Electronic Research Archive

https://www.aimspress.com/journal/ERA

ERA, 33(11): 7247–7276.
DOI: 10.3934/era.2025320
Received: 29 July 2025
Revised: 01 November 2025
Accepted: 11 November 2025
Published: 28 November 2025

Research article

A variational mode decomposition assisted temporal-frequency pure convolutional neural network for highway traffic flow forecasting

Yifei Zheng[1], Xiang Wang[2], Xintong Liu[2], Shaoweihua Liu[3,4], Zhiyuan Liu[4] and Kai Huang[2,*]

[1] School of Cyber Science and Engineering, Southeast University, Nanjing, China
[2] School of Instrument Science and Engineering, State Key Laboratory of comprehensive PNT Network and Equipment Technology, Southeast University, Nanjing, China
[3] Zhejiang Communications Investment Group Technology Research Global Co., Ltd., Hangzhou, China
[4] Jiangsu Key Laboratory of Urban ITS, Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, School of Transportation, Southeast University, Nanjing, China

* **Correspondence:** Email: kaihuang@seu.edu.cn.

Abstract: This paper addressed the problem of highway traffic flow multivariate time series forecasting with the challenges of variate heterogeneity. To improve forecast performance without a heavy computational burden in large-scale networks, we innovatively introduced variational mode decomposition and established a decomposition-assisted multi-tasking deep learning forecasting architecture. To improve variate-specific pattern learning and mitigate pattern mixing, we proposed a novel temporal-frequency pure convolutional neural network incorporating discrete Fourier transform, deepwise convolution, and batchwise feedforward neural network. To verify the proposed model, we conducted a case study on a regional network located in Jiangsu, China. Results demonstrate strong forecast performance and efficient computation. The proposed model offers suitability toward highway operators for large-scale engineering deployment and better facilitates their managerial actions.

Keywords: multivariate time series forecasting; deep learning; signal decomposition; frequency learning; convolutional neural network

1. Introduction

Real-time traffic forecasting constitutes an essential task for highway network operators. With economic growth and increased regional integration, traffic volume has witnessed sustained growth over the decades, bringing heavy pressure and environmental challenges to network infrastructure [1]. Accurate forecasting facilitates proactive congestion interventions such as traffic flow regulation and ramp and toll station control strategies. These regional-based models must offer both convincing results and efficient computations to support operators' real-time decision-making.

In recent decades, traffic forecasting has evolved from statistical approaches to data-driven methods incorporating machine learning algorithms. Recent innovations have incorporated methodologies from computer science, such as computer vision (CV), natural language processing (NLP), and large language models (LLMs), though considerable computational challenges remain [2]. However, the dynamic nature of highway systems continuously poses challenges to forecasting works, highlighting the need for models that better apply to practical deployment.

In this paper, we propose a multi-tasking forecasting architecture that deeply explores the potential of signal decomposition algorithms and deep learning methods. To effectively separate the original multivariate time series, variational mode decomposition (VMD) is employed to decompose the signal into multiple sub-signals named intrinsic mode functions (IMFs). Each IMF centers around a specific frequency, which is determined during decomposition. This architecture facilitates multi-tasking and downstream learning, being more understandable and effective. To improve VMD's consistency, a segmentation strategy is proposed. For the learning part, after VMD targets IMF in a trend-dominated pattern, a deepwise one-dimensional temporal convolution network with patching captures temporal dependencies while mitigating unwanted cross-variate mixing and layer overstacking. Targeting the remaining IMFs with oscillatory-dominated patterns, a deepwise one-dimensional frequency convolution network incorporates discrete Fourier transform (DFT) to capture both amplitude and phase patterns in the frequency domain. Both dwTCN and dwFCN are designed with pure convolutions to maintain low computational complexity. To improve the forecast head, a batchwise feedforward neural network (FFN) that initializes batch-independent projection matrices is also incorporated.

1.1. Literature review

Early studies on traffic forecasting primarily apply traffic theory-based methods [3] and statistical-based methods. The ARIMA advances forecasting capability but struggles with nonlinear, non-stationary dynamics [4,5]. Subsequently, data-driven methods such as machine learning methods emerge to address nonlinearities [6], applying methods like kernel ridge regression, support vector machine, and Gaussian process [7]. More recently, deep learning methods have demonstrated superior performance. Among the variety of methodologies, recurrent neural network (RNN) methods, such as Wavenet [8] and LSTM [9], utilize recurrent and gated units to capture temporal dependencies. Multi-layer perceptron (MLP) methods, including Dlinear [10] and TIDE [11], capture them by stacking fully-connected layers. Convolutional neural network (CNN) methods, for example TCN [12], apply convolutional filters to capture temporal dependencies. To address spatial dependencies, graph convolution is also integrated into modeling studies; for example, STGCN [13], GraphWavenet [14], and MTGNN [15] improve typical spatial-temporal learning by including adaptive graph and multiple temporal kernels. In view of the complexity and cross-variate over-smoothing problem, studies using PatchMixer [16], PatchTST [17], and ModernTCN [18] proposed variate-independent learning. These approaches either treat spatial learning as a separate module or disregard it. This strategy has shown improved performance in scenarios with high heterogeneity.

