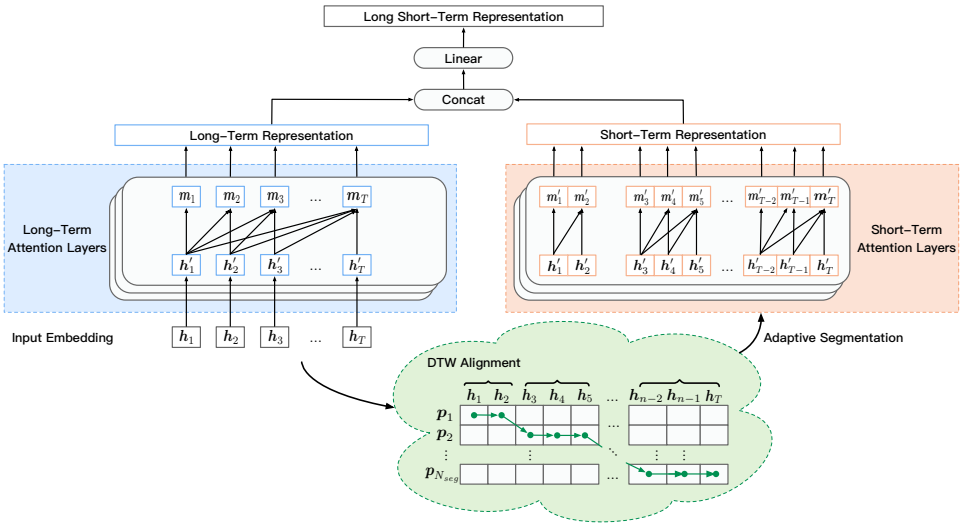


Fig. 3: Illustration of Long Short-Term Attention (LSA). The long-term representation is extracted by multi-head attention mechanism operating on the whole sequence, while the short-term representation is extracted by segment-wise attention that operates within non-overlapping subsequences. For splitting the sequence, an adaptive segmentation strategy is utilized. In this example, given the input embedding $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, ..., \boldsymbol{h}_T\}$, we utilize the prototypical hidden series $\boldsymbol{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{N_s}\}$ to perform DTW-alignment. According to the binary alignment path, we obtain the segmentation results: 1-st segment: $\{\boldsymbol{h}'_1, \boldsymbol{h}'_2\}$, 2-nd segment: $\{\boldsymbol{h}'_3, \boldsymbol{h}'_4, \boldsymbol{h}'_5\}$, and N_s-th segment: $\{\boldsymbol{h}'_{T-2}, \boldsymbol{h}'_{T-1}, \boldsymbol{h}'_T\}$. By aggregating long-term representation $\boldsymbol{M}_{long} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, ..., \boldsymbol{m}_T\}$ and short-term representation $\boldsymbol{M}_{short} = \{\boldsymbol{m}'_1, \boldsymbol{m}'_2, \boldsymbol{m}'_3, ..., \boldsymbol{m}'_T\}$, the long short-term representation is as the output of LSA.

Inspired by [22], we define a prototypical hidden series $\boldsymbol{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{N_s}\} \in \mathbb{R}^{N_s \times d_s}$ of length N_s which best summarizes the high-level features of N_s segments, and d_s is the dimension of segment features. For temporal segmentation, we utilize Dynamic Time Warping (DTW) [38] to determine the optimal alignment between a target sequence and the prototypical hidden series. DTW is a popular technique for measuring the discrepancy between two time series of different length, based on point-to-point matching with the temporal consistency. Given the prototypical series $\boldsymbol{P}$ and target sequence $\boldsymbol{H}$ of length N_s and T, the DTW distance is defined as:

$$\mathrm{DTW}(\boldsymbol{P}, \boldsymbol{H}) = \min\left\{\langle \mathbf{B}, \boldsymbol{\Delta}(\boldsymbol{P}, \boldsymbol{H})\rangle, \boldsymbol{B} \in \mathcal{B}\right\}, \tag{6}$$

where $\boldsymbol{H}$ is the hidden representation in our model. $\mathcal{B} \subset \{0,1\}^{N_s \times T}$ is the set of possible binary alignment matrices whose (n,t)-th entry indicates whether $\boldsymbol{p}_n$ and $\boldsymbol{h}_t$ are aligned or not. The total alignment cost is defined by the inner product of the cost matrix $\boldsymbol{\Delta} \in \mathbb{R}^{N_s \times T}$ and the binary matrix $\boldsymbol{B} \in \mathbb{R}^{N_s \times T}$. Specifically, $\boldsymbol{\Delta}$ is the alignment cost matrix whose (n,t)-th entry $\delta(\boldsymbol{p}_n, \boldsymbol{h}_t)$ encodes the distance between $\boldsymbol{p}_n$ and $\boldsymbol{h}_t$. We define the cost by using their cosine similarity as follows:

$$\delta(\boldsymbol{p}_n, \boldsymbol{h}_t) = 1 - \frac{\boldsymbol{p}_n \cdot \boldsymbol{h}_t}{\|\boldsymbol{p}_n\|_2 \|\boldsymbol{h}_t\|_2}. \tag{7}$$

The goal of DTW is to find the optimal alignment matrix between $\boldsymbol{P}$ and $\boldsymbol{H}$, which minimizes their total alignment cost with the time consistency among the aligned point pairs. The optimal alignment matrix $\boldsymbol{B}^*$ is obtained by:

$$\boldsymbol{B}^* = \arg\min\left\{\langle \mathbf{B}, \boldsymbol{\Delta}(\boldsymbol{P}, \boldsymbol{H})\rangle, \boldsymbol{B} \in \mathcal{B}\right\}. \tag{8}$$

However, the DTW is known to be non-differentiable. We use the smooth min operator $\min_\gamma \{\beta_1, \ldots, \beta_{\dot{n}}\}$ with smoothing parameter $\gamma > 0$ to define our differentiable soft-DTW distance [8]:

$$\begin{aligned} \mathrm{DTW}_\gamma(\boldsymbol{P}, \boldsymbol{H}) &= \min_\gamma \left\{\langle \mathbf{B}, \boldsymbol{\Delta}(\boldsymbol{P}, \boldsymbol{H})\rangle, \mathbf{B} \in \mathcal{B}\right\}, \\ \min_\gamma \{\beta_1, \ldots, \beta_{\dot{n}}\} &= \begin{cases} \min\limits_{i \le \dot{n}} \beta_i, & \gamma = 0 \\ -\gamma \log \sum_{i=1}^{\dot{n}} e^{-\beta_i/\gamma}, & \gamma > 0 \end{cases}. \end{aligned} \tag{9}$$

To ensure that each time point is aligned only with a single segment, we follow [22] to impose an additional constraint on each alignment matrix in $\mathcal{B}$, such that it represents a single path connecting the upper-left $(1,1)$-th entry to the lower-right (N_s, T)-th entry using only $\rightarrow$, $\searrow$ moves.

Based on the optimal alignment matrix, we can split the input sequence into multiple segments, as shown in Fig. 3. Assume that $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are divided into non-overlapping subsequences (segments) by using the above adaptive segmentation method as Equation (10). We note that the $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are shared with the long-term attention. Except for the prototypical hidden series, the short-term attention does not introduce extra parameters.

$$\begin{aligned} \mathrm{AdaSeg}(\boldsymbol{Q}, N_s) &= \{\boldsymbol{Q}^1, \boldsymbol{Q}^2, ..., \boldsymbol{Q}^{N_s}\}, \boldsymbol{Q}^i \in \mathbb{R}^{L^i \times d_q}, \\ \mathrm{AdaSeg}(\boldsymbol{K}, N_s) &= \{\boldsymbol{K}^1, \boldsymbol{K}^2, ..., \boldsymbol{K}^{N_s}\}, \boldsymbol{K}^i \in \mathbb{R}^{L^i \times d_k}, \\ \mathrm{AdaSeg}(\boldsymbol{V}, N_s) &= \{\boldsymbol{V}^1, \boldsymbol{V}^2, ..., \boldsymbol{V}^{N_s}\}, \boldsymbol{V}^i \in \mathbb{R}^{L^i \times d_v}, \end{aligned} \tag{10}$$

where $\mathrm{AdaSeg}(\boldsymbol{Q}, N_s)$ is the segmentation operation, which outputs N_s non overlapping subsequences. Each segmented subsequence $\boldsymbol{Y}^i$ has a length L^i, and similarly, $\boldsymbol{K}$ and $\boldsymbol{V}$ are also segmented.