Transformer-based methods have also emerged in recent years and have drawn significant attention. The Vanilla transformer is the most representative benchmark [19]. Other extensions include CrossFormer with cross attention [20], AutoFormer with auto-correlation [21], and informer with sparseity [22]. ASTGCN integrates the self-attention mechanism with graph convolution without the traditional encoder-decoder architecture [23]. Difformer redefines the graph diffusion process by fusing graph attention with graph convolution [24]. To alleviate computational burden, iTransformer restricts the self-attention mechanism to operate solely on the temporal dimension [25]. Linear transformer introduces a reparameterization of the dot-product self-attention mechanism to achieve linear computational complexity [26]. Graphormer introduces sparsity into the graph attention mechanism, thereby reducing the number of parameters required [27].

Improvements are also explored through other mechanisms, including frequency learning, signal decomposition, contextual embedding, and normalization strategies. Frequency learning methods transform the original time series into the frequency domain to model fluctuation. Representative works include TimesNet [28], FITS [29], and FreDF [30]. However, these methods remain limited to specific signal patterns. Decomposition-based methods separate the raw time series into sub-series using statistical approaches such as moving average (MA). To achieve better signal separation, advanced algorithms have been studied and integrated, for example, seasonal-trend decomposition using Loess (STL), empirical mode decomposition (EMD), and discrete Wavelet transform (DWT). Identity-based methods, such as MGC-RNN [31], DeepSTD [32], and CATS [33], address external covariates such as weekdays, holidays, and weather conditions. Normalization strategies, including instance normalization [34], batch normalization [35], and reversible instance normalization (ReVIN) [36], have also been investigated to stabilize training and improve generalization.

1.2. Objectives and contributions

Though existing studies have explored various methodologies, several research gaps remain.

For the large-scale forecasting problem in practical situations, training efficiency is critical owing to the need for frequent retraining and fine-tuning to accommodate evolving traffic patterns. Extensive training time can therefore hinder timely model updates.

Also, while the regional-based model has benefits in engineering practicality and convenience for large-scale networks, it remains highly challenging to effectively capture localized patterns without entanglement across variates. Especially when regional traffic has a high level of variate heterogeneity, it may cause unalignment to heterogenetic data. To address the issue, a new model design is required.

Besides, a forecasting model favorable to the operator is vital. Whether the process is more understandable or aligns with the operators' knowledge regarding traffic pattern is an essential consideration for their acceptance.

To bridge the gap between existing studies and practical deployment needs, this paper presents the following key contributions:

• A multi-tasking forecasting architecture integrating variational mode decomposition (VMD), where the raw signal is decomposed to distinguishable sub-signals for independent modeling. The architecture facilitates diverse model designs, improves learning capability, and improves operator's usability. To improve VMD's output stability and convergence speed, a segmenting strategy prior to training is introduced, allowing the process to operate on fixed overlapped time intervals.

• A temporal-frequency pure convolutional network as the backbone that jointly captures patterns of trend and fluctuations. dwTCN enhances the typical temporal convolutional network through deepwise operation and a patching strategy to capture the trend pattern. dwFCN employs DFT and 1D frequency convolution to capture the fluctuated patterns. To mitigate cross-variate mixing, deepwise operation is employed in both dwTCN and dwFCN. The backbone improves the learning capability toward highly heterogeneous data in real scenarios.

• The combination of low-complexity but effective algorithms, by incorporating s-VMD to decrease decomposing repetition and a pure convolutional network to establish the learning backbone and improve forecasting on batchwise FFN. The overall model presents efficient training and facilitates practical deployment toward large-scale networks.

The remainder of this paper is organized as follows: Section 2 defines the proposed model, Section 3 provides a case study and results, Section 4 provides ablation studies, Section 5 offers further discussion, and Section 6 concludes the study.

2. Model formulation

2.1. Problem definition

The number of variates, denoted as M, corresponds to the total number of highway traffic sensors (ETC gantries). Each variate records a univariate time series with C features over the historical length of L, resulting in a data size of $[L, C]$ for each variate and $[L, M, C]$ for the multivariate dataset.

To construct sample pairs for training, validation, and testing purposes, a sliding window approach is applied, where each sample pair consists of an input sequence and output sequence. In the multivariate case, the input sequence has a size of $[M, seq\_len]$, and the output sequence has a size of $[M, pred\_len]$, where *M* denotes the number of variates, *seq_len* the number of time steps used as the model's input, and *pred_len* the number of future time steps as the model's output. Typically, these sequences are adjacent or partially overlapping along the temporal axis.

For the forecasting problem, the objective is to learn a latent mapping function $f(\cdot)$ that maps the input sequences to its corresponding output sequences, defined as follows:

$$f(\{x_1, x_2 \dots x_T\}) = \{\hat{y}_1, \hat{y}_2 \dots \hat{y}_T\} \tag{1}$$

where:

$\{x_1, x_2 \dots x_T\}$ denotes the input sequences.

$\{\hat{y}_1, \hat{y}_2 \dots \hat{y}_T\}$ denotes the forecasted output sequences.

2.2. Prerequisite

Variational mode decomposition (VMD) is an optimization-based decomposition method that decomposes a real valued series input into a discrete number of intrinsic mode functions (IMFs), with each IMF to be mostly compact around a center frequency. Center frequencies are to be determined along with the decomposition [37]. As a solid decomposing algorithm, VMD has drawn increasing attention by signal-related studies such as wind speed forecasting [38], power forecasting [39], and fault diagnosis [40]. The methodological improvement includes coherent mode extraction of multiple signals [41], parameter optimization [42], very short time variants [43], and 2D extensions [44].

Given a univariate time series $U(t)$, let $\{u_k(t)\}_{k=1}^{K}$ and $\{w_k\}_{k=1}^{K}$ denote the sets of IMFs and the corresponding center frequencies, respectively. VMD aims to decompose $U(t)$ into K IMFs, during which it determines $\{u_k(t)\}_{k=1}^{K}$ and $\{w_k\}_{k=1}^{K}$ by optimizing the following function:

$$min_{u_k,w_k} \sum_{k=1}^{K} \left\| \partial_t \left[\left(\delta(t) + j\frac{1}{\pi t} \right) * u_k(t) \right] e^{-jw_k} \right\|_2^2, \tag{2}$$

subject to:

$$\sum_{k=1}^{K} u_k\,(t) = U(t), \tag{3}$$

where:

$u_k(t)$ denotes k-th IMF (time series), with $k \in \{1, 2 \ldots, K\}$.

w_k denotes the center frequency corresponding to k-th IMF.

∂_t denotes the partial derivative operation with respect to time.

$\delta(t)$ denotes the Dirac delta function.

j denotes the imaginary unit.

π denotes the mathematical constant (≈ 3.1416).

The solution is obtained by iteratively updating the center frequencies and bandwidths of IMFs using alternating direction method of multipliers [45]. The iteration proceeds until the relative error between consecutive updates falls below a predefined threshold, given by:

$$e = \frac{\sum_k \left\| \hat{u}_k^{(n+1)} - \hat{u}_k^{(n)} \right\|_2^2}{\sum_k \left\| \hat{u}_k^{(n)} \right\|_2^2} < \epsilon, \tag{4}$$

where:

n denotes the iteration index.

ϵ denotes the predefined threshold.

Temporal convolutional neural network (TCN) applies 1D convolution with dilated kernels to capture temporal dependencies by convolution kernel sliding along the temporal axis. Let the input sequence be x, the output sequence y, and 1D convolution is given by:

$$y(t) = \sum_{i=0}^{ks-1} f_{ks}\,(i) \cdot x(t - d \cdot i), \tag{5}$$

where:

d denotes the dilation step.

ks denotes the kernel size.

$f_{ks}(\cdot)$ denotes the kernel filtering function.

A typical TCN block also incorporates a dropout layer, a residual connection layer, and the nonlinear activation; the update of hidden representation is therefore given by:

$$h^r = \text{Dropout}\left(\sigma\left(h^{(r-1)} * \Theta^{(r)} + b^{(r)}\right) + h^{(r-1)}\right), \tag{6}$$

where:

$h^{(r)}$ denotes the hidden representation of the r-th layer.

$\Theta^{(r)}$ denotes the convolution kernel of the r-th layer.

$*$ denotes the 1D convolution operation.

$b^{(r)}$ denotes the bias vector.

$\sigma(\cdot)$ denotes a nonlinear activation function.

The discrete Fourier transform (DFT) converts a discrete signal from the temporal domain to the frequency domain. Denoting the input sequence as $x \in R^{M \times seq\_len}$ and its value at time step t as $x[t] \in R^{M}$, the DFT operation is given by:

$$\mathcal{H}[k] = \sum_{t=0}^{seq_{len}-1} x[t] e^{-j\frac{2\pi kt}{seq\_len}}, \tag{7}$$

where:

$\mathcal{H}[k]$ denotes the frequency domain representation of x at the k-th frequency with $k \in \{0, 1, \ldots, seq\_len - 1\}$.

seq_len denotes the length of input sequence.

j denotes the imaginary unit.

π denotes the mathematical constant (≈ 3.1416).

Applying Euler's formula to expand the exponential term in Eq (7):

$$e^{-j\frac{2\pi kt}{seq\_len}} = cos\left(\frac{2\pi kt}{seq\_len}\right) - j\, sin\left(\frac{2\pi kt}{seq\_len}\right). \tag{8}$$

Therefore, Eq (7) can be rewritten as:

$$\mathcal{H}[k] = \sum_{t=0}^{seq\_len-1} x[t]\left(cos\left(\frac{2\pi kt}{seq\_len}\right) - j\, sin\left(\frac{2\pi kt}{seq\_len}\right)\right). \tag{8}$$

Separating the real and imaginary parts, $\mathcal{H}[k]$ can be rewritten in complex valued formula:

$$\mathcal{H}[k] = \sum_{t=0}^{seq\_len-1} x[t]\, cos\left(\frac{2\pi kt}{seq\_len}\right) + \left(-\sum_{t=0}^{seq\_len-1} x[t]\, sin\left(\frac{2\pi kt}{seq\_len}\right)\right) j = a_k + b_k j. \tag{10}$$

a_k and b_k exhibit the amplitude and phase of $\mathcal{H}[k]$ respectively given as:

$$|\mathcal{H}[k]| = \sqrt{a_k^2 + b_k^2}, \tag{11}$$

$$\theta_k = arg(\mathcal{H}[k]) = tan^{-1}\left(\frac{b_k}{a_k}\right). \tag{12}$$

Here, the amplitude indicates the strength of signal, while the phase denotes the relative shift of the signal with respect to the reference.