To extract short-term features within each segment, we perform segment-wise attention. The calculation is as follows:

$$\begin{aligned} \boldsymbol{e}^i &= \frac{\boldsymbol{Q}^i \left(\boldsymbol{K}^i\right)^T}{\sqrt{d_k}} \in \mathbb{R}^{L^i \times L^i}, i = \{1, 2, ..., N_s\}, \\ \boldsymbol{\alpha}_{mn} &= \frac{\exp\left(\boldsymbol{e}^i_{m,n}\right)}{\sum_{k=1}^{L^i} \exp\left(\boldsymbol{e}^i_{m,k}\right)}, \boldsymbol{\alpha}_{mn} \in \boldsymbol{A}^i_{short}, \\ \boldsymbol{M}^i_{short} &= \boldsymbol{A}^i_{short} \boldsymbol{V}^i \in \mathbb{R}^{L^i \times d_{model}}, i = \{1, 2, ..., N_s\}, \end{aligned} \tag{11}$$

where $\boldsymbol{e}^i$ is the attention score matrix for the i-th subsequence, $\boldsymbol{\alpha}_{mn}$ is the attention weight between the m-th and n-th time steps within the i-th subsequence, and $\boldsymbol{A}^i_{short}$ denotes the attention weight matrix for the i-th subsequence. $\boldsymbol{M}^i_{short}$ represents the local representation of the i-th subsequence. The short-term representation $\boldsymbol{M}_{short}$ is obtained by concatenating the local representations of all subsequences along the time dimension:

$$\boldsymbol{M}_{short} = \{\boldsymbol{M}^1_{short}, \boldsymbol{M}^2_{short}, ..., \boldsymbol{M}^{N_s}_{short}\}. \tag{12}$$

Finally, the long-term and short-term features of the sequence are fused to obtain the long short-term representation, as follows:

$$\mathrm{LSA}(\boldsymbol{X}_{emb}, \boldsymbol{X}_{emb}, \boldsymbol{X}_{emb}) = \mathrm{FC}\left(\left(\boldsymbol{M}_{long} \parallel \boldsymbol{M}_{short}\right)\right), \tag{13}$$

where $\mathrm{FC}(\cdot)$ denotes the fully connected layer and $\parallel$ is the concatenation operator.

C. Variable-Consistent Long Short-Term Transformer

In this section, we describe the general architecture of the Long Short-Term Transformer, which consists of the encoder and decoder stacks.

1) Embedding Layer: In the Transformer model, the input representation should contain the position information since other main components of the model are entirely invariant to sequence order. The original Transformer model [45] uses absolute positional embedding, which is added to the word embedding at the bottoms of the encoder and decoder stacks. Similarly, in our work, we fuse different encoding schemes, including positional encoding, temporal encoding, and value encoding. Let $\boldsymbol{X}$ be the input time series and $\boldsymbol{Z}$ be the covariate vectors. We omit time subscript here, the time series embedding is summed as follows:

$$\boldsymbol{X}_{emb} = E_{pos}(\boldsymbol{X}) + E_{temporal}(\boldsymbol{Z}) + E_{value}(\boldsymbol{X}), \tag{14}$$

where $E_{pos}(\boldsymbol{X})$, $E_{temporal}(\boldsymbol{Z})$ and $E_{value}(\boldsymbol{X})$ denote positional embedding, temporal embedding and value embedding, respectively. In particular, we utilize sinusoidal positional encoding to represent positions. To extract time information, a fully connected network is employed to embed the time-dependent covariate vectors $\boldsymbol{Z}$. The value encoding uses 1D convolution to extract timestamp-level features from the time series.

2) Encoder: The MR-Transformer encoder is composed of S identical layers. Each layer consists of long short-term self-attention (LSA) and feedforward network (FFN), both followed by a residual connection and layer normalization to facilitate better information flow and prevent vanishing gradients. Let $\boldsymbol{H}^{(0)}_{en}$ denote the input embedding $\boldsymbol{X}_{emb}$.

For the l-th encoding layer, the input is the output of the $(l-1)$-th encoding layer, which can be formulated as:

$$\begin{aligned} \boldsymbol{H}^{(l),1}_{en} &= \mathrm{AddNorm}\left(\mathrm{LSA}\left(\boldsymbol{H}^{(l-1)}_{en}, \boldsymbol{H}^{(l-1)}_{en}, \boldsymbol{H}^{(l-1)}_{en}\right)\right), \\ \boldsymbol{H}^{(l)}_{en} &= \mathrm{AddNorm}\left(\mathrm{FFN}\left(\boldsymbol{H}^{(l),1}_{en}\right)\right), \end{aligned} \tag{15}$$

where $\mathrm{AddNorm}(\cdot)$ represents the residual connection with layer normalization, and $\mathrm{FFN}(\cdot)$ denotes a multi-layer feedforward network as widely adopted in [45, 56, 52]. After stacking S layers, $\boldsymbol{H}^{(S)}_{en}$ is the output of the encoder, which is fed to the long short-term cross-attention of the decoding layers.

3) Decoder: The MR-Transformer decoder has similar composition to the encoder, which is composed of stacked R decoding layers. It also includes an additional long short-term cross-attention. Motivated by the generative style decoder of the Informer model [56], the decoder also receives the reformatting sequence that combines the subsequence of inputs and a padding sequence, and instantly predicts output elements in a generative style instead of in an auto-regressive way. Specifically, the input of the decoder is a part of the input of the encoder, i.e., $\boldsymbol{X}_{t_s:T}$, where t_s is the start time of the input of the decoder and $1 \le t_s \le T$. $\boldsymbol{X}_{t_s:T}$ is converted to $\boldsymbol{H}^0_{de} \in \mathbb{R}^{(T-t_s)\times d_{\text{model}}}$ after passing through the embedding layer of the decoder. $\boldsymbol{H}^0_{de}$ with dimension d_{model} is fed into the first decoding layer. For the l-th decoding layer, the input is the $(l-1)$-th hidden representation of the decoding layer and the output of the encoder. The process of calculating the output of the l-th decoding layer is as follows:

$$\begin{aligned} \boldsymbol{H}^{(l),1}_{de} &= \mathrm{AddNorm}\left(\mathrm{LSA}\left(\boldsymbol{H}^{(l-1)}_{de}, \boldsymbol{H}^{(l-1)}_{de}, \boldsymbol{H}^{(l-1)}_{de}\right)\right), \\ \boldsymbol{H}^{(l),2}_{de} &= \mathrm{AddNorm}\left(\mathrm{LSA}\left(\boldsymbol{H}^{(l),1}_{de}, \boldsymbol{H}^{(S)}_{en}, \boldsymbol{H}^{(S)}_{en}\right)\right), \\ \boldsymbol{H}^{(l)}_{de} &= \mathrm{AddNorm}\left(\mathrm{FFN}\left(\boldsymbol{H}^{(l),2}_{de}\right)\right). \end{aligned} \tag{16}$$

After stacking R layers, $\boldsymbol{H}^{(R)}_{de}$ is the output of the decoder.

We note that there exist three kinds of attention mechanisms in MR-Transformer, i.e., encoder self-attention, decoder self-attention and cross-attention connecting the encoder and the decoder, as shown in Fig. 2. During the encoding process, the model uses Long Short-Term Self-Attention to obtain $\boldsymbol{Q}^{(l)}_{en}, \boldsymbol{K}^{(l)}_{en}, \boldsymbol{V}^{(l)}_{en}$ for encoder self-attention. Similarly, during the decoding process, the model uses Long Short-Term Self-Attention to obtain $\boldsymbol{Q}^{(l)}_{de}, \boldsymbol{K}^{(l)}_{de}, \boldsymbol{V}^{(l)}_{de}$ for decoder self-attention, as follows:

$$\boldsymbol{Q}^{(l)}_{en} = \dot{\boldsymbol{W}}_q \boldsymbol{H}^{(l-1)}_{en}, \boldsymbol{K}^{(l)}_{en} = \dot{\boldsymbol{W}}_k \boldsymbol{H}^{(l-1)}_{en}, \boldsymbol{V}^{(l)}_{en} = \dot{\boldsymbol{W}}_v \boldsymbol{H}^{(l-1)}_{en}, \tag{17}$$

$$\boldsymbol{Q}^{(l)}_{de} = \ddot{\boldsymbol{W}}_q \boldsymbol{H}^{(l-1)}_{de}, \boldsymbol{K}^{(l)}_{de} = \ddot{\boldsymbol{W}}_k \boldsymbol{H}^{(l-1)}_{de}, \boldsymbol{V}^{(l)}_{de} = \ddot{\boldsymbol{W}}_v \boldsymbol{H}^{(l-1)}_{de}, \tag{18}$$

where $\dot{\boldsymbol{W}}_q, \dot{\boldsymbol{W}}_k, \dot{\boldsymbol{W}}_v$ and $\ddot{\boldsymbol{W}}_q, \ddot{\boldsymbol{W}}_k, \ddot{\boldsymbol{W}}_v$ denote learnable projection matrices.