Feedforward neural network (FFN) applies linear transformation with a nonlinear activation. In multivariate cases, due to its simplicity, typical FFN serves as a mapping function from the input sequences to the output sequences with parameters shared among the variates. The update of hidden representation through FFN is given by:

$$h^{(r)} = \sigma\left(W^{(r)} h^{(r-1)} + \varepsilon^{(r)}\right), \tag{13}$$

where:

$h^{(r)}$ denotes the hidden representation of the r-th layer.

$W^{(r)}$ denotes the transformation matrix of the r-th layer.

$\varepsilon^{(r)}$ denotes the bias vector of the r-th layer.

$\sigma(\cdot)$ denotes a nonlinear activation function.

2.3. Model improvement

In existing studies, decomposition is typically coupled with learning process and applied directly to the input sequences; however, it presents several critical challenges, as follows:

Inadequate resolution: Many decompositions require sufficient data to reliably identify signal patterns, and applying decomposition to very short sequences may lead to poor output.

Window edge effect: The lack of contextual values at the boundary of each decomposed window leads to inevitable distortion. Applying decomposition directly on the input sequence will lead to a substantial increase of window edges, thereby exacerbating boundary effects.

Computational inefficiency: Applying decomposition on the input sequence is inefficient as the operation is repeated along both batch and variate axis. For an input sequence of size $[B, M, seq\_len]$, the decomposing process is to be operated $B * M$ times, imposing significant computational burden and strictly limiting the selection of decomposing methods.

To address these issues, a learning-decoupled segmented VMD (s-VMD) strategy is proposed. Specifically, the raw dataset is partitioned into segments with fixed-time interval, allowing the algorithm to operate on localized segments of adequate length. To maintain temporal continuity and minimize window edge effects, overlapping between segments is introduced. Let the historical length of the multivariate dataset be L, segment length be l, and overlap be p; the total number of segments is given by:

$$T = (L - p)//l. \tag{14}$$

For the t-th segment, $\chi_{(t):(t+l)} \in R^{M \times l}$. VMD is operated variate-wise (solving optimization problem independently), yielding K IMFs. The segmented VMD operation of t-th segment is given as:

$$\text{s-VMD}(\chi_{(t):(t+l)}) = \left\{\chi_{(t):(t+l)}^{(1)}, \chi_{(t):(t+l)}^{(2)}, \cdots, \chi_{(t):(t+l)}^{(K)}\right\}, \tag{15}$$

where:

$\chi^{(k)}{}_{(t):(t+l)} \in R^{M \times l}$ denotes the k-th IMF for t-th segment, with $k \in \{1,2,\dots,K\}$.

After, the generated IMFs of segments with the same index k are concatenated along the temporal axis to reconstruct a new sub-dataset. Value continuity between segments is preserved by averaging the values in the overlapping regions:

$$\chi_{(t+l-p):(t+l)}^{(k)} = \frac{1}{2}\left[\chi_{(t+l-p):(t+l)}^{(k)} + \chi_{(t+l-p):(t+2l-p)}^{(k)}\right]. \tag{16}$$

After applying s-VMD, the reconstructed dataset consists of K sub-datasets, each of size $[M, L]$. The algorithmic complexity is estimated nearly $O(M \cdot K \cdot L \cdot \log(L//T))$ per iteration, which is linear to M and L. The overall process of s-VMD is illustrated in Figure 1.

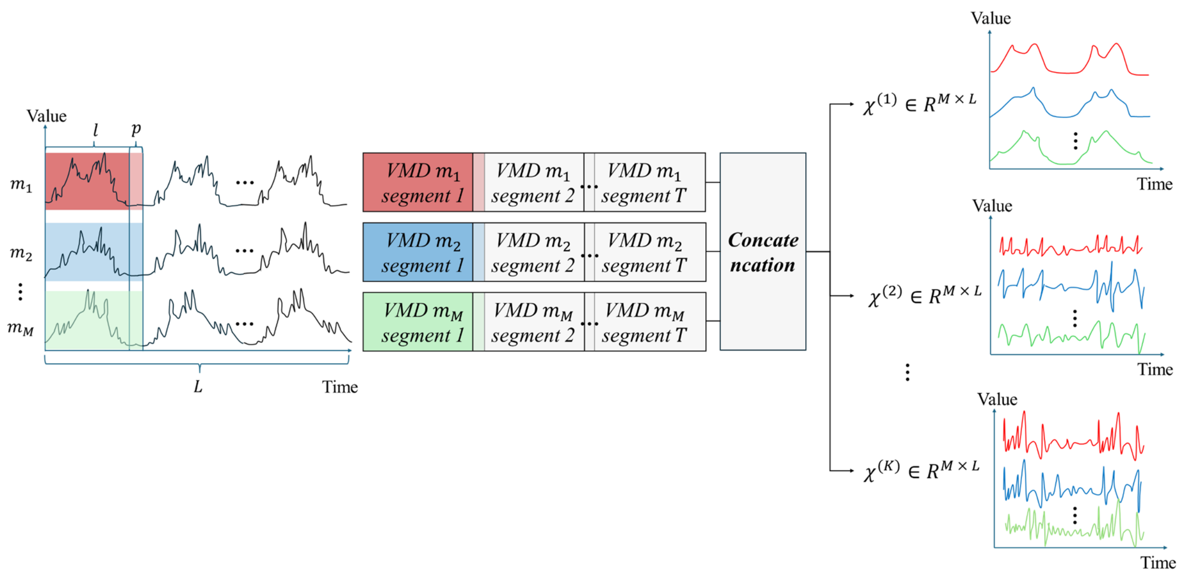


Figure 1. Illustration of s-VMD.

Typical 1D convolution presents two major limitations when applied to multivariate time series. First, it introduces undesirable mixing across variates. In the standard operation, M kernels of size $[M, ks]$ are initialized, with each kernel responsible for capturing one shared feature channel among all variates (see Figure 2). The output is the summation of all feature channels, causing hidden representation being updated by shared parameters. Second, stacking dilated convolution layers increases the risk of over smoothing, potentially diminishing the model's ability to capture variate specific patterns.

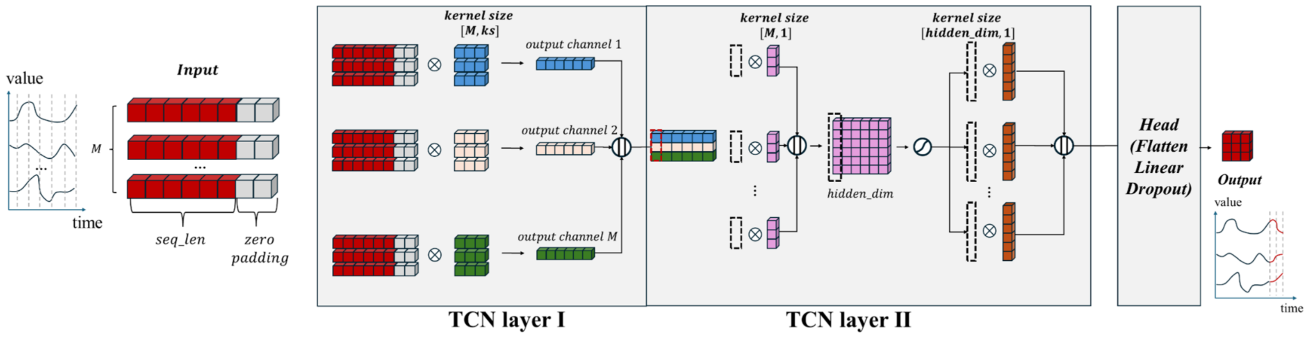


Figure 2. Illustration of standard TCN.

To address these limitations, a patching strategy inspired by computer vision studies is introduced. Patching transforms input sequence of size $[B, M, seq\_len]$ to $[B, M, P, N]$ by expanding the temporal-related axis from one (seq_len) to two (P and N), where P denotes patch size (the number of time steps as a patch), N denotes the number of patches decided by $N = (seq\_len - P)//S$, and S denotes repeated time steps between patches. After patching, the intra-patch axis preserves short-term dependencies, while inter-patch axis preserves longer term dependencies. The restructuring process expands the receptive field of 1D convolution without deep layer stacking.

Besides, a deepwise operation is introduced to mitigate pattern mixing. Specifically for input of size $[B, M, seq\_len]$, M variates are partitioned into G groups, with each group assigned to localized kernel filters of size $[M//G, ks]$. During the operation, each group operates only with the assigned kernels, thereby replacing global parameter sharing with intra-group sharing. To further achieve full variate independence, the number of groups and output channels is set equal to the number of variates, enabling the variate to interact with one dedicated kernel. This design enhances the model's capacity over a typical TCN to capture variate-specific patterns. The overall process of dwTCN is illustrated in Figure 3.

An additional benefit of dwTCN is its lower computational complexity. Typical TCN exhibits up to $O(M^2)$ complexity regarding to M, while dwTCN reduces it to $O(M)$ by reducing the size of the kernel and the number of operations. This reduction is particularly advantageous in practical deployment since most highway networks usually have large M.