To handle input sequences with different lengths and temporal misalignment in the encoder and decoder, we introduce

TABLE I: Statistics of the 13 multivariate datasets.

Datasets	ETTh1	ETTh2	ETTm1	SMLm1	SMLm2	Appliance	Yahoo	SCITOS G5	DSIM	Traffic	Solar-Energy	Electricity	Exchange-Rate
Variants	7	7	7	10	10	26	6	24	16	862	137	321	8
Timesteps	17,420	17,420	69,680	2,764	1,373	6,579	6,041	5,456	1,440	17,544	52,560	26,304	7,588
Granularity	1hour	1hour	15min	15min	15min	30min	1day	-	-	1hour	10min	1hour	1day
Start time	7/1/2016	7/1/2016	7/1/2016	3/13/2012	4/18/2012	1/11/2016	8/26/1996	-	-	1/1/2015	1/1/2006	1/1/2012	1/1/1990
Task type	Multi-step			Multi-step						Single-step			
Data partition	Follow [56]			Training/Validation/Testing: 6/2/2									

two prototypical hidden series $\boldsymbol{P}_{en}$ and $\boldsymbol{P}_{de}$ to segment the input sequences of the encoder and the decoder, respectively. Specifically, in encoder self-attention, $\boldsymbol{Q}_{en}^{(l)}, \boldsymbol{K}_{en}^{(l)}, \boldsymbol{V}_{en}^{(l)}$ are segmented according to $\boldsymbol{P}_{en}$. Similarly, in decoder self-attention, $\boldsymbol{Q}_{de}^{(l)}, \boldsymbol{K}_{de}^{(l)}, \boldsymbol{V}_{de}^{(l)}$ are segmented according to $\boldsymbol{P}_{de}$. However, in the cross-attention, the calculation is as follows:

$$\boldsymbol{Q}_{cross}^{(l)} = \ddot{\boldsymbol{W}}_q \boldsymbol{H}_{de}^{(l),1}, \boldsymbol{K}_{cross}^{(l)} = \ddot{\boldsymbol{W}}_k \boldsymbol{H}_{en}^{(S)}, \boldsymbol{V}_{cross}^{(l)} = \ddot{\boldsymbol{W}}_v \boldsymbol{H}_{en}^{(S)}, \tag{19}$$

where $\ddot{\boldsymbol{W}}_q, \ddot{\boldsymbol{W}}_k, \ddot{\boldsymbol{W}}_v$ denote learnable projection matrices. The keys $\boldsymbol{K}_{cross}^{(l)}$ and values $\boldsymbol{V}_{cross}^{(l)}$ are both transformed from the encoder output $\boldsymbol{H}_{en}^{(S)}$, thus they are divided by aligning $\boldsymbol{P}_{en}$ to ensure that each segment corresponds to a relevant part of the encoder input sequence. Meanwhile, the queries $\boldsymbol{Q}_{cross}^{(l)}$ are segmented by $\boldsymbol{P}_{de}$.

D. Variable-Specific Temporal Convolution

Modeling multiple variable dynamics can be challenging because it requires capturing both the shared temporal patterns between the variables and the unique temporal patterns of each individual variable. To address this challenge, we use 1D convolution to capture the individual temporal patterns of each variable. Specifically, for the i-th variable, we apply a set of convolution filters $F^{(i)} = \{F_1^{(i)}, F_2^{(i)}, \ldots, F_{N_F}^{(i)}\}$ to its input $\boldsymbol{X}^{(i)}$. Here, N_F is the number of filters in the set. The convolution filters sweep through the input and produce a set of feature maps, as follows:

$$\dot{\boldsymbol{H}}_{tc}^{(i)} = \mathrm{TC}\left(\boldsymbol{X}^{(i)}, F^{(i)}\right), \tag{20}$$

where $\mathrm{TC}(\cdot)$ denotes the temporal convolution operation and $\dot{\boldsymbol{H}}_{tc}^{(i)} \in \mathbb{R}^{N_F}$ represents the the output specific to each variable. In the temporal convolution operation, filters of size N_k and stride 1 are convolved with the input, with appropriate padding to ensure that the output has the same length as the input. This operation is used to capture temporal relations in each univariate time series and produce a sequence of feature maps. To further extract salient features from the feature maps, we employ a max pooling operation that aggregates the output of the convolution filters across time and selects the maximum value for each filter. The resulting vector representation captures the most important features of the input with respect to the corresponding filter.

Multivariate time series often exhibits complex dependencies between different variables. To capture the specific features of a single variable, we also consider the correlations between the variable and others. In existing literature, the primary tool used to capture similarities among multiple variables is the graph neural network (GNN) [50, 51]. However, in multivariate time series analysis, accurately quantifying the internal dependencies among variables is challenging, and constructing an adjacency matrix is difficult. Inspired by the effectiveness of self-attention [25, 35, 37], we introduce a variable attention layer to model the dependencies between different variables. This layer takes the form $\boldsymbol{H}_{tc} = \mathrm{Attention}\left(\dot{\boldsymbol{H}}_{tc}, \dot{\boldsymbol{H}}_{tc}, \dot{\boldsymbol{H}}_{tc}\right)$, and the calculation is as follows:

$$\boldsymbol{Q}_{\boldsymbol{tc}} = \widetilde{\boldsymbol{W}}_q \dot{\boldsymbol{H}}_{tc}, \boldsymbol{K}_{\boldsymbol{tc}} = \widetilde{\boldsymbol{W}}_k \dot{\boldsymbol{H}}_{tc}, \boldsymbol{V}_{\boldsymbol{tc}} = \widetilde{\boldsymbol{W}}_v \dot{\boldsymbol{H}}_{tc}, \tag{21}$$

$$\boldsymbol{H}_{tc} = \mathrm{Softmax}\left(\frac{\boldsymbol{Q}_{\boldsymbol{tc}} \boldsymbol{K}_{\boldsymbol{tc}}^{\top}}{\sqrt{d_{tc}}}\right) \boldsymbol{V}_{\boldsymbol{tc}}, \tag{22}$$

where the matrices $\boldsymbol{Q}_{tc}, \boldsymbol{K}_{tc}, \boldsymbol{V}_{tc} \in \mathbb{R}^{N \times N_k}$ represent queries, keys and values in the variable attention mechanism. $\widetilde{\boldsymbol{W}}_q, \widetilde{\boldsymbol{W}}_k, \widetilde{\boldsymbol{W}}_v$ are learnable projection matrices, and d_{tc} is the dimension of $\boldsymbol{K}_{tc}$. $\boldsymbol{H}_{tc} = \{\boldsymbol{H}_{tc}^{(1)}, \boldsymbol{H}_{tc}^{(2)}, \ldots, \boldsymbol{H}_{tc}^{(n)}\}$ denotes variable-specific representation.

E. Forecasting and Learning

We combine the representations from both the variable-consistent and variable-specific modules by linear projections and concatenation. The forecasting results are obtained by:

$$\widehat{\boldsymbol{X}}_{T+1:T+h} = \mathrm{FC}\left(\boldsymbol{W}_1 \boldsymbol{H}_{de}^{(R)} \parallel (\boldsymbol{W}_2 \boldsymbol{H}_{tc})^{\mathrm{T}}\right), \tag{23}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d_{model} \times N}$, $\boldsymbol{W}_2 \in \mathbb{R}^{N_k \times h}$ are learnable projection matrices.

In the learning phase, our objective is to minimize the mean squared error (MSE) [30] between the predicted values and ground truth values, meanwhile optimizing the prototypical representation. To ensure effective segmentation, the prototypical hidden series $\boldsymbol{P} = \{\boldsymbol{P}_{en}, \boldsymbol{P}_{de}\}$ should be optimized so that its i-th vector learns the latent features corresponding to the i-th segment. Given a training set, we obtain the prototypical hidden series by minimizing its soft-DTW distance from the encoder and decoder representations of all the time series instances:

$$\begin{aligned}\mathcal{L}_{\mathrm{proto}}(\boldsymbol{P}) = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \Big(&\mathrm{DTW}_{\gamma}\left(\boldsymbol{P}_{en}, \boldsymbol{H}_{en}^{(S)}\right) \\ &+\mathrm{DTW}_{\gamma}\left(\boldsymbol{P}_{de}, \boldsymbol{H}_{de}^{(R)}\right),\end{aligned} \tag{24}$$

where N_{train} is the number of training samples. Since soft-DTW is differentiable with respect to its input, we can easily optimize $\boldsymbol{P}$ using its gradients $\nabla_{\boldsymbol{P}} \boldsymbol{DTW}_{\gamma}$.