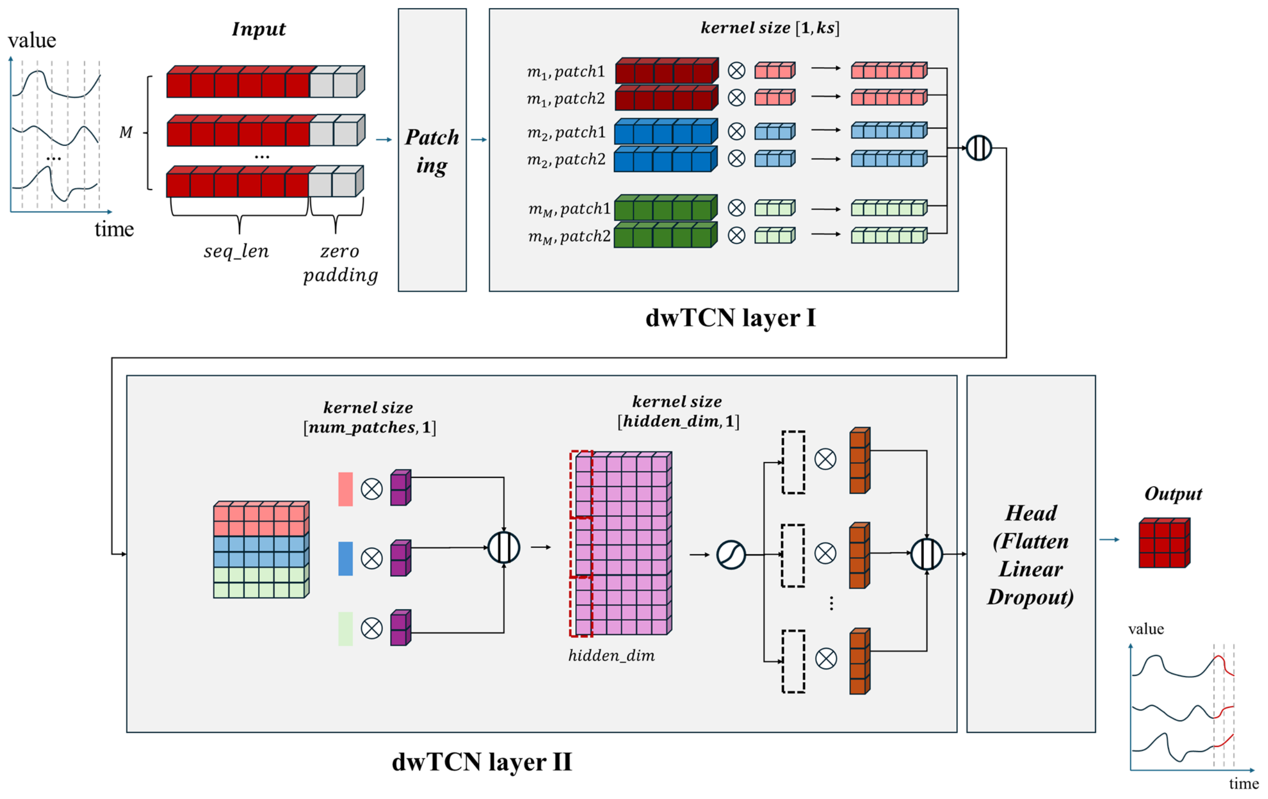


Figure 3. Illustration of dwTCN.

Highway traffic flow is typically recorded at very short time intervals for real-time tolling purposes. Owing to the dynamic and uncertain properties of short-term traffic flow [46] with unpredictable external impacts such as accidents, bad weather, and road closure, signals of traffic flow often exhibit oscillatory patterns. Importantly, these oscillations are not merely noises but reflect meaningful temporal dynamics inherent to traffic behaviors. Following VMD, one IMF with near-zero frequency (DC) preserves the trend-dominated pattern, while the remaining IMFs with non-zero frequency (ACs) preserve the oscillatory-dominated patterns. The dwFCN specifically targets ACs and, for both effectiveness and efficiency purposes, learns these high fluctuations in the frequency domain rather than in the temporal domain.

Given the input sequence from a single AC as $x \in R^{B\times M\times seq\_len}$ with batch size B, number of variates M and sequence length seq_len, each dwFCN block comprises the following layers:

DFT layer: A real-valued fast Fourier transform (FFT) algorithm to generate the frequency domain representation of the input:

$$\mathcal{H} = FFT(x) = a + bj,\ a, b \in R^{B\times M\times freq\_len}, \tag{17}$$

where:

a and b denote the real and imaginary parts, respectively.

$freq\_len$ denotes length of frequency. Owing to real valued FFT's symmetry, $freq\_len = seq\_len\ //2$.

Concat layer: Indicated by Eqs (11) and (12), the amplitude and phase of $\mathcal{H}$ are jointly represented by its real part and imaginary part. A concatenation is applied to construct a new tensor:

$$ri_{(a,b)} = Concat(a, b) \in R^{B\times(2M)\times freq\_len}. \tag{18}$$

FCN layer I: A deepwise 1D convolution with padding along the frequency axis of $ri_{(a,b)}$ is applied. The number of groups $G_{dw} = M$ so that $2M$ kernels are assigned to M groups and to capture real and imaginary patterns for each variate:

$$h_{(a,b)} = dwconv\left(ri_{(a,b)}\right) \in R^{B\times(2M)\times freq\_len}. \tag{19}$$

FCN layer II: Two pointwise (kernel size = 1) convolutions along the frequency axis of $h_{(a,b)}$ are applied accordingly. Number of groups $G_{pw} = M$ is set to fuse the real and imaginary pattern within each variate. Given $hidden\_dim$ as the hidden dimension number, a bottleneck to improve training is also applied by increasing and decreasing the number of channels, where $pwconv1(\cdot): 2M \to hidden\_dim$, $pwconv2(\cdot): hidden\_dim \to 2M$. Nonlinear activation σ(GeLU) is also introduced in between. The output keeps $2M$ channels:

$$h_{(fusion)} = pwconv2\left(\sigma\left(pwconv1\left(h_{(a,b)}\right)\right)\right) \in R^{B\times(2M)\times freq\_len}. \tag{20}$$

iDFT layer: The updated real and imaginary parts are obtained by splitting $h_{(fusion)}$ along the variate axis and are utilized to reconstruct its complex valued form $\mathcal{H}_{(fusion)}$. The inverse FFT is applied to generate its temporal domain representation $x_{(fusion)}$, followed by a residual connection to generate block's output, denoted as $dwFCN(x)$.

$$\left[a_f, b_f\right] = Split\left(h_{(fusion)}\right),\ a_f, b_f \in R^{B\times M\times freq\_len} \tag{21}$$

$$\mathcal{H}_{(fusion)} = (a_f + b_f j) \in C^{B\times M\times req\_len} \tag{22}$$

$$x_{(fusion)} = iFFT(\mathcal{H}_{(fusion)}) \in R^{B\times M\times seq\_len} \tag{23}$$

$$dwFCN(x) = x + x_{(fusion)} \tag{24}$$

The overall process of dwFCN is illustrated in Figure 4.

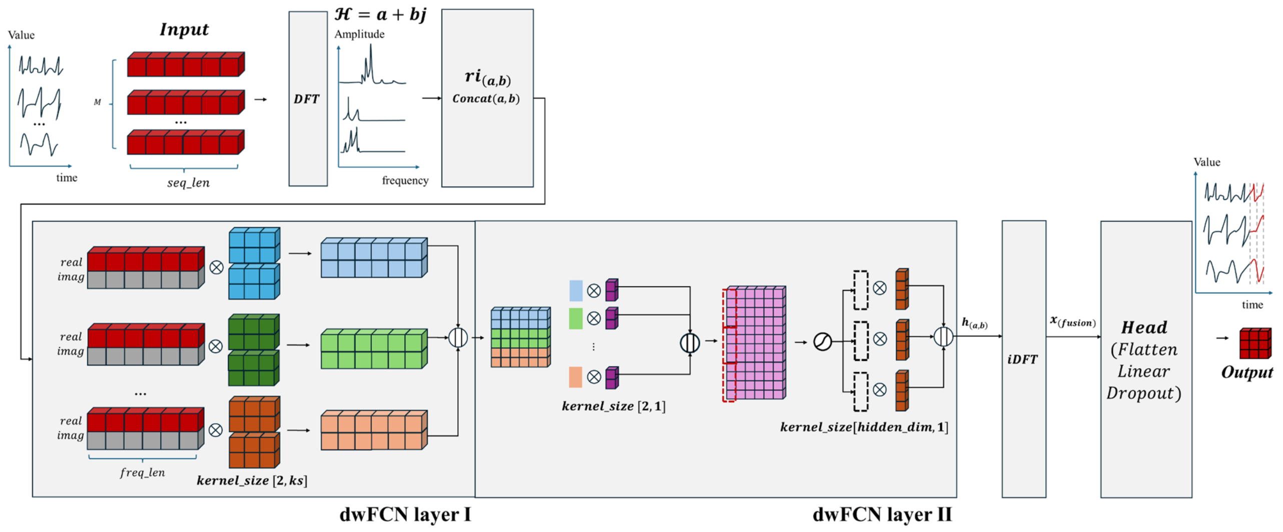


Figure 4. Illustration of dwFCN.

Typical FFN employs globally shared parameters among batch and variate axis. Previous attempts introduce variate-wise projection matrices, expanding the total number of parameters from $[seq\_len, pred\_len]$ to $[M, seq\_len, pred\_len]$. However, this design imposes an unacceptable computational burden when M is huge. Alternatively, we improve by initializing batchwise projection matrices. Specifically, for each batch of the input sequence, an independent projection matrix of size $[seq\_len, pred\_len]$ is initialized. The update of hidden representation through batchwise FFN is given by:

$$h_b^{(r)} = \sigma\left(W_b^{(r)} h_b^{(r-1)} + \varepsilon_b^{(r)}\right)(\tag{25}$$

where:

${h_b}^{(r)}$ denotes the hidden representation of the b-th batch of input, with $b \in \{0, 1, \dots, B-1\}$.

B denotes the batch size.

σ denotes a nonlinear activation function.