Suppose $\hat{\boldsymbol{x}}_t$ is the prediction and $\boldsymbol{x}_t$ is the aligned ground truth at time stamp t, We adopt the MSE as the objective function for prediction:

$$\mathcal{L}_{\mathrm{pred}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{N_{train} h} \sum_{i=1}^{N_{train}} \sum_{j=1}^{h} \left(\boldsymbol{x}_j^i - \hat{\boldsymbol{x}}_j^i\right)^2, \tag{25}$$

TABLE II: Performance comparison on the multi-step forecasting datasets. The best results are highlighted in bold and the second best results are underlined. The last row is the winning count. A lower MSE or MAE indicates a better prediction.

Methods		MR-Transformer		Informer		LSTNet		Transformer		LogTrans		Reformer		LSTMa		TPA-LSTM		N-BEATS		TCN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	**0.354**	**0.396**	0.577	0.549	1.293	0.901	0.652	0.574	0.686	0.604	0.991	0.754	0.650	0.624	0.901	0.640	1.285	0.913	0.772	0.662
	48	**0.410**	**0.434**	0.685	0.625	1.456	0.960	0.728	0.719	0.766	0.757	1.313	0.906	0.702	0.675	0.978	0.688	1.300	0.915	0.862	0.829
	168	**0.652**	**0.569**	0.931	0.752	1.997	1.214	0.952	0.804	1.002	0.846	1.824	1.138	1.212	0.867	1.697	1.214	1.318	0.924	1.127	0.927
	336	**0.971**	**0.739**	1.128	0.873	2.655	1.369	1.294	0.904	1.362	0.952	2.117	1.280	1.424	0.994	1.994	1.392	1.505	1.025	1.532	1.043
	720	**1.046**	**0.748**	1.215	0.896	2.143	1.380	1.327	1.226	1.397	1.291	2.415	1.520	1.960	1.322	2.744	1.851	3.475	1.014	1.572	1.414
ETTh2	24	**0.253**	**0.366**	0.720	0.665	2.742	1.457	0.787	0.713	0.828	0.750	1.531	1.613	1.143	0.813	3.139	1.351	3.475	1.448	0.932	0.822
	48	**0.439**	**0.504**	1.457	1.001	3.567	1.687	1.716	0.982	1.806	1.034	1.871	1.735	1.671	1.221	3.156	1.360	3.442	1.439	2.032	1.133
	168	**1.930**	**1.093**	3.489	1.515	3.242	2.513	3.867	1.597	4.070	1.681	4.660	1.846	4.117	1.674	5.764	2.344	3.422	1.430	4.579	1.841
	336	**2.479**	**1.285**	2.723	1.340	2.544	2.591	3.681	1.675	3.875	1.763	4.028	1.688	3.434	1.549	4.808	2.169	3.397	1.419	4.360	1.931
	720	**2.943**	**1.380**	3.467	1.473	4.625	3.709	3.717	1.474	3.913	1.552	5.381	2.015	3.963	1.788	5.548	2.503	3.439	1.430	4.403	1.700
ETTm1	24	**0.108**	**0.218**	0.323	0.369	1.968	1.170	0.398	0.391	0.419	0.412	0.724	0.607	0.621	0.629	0.869	0.881	3.520	1.463	0.471	0.451
	48	**0.173**	**0.286**	0.494	0.503	1.999	1.215	0.482	0.554	0.507	0.583	1.098	0.777	1.392	0.939	1.949	1.315	3.429	1.437	0.570	0.639
	96	**0.264**	**0.370**	0.678	0.614	2.762	1.542	0.730	0.752	0.768	0.792	1.433	0.945	1.339	0.913	1.875	1.278	3.402	1.430	0.864	0.868
	288	**1.028**	**0.600**	1.056	0.786	1.257	2.076	1.389	1.254	1.462	1.320	1.820	1.094	1.740	1.124	2.436	1.574	3.345	1.414	1.645	1.446
	672	**0.946**	**0.778**	1.192	0.926	1.917	2.941	1.586	1.388	1.669	1.461	2.187	1.232	2.736	1.555	3.830	2.177	3.502	1.452	1.980	1.144
SMLm1	4	**0.039**	**0.100**	0.223	0.365	0.389	0.512	0.510	0.562	0.741	0.700	0.889	0.840	0.428	0.563	0.396	0.510	1.289	0.798	0.833	0.767
	12	**0.232**	**0.248**	0.424	0.498	0.440	0.554	0.737	0.696	0.749	0.713	0.899	0.856	0.484	0.609	0.427	0.536	1.442	0.883	0.843	0.781
	24	**0.228**	**0.316**	0.619	0.611	0.614	0.647	0.852	0.760	0.816	0.738	0.979	0.885	0.675	0.711	0.622	0.650	1.709	1.004	0.918	0.808
SMLm2	4	**0.096**	**0.207**	1.970	1.209	2.121	1.251	2.763	1.382	2.974	1.459	3.569	1.751	2.333	1.376	2.082	1.242	2.596	1.347	3.346	1.599
	12	**0.599**	**0.563**	3.115	1.490	2.211	1.298	3.247	1.510	3.028	1.482	3.634	1.778	2.432	1.428	2.130	1.269	2.451	1.237	3.407	1.623
	24	**1.754**	**1.026**	4.086	1.707	2.636	1.406	3.644	1.591	3.087	1.482	3.704	1.778	2.899	1.547	2.235	1.313	2.213	1.136	3.473	1.623
Appliance	4	**0.241**	**0.283**	0.520	0.814	2.560	1.192	1.947	1.073	2.444	1.204	2.932	1.445	2.816	1.311	2.549	1.178	3.515	1.526	2.749	1.319
	12	**0.557**	**0.497**	1.799	1.010	2.581	1.217	2.166	1.130	2.517	1.222	3.021	1.466	2.840	1.338	2.567	1.206	3.567	1.538	2.832	1.338
	24	**0.738**	**0.622**	2.191	1.129	2.727	1.285	2.289	1.157	2.409	1.199	2.891	1.438	3.000	1.413	2.577	1.221	3.965	1.622	2.710	1.313
DSIM	4	**0.090**	**0.218**	1.108	0.716	1.873	0.938	1.874	0.987	2.610	1.115	3.133	1.338	2.060	1.032	1.863	0.927	0.164	0.279	2.937	1.221
	12	**0.134**	**0.259**	1.180	0.743	1.936	0.965	1.553	0.884	2.663	1.119	3.195	1.343	2.130	1.061	1.915	0.940	0.172	0.282	2.996	1.226
	24	**0.384**	**0.353**	1.361	0.793	2.064	1.001	1.830	0.964	2.748	1.135	3.298	1.362	2.270	1.101	2.023	0.968	0.401	0.373	3.092	1.243
SCITOS G5	4	**0.350**	**0.332**	0.352	0.341	0.725	0.632	0.517	0.459	0.667	0.555	0.800	0.666	0.797	0.695	0.727	0.634	2.083	0.890	0.750	0.608
	12	**0.444**	**0.398**	0.465	0.420	0.773	0.646	0.529	0.466	0.660	0.551	0.791	0.661	0.850	0.710	0.787	0.659	1.977	0.859	0.742	0.603
	24	**0.502**	**0.438**	0.537	0.454	0.785	0.651	0.515	0.439	0.658	0.550	0.790	0.660	0.864	0.716	0.837	0.678	1.860	0.842	0.741	0.603
Yahoo	4	**0.247**	**0.202**	0.493	0.468	1.990	1.226	0.617	0.527	1.458	1.004	1.750	1.204	2.189	1.349	1.832	1.185	3.466	1.821	1.641	1.099
	12	**0.277**	**0.238**	0.660	0.588	1.996	1.234	0.588	0.512	1.332	0.962	1.598	1.155	2.196	1.357	1.830	1.185	3.983	1.956	1.498	1.054
	24	**0.320**	**0.301**	0.921	0.750	1.999	1.232	0.661	0.552	1.622	1.082	1.946	1.298	2.199	1.355	1.978	1.212	4.241	1.969	1.825	1.185
P-value		-	-	4.7E-03	3.1E-04	3.7E-09	2.5E-08	2.5E-04	5.7E-06	7.0E-06	2.4E-08	2.3E-08	6.3E-12	2.9E-07	5.3E-10	2.4E-07	3.3E-09	3.2E- 11	1.9E-10	1.8E-06	1.1E-09
Average Rank		**1.0**	**1.0**	2.8	2.6	4.1	4.0	6.4	7.2	5.2	5.3	8.2	8.1	6.4	6.4	6.6	6.4	7.8	6.9	6.6	6.9
Best Count		**66**		0																	

where h is the length of the prediction horizon. All the parameters, including the prototypical hidden series and the network parameters are optimized simultaneously by minimizing the corresponding losses. The update process can be described as follows:

$$\begin{aligned} \boldsymbol{P} &\leftarrow \boldsymbol{P} - \eta \cdot \partial \mathcal{L}_{\text{proto}} / \partial \boldsymbol{P}, \\ \Phi &\leftarrow \Phi - \eta \cdot \partial \mathcal{L}_{\text{pred}} / \partial \Phi, \end{aligned} \tag{26}$$

where η is the learning rate.

IV. Experiment Setup

A. Datasets

We evaluated our approach on on 13 multivariate time series datasets, namely *ETTh1*, *ETTh2*, *ETTm1*, *Traffic*, *SolarEnergy*, *Electricity*, *ExchangeRate*, *SMLm1*, *SMLm2*, *Appliance*, *Yahoo*, *SCITOS G5* and *DSIM*, ranging from various domains, including energy, finance and traffic domains. During data pre-processing, we perform zero-score normalization. Their properties are summarized in Table I and the details of these datasets are described in Section A of the Supplementary Material.

B. Metrics

In order to provide a comprehensive evaluation of our method, we utilize two commonly-used metrics: Mean Absolute Errors (MAE) [16] and Mean Squared Errors (MSE) [30]. For single-step forecasting tasks, we calculate these metrics based on the last step prediction, while for multi-step forecasting tasks, we average the metrics over all step predictions. The MAE and MSE are defined as follows:

$$\text{MAE} = \frac{1}{N_{test} h} \sum_{i=1}^{N_{test}} \sum_{j=1}^{h} \left| \hat{\boldsymbol{x}}_j - \boldsymbol{x}_j \right|, \tag{27}$$

$$\text{MSE} = \frac{1}{N_{test} h} \sum_{i=1}^{N_{test}} \sum_{j=1}^{h} \left(\hat{\boldsymbol{x}}_j - \boldsymbol{x}_j \right)^2, \tag{28}$$

where N_{test} is the total number of testing samples. $\hat{\boldsymbol{x}}_j$ denotes the model's predicted value at the j-th time step, while $\boldsymbol{x}_j$ represents the corresponding ground-truth value.

C. Implementation

Baselines: To validate the effectiveness of our proposed model, we conducted experiments to compare MR-Transformer with three categories of multivariate time series forecasting methods: **(1) Transformer-based methods**, including Informer [56], Reformer [19], LogTrans [23], Transformer [45]. **(2) RNN-based methods**, including LSTNet [21], TPA-LSTM[42], LSTMa [1] and DARNN [35]. **(3) CNN-based methods**, TCN [3]. Additionally, we compared our model with N-BEATS [34], which achieves state-of-the-art performance on the large-scale M4 benchmark dataset [29]. It is worth noting that all experiments were conducted on multivariate forecasting.

TABLE III: Performance comparison on the single-step forecasting datasets.

Methods		MR-Transformer		Informer		LSTNet		Transformer		Reformer		LogTrans		LSTMa		DARNN		TPA-LSTM		N-BEATS		TCN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Traffic	4	**0.537**	**0.304**	0.693	0.372	1.036	0.617	0.783	0.457	0.875	0.504	0.729	0.420	1.018	0.595	1.066	0.571	1.084	0.649	0.954	0.550	0.820	0.460
	12	**0.532**	**0.322**	0.669	0.358	1.038	0.620	0.787	0.459	0.940	0.558	0.783	0.465	1.023	0.596	1.080	0.571	1.092	0.641	1.024	0.609	0.881	0.510
	24	**0.517**	**0.316**	0.699	0.371	1.052	0.631	0.787	0.460	0.948	0.541	0.790	0.451	1.024	0.598	1.058	0.559	1.097	0.646	1.033	0.590	0.889	0.494
	48	**0.545**	**0.335**	0.824	0.440	1.090	0.658	0.772	0.447	0.946	0.528	0.788	0.440	1.004	0.582	0.850	0.492	1.179	0.696	1.031	0.575	0.887	0.482
	96	**0.459**	**0.307**	0.743	0.390	1.138	0.666	0.755	0.433	1.001	0.549	0.834	0.458	0.982	0.563	0.831	0.477	1.224	0.711	1.091	0.599	0.939	0.501
Solar-Energy	4	**0.028**	**0.081**	0.039	0.103	0.036	0.084	0.109	0.209	0.100	0.204	0.084	0.170	0.142	0.272	0.101	0.180	0.431	0.614	0.109	0.222	0.094	0.186
	12	**0.074**	**0.135**	0.083	0.153	0.078	0.143	0.142	0.247	0.206	0.299	0.172	0.249	0.185	0.321	0.182	0.265	0.490	0.657	0.225	0.326	0.194	0.273
	24	**0.148**	**0.210**	0.165	0.216	0.184	0.217	0.229	0.298	0.256	0.348	0.214	0.290	0.298	0.387	0.247	0.338	0.575	0.693	0.280	0.379	0.240	0.318
	48	**0.199**	0.252	0.212	**0.240**	0.278	0.273	0.225	0.312	0.285	0.353	0.237	0.294	0.292	0.405	0.247	0.343	0.628	0.731	0.310	0.385	0.267	0.323
	96	**0.200**	0.258	0.211	**0.244**	0.215	0.261	0.225	0.308	0.251	0.332	0.209	0.277	0.292	0.400	0.247	0.339	0.548	0.683	0.274	0.362	0.235	0.303
Electricity	4	0.144	0.263	0.290	0.376	**0.122**	**0.224**	0.320	0.394	0.419	0.507	0.349	0.422	0.416	0.512	0.603	0.575	0.675	0.609	0.457	0.552	0.393	0.462
	12	0.155	0.265	0.290	0.368	**0.146**	**0.246**	0.329	0.403	0.413	0.499	0.344	0.416	0.428	0.523	0.576	0.566	0.683	0.612	0.450	0.544	0.387	0.456
	24	0.154	0.254	0.286	0.365	**0.146**	**0.240**	0.330	0.399	0.445	0.517	0.371	0.431	0.429	0.519	0.498	0.509	0.684	0.612	0.485	0.564	0.417	0.472
	48	**0.169**	**0.268**	0.324	0.391	0.223	0.307	0.352	0.417	0.444	0.520	0.370	0.434	0.457	0.542	0.387	0.458	0.713	0.634	0.484	0.567	0.416	0.475
	96	**0.176**	**0.279**	0.339	0.409	0.218	0.314	0.366	0.426	0.464	0.537	0.386	0.448	0.476	0.554	0.402	0.469	0.724	0.644	0.505	0.586	0.435	0.490
Exchange-Rate	4	**0.014**	**0.081**	0.436	0.495	0.018	0.094	1.506	0.968	2.356	1.309	1.963	1.091	3.163	2.033	2.450	1.292	2.497	1.144	2.568	1.427	2.209	1.195
	12	**0.040**	**0.140**	0.555	0.572	0.062	0.179	1.923	1.095	2.367	1.316	1.973	1.097	4.038	2.300	2.621	1.343	2.500	1.151	2.580	1.434	2.219	1.201
	24	**0.120**	**0.256**	1.199	0.826	0.123	0.267	2.034	1.136	2.648	1.376	2.207	1.146	4.272	2.385	2.591	1.323	2.505	1.158	2.886	1.499	2.483	1.256
	48	0.530	0.530	2.021	1.104	**0.243**	**0.382**	3.315	1.404	3.684	1.636	3.070	1.363	6.962	2.948	2.664	1.358	2.527	1.186	4.015	1.783	3.454	1.493
	96	1.788	0.962	2.282	1.209	**0.416**	**0.518**	3.302	1.405	4.034	1.734	3.362	1.445	6.935	2.951	3.633	1.546	2.548	1.204	4.397	1.890	3.783	1.583
P-value		-	-	7.7E-02	4.4E-02	6.0E-01	3.6E-01	1.5E-02	3.9E-03	5.5E-03	7.3E-04	1.1E-02	3.9E-03	8.9E-03	1.8E-03	3.3E-03	6.4E-04	6.7E-05	6.6E-09	4.0E-03	8.0E-04	7.0E-03	1.7E-03
Average Rank		**1.3**	**1.4**	2.8	2.6	4.0	3.9	4.3	4.5	7.6	7.6	4.5	4.5	8.9	9.5	7.6	7.1	9.9	9.3	9.2	9.3	6.2	6.3
Best Count		**28**		2		10		0															

Implementation details: In our experiments, we fixed the number of encoder and decoder layers to 1 and 2, respectively, and observed that MR-Transformer performed well under this setting. With further tuning, the performance could be improved. We employed the Adam optimizer [18] with an initial learning rate of $1e-4$, which was decayed twice every epoch. We set the total number of epochs to 6 and used early stopping to prevent overfitting. Additionally, we conducted a grid-search strategy to find the optimal hyperparameters, selecting the recommended parameters for comparison. We varied the batch size from $\{2, 4, 8, 16, 32\}$, the number of segments from $\{6, 8, 12, 24, 28, 48, 56\}$, and the number of filters in the temporal convolution module from $\{64, 128, 256, 512\}$. The filter size was determined based on the frequency of the dataset. Specifically, we set the filter size to 7 for daily data, 24 for hourly data, 16 for 30-minute data, 32 for 15-minute data, and 48 for 10-minute data. Moreover, we adjusted the look-back window T to accommodate different lengths of the future horizon h, setting $T = 168$ for $h \in \{4, 12, 24\}$ and $T = 336$ for $h \in \{48, 96\}$. It's worth noting that for the *ETT* datasets, we adopted the same look-back window settings as [56]. To reduce the impact of random initialization, we repeated each experiment 5 times and reported the mean results. All experiments were conducted on the PyTorch platform using an Intel Core i7 – 6850K, 3.60-GHz CPU, 64-GB RAM and a GeForce GTX 1080-Ti 11G GPU.

V. Experimental Results

A. Compare with State-of-the-Arts

In this section, we compare MR-Transformer to several state-of-the-art baselines, including multi-step and single-step forecasting scenarios. For quantitative evaluation, we conduct the Wilcoxon signed-rank test [47] to verify the significance of differences between our proposed model and other models. The Wilcoxon signed-rank test is a paired difference test, and P-value < 0.05 indicates significant improvements over the compared model. The comparison results on multi-step forecasting datasets and single-step forecasting are shown in Table II and Table III, respectively.

Multi-Step Forecasting Scenario As shown in Table II, MR-Transformer achieves the best performance among all methods. Note that among these datasets, *ETT* datasets are used to verify the effectiveness of LTSF tasks. Compared with RNN-based methods such as LSTMa and LSTnet, we can observe that Transformer-based methods achieve better results, which demonstrates that Transformer-based methods are better at capturing long-term patterns, thus suffering lower prediction errors. In this case, MR-Transformer also surpasses other Transformer-based methods, especially the latest Informer model. We credit it to the consideration of short-term information. Specifically, MR-Transformer is statistically better than all other methods with significant differences.

Single-Step Forecasting Scenario Besides the outstanding performance on multi-step forecasting, MR-Transformer also performs well and outperforms most of the other baselines, except for a few cases.

We contribute these outstanding results to the following reasons.

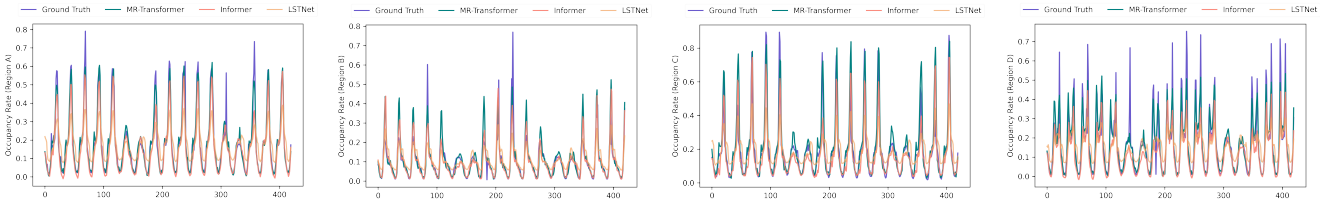
(i) MR-Transformer captures both long-term and short-term temporal patterns and thus better modeling the time series. (ii) Besides the variable-consistent features, MR-Transformer also carefully considers the specific characteristics of each variable and leverages a well-designed module to capture them. (iii) MR-Transformer takes an adaptive way to automatically split temporal segments, which is more flexible to adjust to different short-term patterns in various time series.

We have conducted additional experiments to verify that our techniques can be applied to advanced model like FEDformer [57] and Autoformer [52]. Detailed experimental analysis can be found in Section C of the Supplementary Material.

B. Ablation Studies

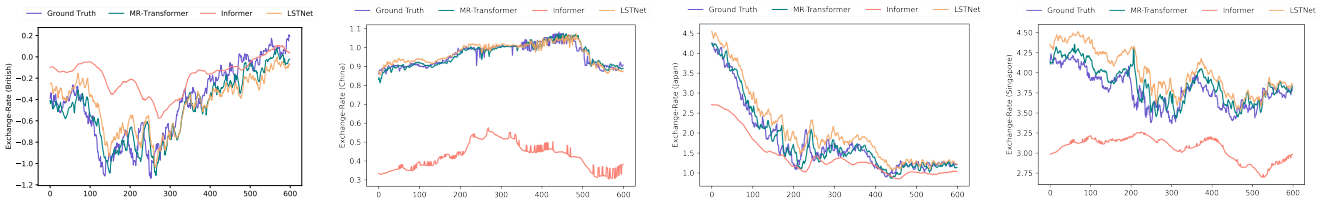
In order to investigate the contribution of each component to the final prediction performance, we design four variants of MR-Transformer, namely:

- **w/o TC**: MR-Transformer without the temporal convolution module that captures variable-specific features. We



(a) Region A (b) Region B (c) Region C (d) Region D

Fig. 4: Prediction results under the $T = 168, h = 12$ setting (*Traffic*).



(a) British (b) China (c) Japan (d) Singapore

Fig. 5: Prediction results under the $T = 168, h = 12$ setting (*Exchange-Rate*).

removed the temporal convolution module and only kept the variable-consistent module.

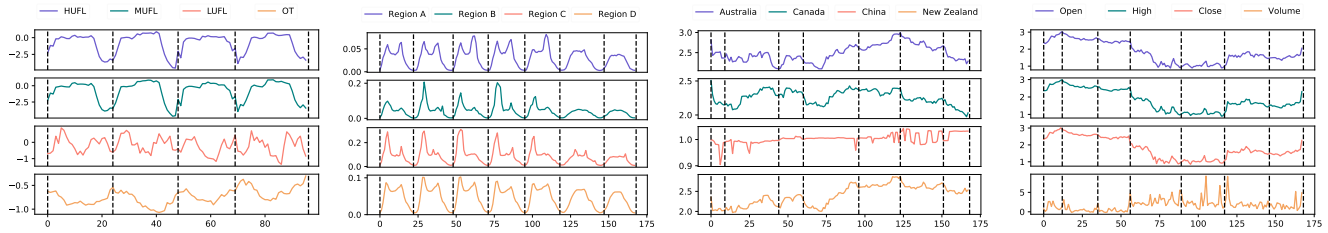
- **w/o Long-term**: MR-Transformer without capturing the long-term information. We removed the long-term attention in LSA and only used the short-term attention.
- **w/o Short-term**: MR-Transformer without explicitly capturing the short-term information. We removed the short-term attention in LSA and only used the vanilla attention.
- **w/o AdaSeg**: MR-Transformer without the adaptive segmentation. We replaced the adaptive segmentation with static segmentation, which is fixed-length segmentation.
- **w/o VariableAttn**: MR-Transformer without the variable attention mechanism in temporal convolution module. We removed the variable attention layer and only kept the convolution layer.

As shown in Table IV, we observe that the full model MR-Transformer achieves significant improvements across all metrics and datasets, which verifies the effectiveness of each component. Specifically, the comparison with **w/o TC** indicates the importance of variable-specific modeling for multivariate time-series forecasting. Moreover, the results of **w/o VariableAttn** demonstrate that capturing variable correlations is crucial for enhancing the performance of the Variable-Specific Temporal Convolution module. The comparison with **w/o Long-term** demonstrates the inclusion of long-term attention improves prediction performance and better captures temporal patterns. The comparison with **w/o Short-term** shows the advantages of leveraging the short-term attention implemented within multiple segments of the sequence. Particularly for longer-term predictions, the effectiveness of long-term attention becomes more prominent. These results verify the benefits of our multi-resolution modeling from a more comprehensive view, which helps to learn complementary information and enhance time series representations. Furthermore, we use a fixed-length segment strategy instead of our adaptive segmentation in **w/o AdaSeg**. The comparison results indicate the superiority of automatically discovering short-term patterns in the adaptive learning way.

TABLE IV: Ablation analysis of MR-Transformer on *ETT* datasets.

Methods	Metric	ETTh1		ETTh2		ETTm1	
		168	336	48	168	24	48
Informer	MSE	0.931	1.128	1.457	3.489	0.323	0.494
	MAE	0.752	0.873	1.001	1.515	0.369	0.503
w/o TC	MSE	0.865	0.975	0.806	2.329	0.196	0.243
	MAE	0.719	0.759	0.707	1.187	0.323	0.368
w/o Long-term	MSE	0.906	1.001	0.961	2.736	0.164	0.260
	MAE	0.741	0.783	0.816	1.306	0.291	0.383
w/o Short-term	MSE	0.871	0.981	1.298	2.963	0.183	0.243
	MAE	0.722	0.773	0.886	1.346	0.314	0.343
w/o AdaSeg	MSE	0.913	0.988	1.037	2.911	0.154	0.263
	MAE	0.753	0.773	0.817	1.319	0.272	0.376
w/o VariableAttn	MSE	0.763	0.970	0.514	2.259	0.126	0.242
	MAE	0.669	0.748	0.593	1.284	0.256	0.337
MR-Transformer	MSE	**0.652**	**0.970**	**0.439**	**1.934**	**0.108**	**0.173**
	MAE	**0.569**	**0.739**	**0.504**	**1.093**	**0.218**	**0.286**

In our detailed analysis of the different variants, we found that short-term attention in LSA and adaptive segmentation contributed more to the overall prediction performance, highlighting the importance of effectively capturing short-term information in Transformer-based models. Additionally, we observed that LSA is particularly effective for longer look-back window sizes, as more information can be extracted from the adaptive segmented subsequences. The introduction of adaptive segmentation provided a self-adaptive approach for generating segments based on dynamic information, using learnable segment-level prototypical series, and this further improved the model's ability to flexibly discover dynamic short-term patterns.



(a) $T = 96, h = 48$ (*ETTh1*) (b) $T = 168, h = 12$ (*Traffic*) (c) $T = 168, h = 24$ (*Exchange-Rate*) (d) $T = 168, h = 12$ (*Yahoo*)

Fig. 6: The segment visualizations on the datasets (a) *ETTh1* (b) *Traffic* (c) *Exchange-Rate* (d) *Yahoo*. The black dotted lines indicate the segment points.

C. Forecast Visualization

In this section, we provide a visualization of our model's prediction performance compared with other methods. Here we randomly select the *Traffic* and *Exchange-Rate* datasets as cases. The prediction results are as shown in Fig. 4 and Fig. 5.

As mentioned earlier, the *Traffic* dataset exhibits clear daily and weekly periodicity patterns. The traffic occupancy patterns differ significantly on weekdays and weekends, and traffic conditions vary greatly at different times of the day. So it is challenging to make accurate predictions.

Fig. 4 shows the occupancy rate evolution of different regions, where the series of different regions share some similar trends while each has its individual characteristics. We observe that Transformer-based methods including MR-Transformer and Informer are capable of distinguishing weekdays and weekends patterns and thus make a more suitable prediction. This suggests that compared to RNN-based models like LSTNet, these methods can look further back and capture long-term dependencies. Furthermore, compared to Informer, MR-Transformer shows more accurate prediction performance, especially in details such as some peaks and troughs. We attribute this to MR-Transformer's multi-resolution design, which includes short-term and variable-specific modeling modules, allowing for a more detailed modeling of the multivariate time series. The *Exchange-Rate* dataset lacks clear periodicity due to its nature as economic data, resulting in considerable uncertainty and making prediction extremely challenging. Consequently, the local temporal dynamics are more influential in determining future values of the time series compared to trend and seasonal information. While Informer is designed to capture long-range dependencies, its emphasis on trend and seasonal patterns in the look-back window may not be sufficient for modeling short-term information, making it less suitable for such scenarios. Conversely, LSTNet's RNN architecture is better suited to capture nearby information. Nevertheless, MR-Transformer's explicit incorporation of a short-term attention mechanism and a variable-specific module enables it to achieve superior prediction performance, with prediction curves that are closer to the ground truth than both Informer and LSTNet. Overall, the strong performance of MR-Transformer on both the *Traffic* and *Exchange-Rate* datasets showcases its ability to handle different types of data and its generalizability to a variety of real-world applications.

D. Segment Visualization

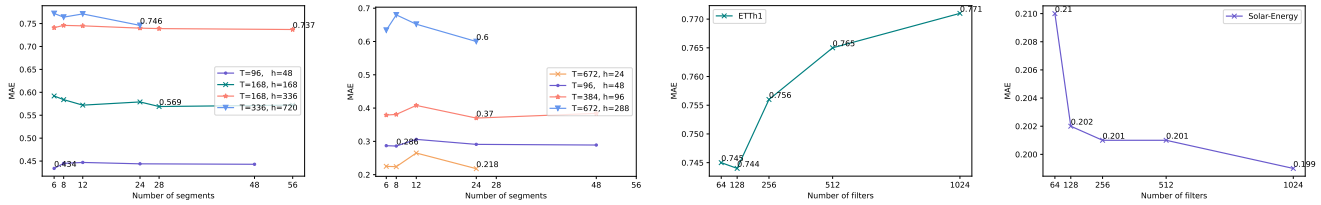
Our model's adaptive segmentation enhances its predictive ability by explicitly identifying different short-term patterns. In this section, we investigate whether our adaptive segmentation can effectively detect meaningful segment points. We present visualizations of the segment points detected by our adaptive segmentation mechanism on various datasets in Fig. 6.

As for the datasets with obvious periodicities, such as *ETTh1* and *Traffic*, shown in Fig. 6 (a) and (b) respectively, the adaptive segmentation mechanism successfully identifies each period, which helps to model the local fluctuations within periods. On the other hand, for the aperiodic datasets such as *Exchange-Rate* and *Yahoo*, shown in Fig. 6 (c) and (d), we observe that the duration of local patterns varies greatly, making segmentation decisions more challenging. This observation highlights the importance of performing segmentation in an adaptive manner rather than using pre-defined or fixed methods. In this case, our adaptive segmentation mechanism successfully detects meaningful segment points. For instance, in the exchange-rate series of China shown in Fig. 6 (c), which exhibits irregular changes in the financial market, the adaptive segmentation mechanism divides the time series into segments based on trend patterns, including several troughs at the first time (1-st and 2-nd segment), a generally stable period (3-rd segment), and periodic rises and falls (6-th segment). Similarly, despite *Yahoo* presents different irregular changes and local patterns as shown in Fig. 6 (d), the adaptive segmentation can identify each segment location based on overall time-variant patterns like the rising process (1-st segment), the decline process (4-th segment).

Currently, our segmentation strategy operates at the sequence level rather than the variable level, meaning that the segmentation results are the same for all variables in a sequence. However, our ultimate goal is to learn a global segment-level representation of all variables, as different variables collected in a multivariate time series scenario are typically correlated. We also plan to explore a more fine-grained segmentation strategy for each variable in the future.

E. Hyper-parameter Analysis

In this section, we investigate the impact of the number of segments N_s in LSA and the number of output channels N_F in the temporal convolution module on the prediction



(a) Segment number N_s (*ETTh1*) (b) Segment number N_s (*ETTm1*) (c) Filter number N_F (*ETTh1*) (d) Filter number N_F (*Solar-Energy*)

Fig. 7: Hyper-parameter study of the number of segments N_s in the long short-term attention (LSA) and the number of output channels N_F in the temporal convolution module.

TABLE V: Statistical evaluation of univariate time series forecasting for our MR-Transformer and the other models.

	MR-Transformer	Informer	Reformer	LogTrans	DeepAR	LSTMa	Prophet	ARIMA
P-value	-	1.7E-02	2.1E-05	3.7E-03	2.3E-03	1.7E-04	8.5E-03	8.5E-05
Average Rank	**1.00**	2.00	7.3	3.3	5.0	5.3	6.4	5.5

performance of our model. The parameter analysis has been conducted on all datasets and shows similar conclusions. Due to the space limitation, we present results in Fig. 7 for the *ETT* and *Solar-Energy* datasets as examples.

Impact of the number of segments. We present the performance of MR-Transformer under different segment numbers in LSA, ranging from 6 to 56. In Fig. 7 (a) and (b), each line represents the MAE value of the prediction performance for a given prediction horizon h and look-back window size T, with the optimal value of N_s (i.e., the number of segments) tagged with the corresponding MAE value. We observe that for longer successive sequences, i.e., larger look-back window sizes T, a larger N_s tends to result in a lower prediction error. This is because a larger number of segments can discover more short-term patterns, which subsequently motivates fine-grained learning within multiple segments.

Impact of the number of filters. We also investigate how the choice of output channel N_F in the temporal convolution module affects the performance of MR-Transformer. The results are presented in Fig. 7(b). We observe that datasets with a small number of variables tend to perform better with smaller N_F values. Specifically, in the case of *ETTh1* dataset with 7 variables, smaller N_F leads to better performance. However, as N_F continues to increase, the prediction performance begins to degrade rapidly due to overfitting problem. On the other hand, for the *Solar-Energy* dataset, which has a large number of variables (137), larger N_F tends to be more beneficial for achieving better prediction performance.

F. Apply to Univariate Time Series

The proposed model is also suitable for univariate time series forecasting scenarios, where the variable-specific module processes only one variable. We conduct experiments on various univariate time series datasets. For all the datasets, we train each predictor five times with different random seeds and report the average MAE. The detailed results are reported in Section A of the Supplementary Material. As shown in Table V, the results demonstrate that our MR-Transformer outperforms other baselines significantly, confirming that the multi-resolution modeling is effective and can be applied to univariate time series forecasting.

VI. Conclusion

This paper proposes a novel Multi-Resolution Transformer (MR-Transformer) model for MTS prediction, modeling multivariate time series from a comprehensive view. For the variable resolution, to better learn the variable-consistent and variable-specific features, we decouple the model into two components: a long short-term Transformer module and a temporal convolution module. For the temporal resolution, we design the long short-term attention (LSA) to better extract short-term patterns. Our extensive experiments show that the proposed MR-Transformer significantly improves the state-of-the-art results on various real-world datasets, including both multivariate and univariate time series predictions. With further analysis and empirical evidence, we show the model can successfully capture different temporal patterns. The limitation of our work is that it is currently unable to handle the data with missing values, which will also be the direction of our future improvement.

Acknowledgments

We thank the anonymous reviewers for their helpful feedbacks. The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 62272173, 61872148), the Natural Science Foundation of Guangdong Province (Grant-Nos. 2022A1515010179, 2019A1515010768), the Science and Technology Planning Project of Guangdong Province (Grant No. 2023A0505050106).

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR*. 2015.

[2] Lu Bai et al. "Entropic dynamic time warping kernels for co-evolving financial time series analysis". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

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

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

[5] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. "Time series analysis: Forecasting and control". In: *Journal of Time* 14.2 (1994), pp. 238–242.

[6] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229.

[7] Jerome Connor, Les E Atlas, and Douglas R Martin. "Recurrent networks and NARMA modeling". In: *Advances in Neural Information Processing Systems, NIPS*. 1992, pp. 301–308.

[8] Marco Cuturi and Mathieu Blondel. "Soft-DTW: a differentiable loss function for time-series". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 894–903.

[9] Jinliang Deng et al. "St-norm: Spatial and temporal normalization for multi-variate time series forecasting". In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 269–278.

[10] Chenyou Fan et al. "Multi-horizon time series forecasting with temporal attention learning". In: *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*. 2019, pp. 2527–2535.

[11] Shoubo Feng et al. "Learning Both Dynamic-Shared and Dynamic-Specific Patterns for Chaotic Time-Series Prediction". In: *IEEE Transactions on Cybernetics* (2020).

[12] Liangzhe Han et al. "Dynamic and Multi-faceted Spatio-temporal Deep Learning for Traffic Speed Forecasting". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 547–555.

[13] Hui He et al. "CATN: Cross Attentive Tree-Aware Network for Multivariate Time Series Forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 4. 2022, pp. 4030–4038.

[14] Keke Huang et al. "Metric Learning-Based Fault Diagnosis and Anomaly Detection for Industrial Data With Intraclass Variance". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[15] Siteng Huang et al. "DSANet: Dual Self-Attention Network for Multivariate Time Series Forecasting". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 2129–2132.

[16] Rob J Hyndman and Anne B Koehler. "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688.

[17] Rudolph Emil Kalman. "A new approach to linear filtering and predicted problems". In: *J Basic Eng* 82 (1960).

[18] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations, ICLR*. 2015.

[19] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The Efficient Transformer". In: *8th International Conference on Learning Representations, ICLR*. 2020.

[20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer". In: *arXiv preprint arXiv:2001.04451* (2020).

[21] Guokun Lai et al. "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.

[22] Dongha Lee, Seonghyeon Lee, and Hwanjo Yu. "Learnable Dynamic Temporal Pooling for Time Series Classification". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*. 2021.

[23] Shiyang Li et al. "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting". In: *Advances in Neural Information Processing Systems, NeurIPS*. 2019, pp. 5243–5253.

[24] Yangfan Li et al. "Modeling temporal patterns with dilated convolutions for time-series forecasting". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16.1 (2021), pp. 1–22.

[25] Yuxuan Liang et al. "GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*. Vol. 2018. 2018, pp. 3428–3434.

[26] Bryan Lim and Stefan Zohren. "Time-series forecasting with deep learning: a survey". In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200209.

[27] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *arXiv preprint arXiv:2103.14030* (2021).

[28] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[29] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M4 Competition: 100,000 time series and 61 forecasting methods". In: *International Journal of Forecasting* 36.1 (2020), pp. 54–74.

[30] Spyros Makridakis et al. "The accuracy of extrapolation (time series) methods: Results of a forecasting competition". In: *Journal of Forecasting* 1.2 (1982), pp. 111–153.

[31] Joao FL de Oliveira, Eraylson G Silva, and Paulo SG de Mattos Neto. "A hybrid system based on dynamic

selection for time series forecasting". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.8 (2021), pp. 3251–3263.

[32] Aaron van den Oord et al. "WaveNet: A Generative Model for Raw Audio". In: *The 9th ISCA Speech Synthesis Workshop*. 2016, p. 125.

[33] Boris N Oreshkin et al. "FC-GAGA: Fully Connected Gated Graph Architecture for Spatio-Temporal Traffic Forecasting". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*. 2021, pp. 9233–9241.

[34] Boris N Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *8th International Conference on Learning Representations, ICLR*. 2019.

[35] Yao Qin et al. "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*. 2017, pp. 2627–2633.

[36] Lei Ren et al. "A data-driven self-supervised LSTM-DeepFM model for industrial soft sensor". In: *IEEE Transactions on Industrial Informatics* 18.9 (2021), pp. 5859–5869.

[37] Lei Ren et al. "MCTAN: A novel multichannel temporal attention-based network for industrial health indicator prediction". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[38] Hiroaki Sakoe and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49.

[39] David Salinas et al. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3 (2020), pp. 1181–1191.

[40] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting". In: *Advances in Neural Information Processing Systems, NeurIPS*. 2019.

[41] Xingjian Shi et al. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems, NIPS*. 2015.

[42] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. "Temporal pattern attention for multivariate time series forecasting". In: *Machine Learning* 108.8 (2019), pp. 1421–1441.

[43] Huan Song et al. "Attend and Diagnose: Clinical Time Series Analysis using Attention Models". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI*. 2018.

[44] Dat Thanh Tran et al. "Temporal attention-augmented bilinear network for financial time-series data analysis". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.5 (2018), pp. 1407–1418.

[45] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems, NIPS*. 2017, pp. 5998–6008.

[46] Heyuan Wang et al. "Hierarchical Adaptive Temporal-Relational Modeling for Stock Trend Prediction". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*. 2021.

[47] Frank Wilcoxon. "Individual comparisons by ranking methods". In: *Breakthroughs in Statistics*. Springer, 1992, pp. 196–202.

[48] Neo Wu et al. "Deep transformer models for time series forecasting: The influenza prevalence case". In: *arXiv preprint arXiv:2001.08317* (2020).

[49] Sifan Wu et al. "Adversarial Sparse Transformer for Time Series Forecasting". In: *Advances in Neural Information Processing Systems, NeurIPS*. 2020.

[50] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[51] Zonghan Wu et al. "Connecting the dots: Multivariate time series forecasting with graph neural networks". In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020, pp. 753–763.

[52] Jiehui Xu, Jianmin Wang, Mingsheng Long, et al. "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting". In: *Advances in Neural Information Processing Systems*. 2021.

[53] Zhangjing Yang et al. "Adaptive temporal-frequency network for time-series forecasting". In: *IEEE Transactions on Knowledge and Data Engineering* 34.4 (2020), pp. 1576–1587.

[54] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. "Temporal regularized matrix factorization for high-dimensional time series prediction". In: *Advances in neural information processing systems, NIPS*. 2016, pp. 847–855.

[55] Fan Zhou et al. "Forecasting the Evolution of Hydropower Generation". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 2861–2870.

[56] Haoyi Zhou et al. "Informer: Beyond efficient transformer for long sequence time-series forecasting". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*. 2021.

[57] Tian Zhou et al. "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 27268–27286.