${W_b}^{(r)}$, ${\varepsilon_b}^{(r)}$ are learnable parameters.

This design allows input sequences at different "positions" on the temporal axis to maintain separate parameterizations. By batchwise FFN, the number of parameters has increased from $seq\_len \times pred\_len$ to $B \times seq\_len \times pred\_len$, since B is much smaller than M, and it brings better scalability.

The overall architecture is illustrated in Figure 5. Blocks and algorithms include s-VMD, batching, dwTCN block, dwTCN block, and batchwise FFN head. To facilitate the training process, ReVIN, batch normalization, dropout, and residual connection operations are incorporated.

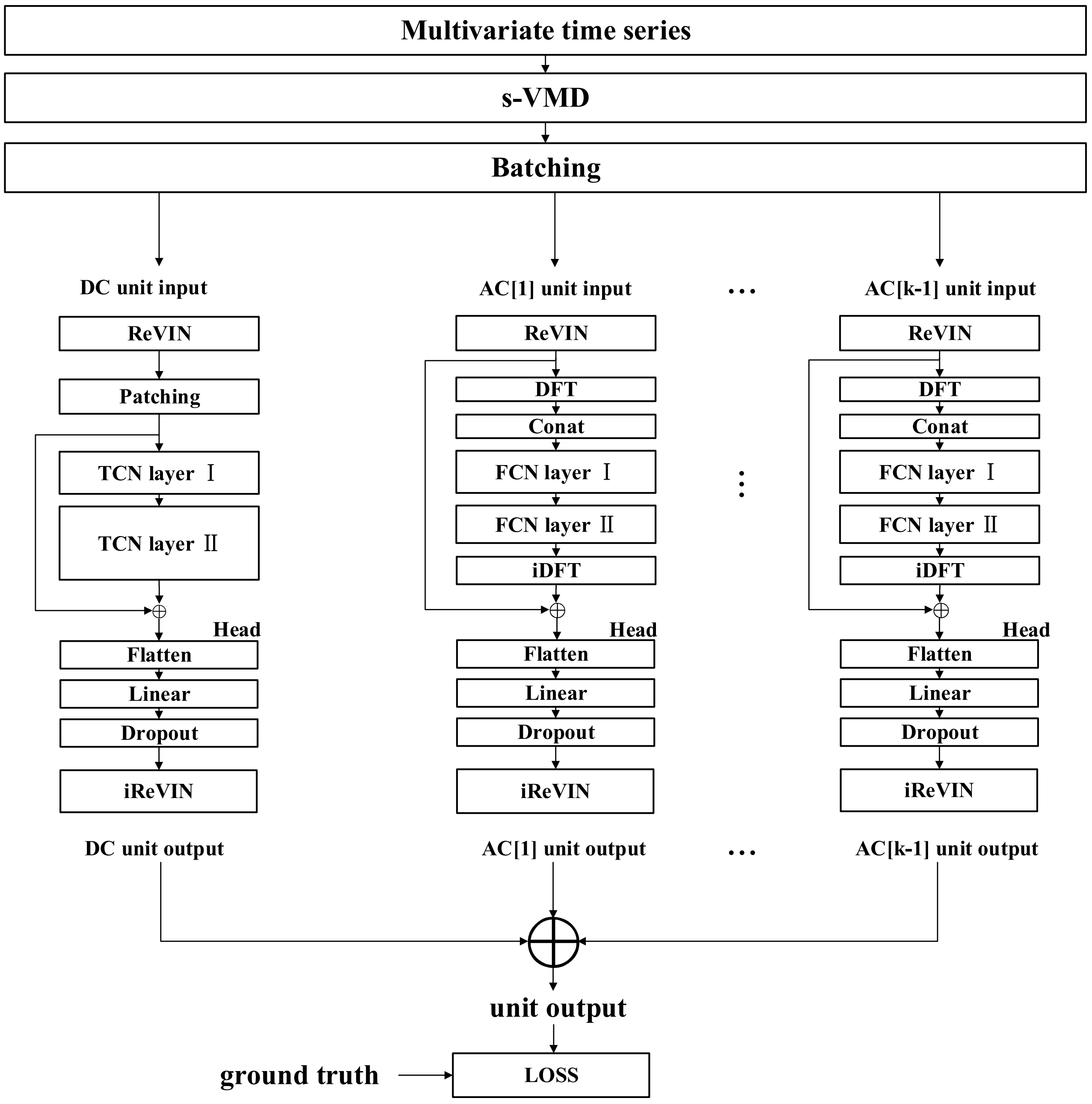


Figure 5. Illustration of overall architecture.

2.4. Training algorithm

Detailed steps are given as follows:

```
STAGE-1: s-VMD
Given multivariate dataset $\chi$, the number of IMFs $K$, VMD converge tolerance $\epsilon$, segment length $l$, overlap between segments $p$
Step 1. Determines number of segments $T$ and segmenting dataset $\chi$.
Step 2. For the $t$-th segment $\chi_{(t):(t+l)}$, apply VMD to determine its set of IMFs, repeat Step 2 until all segments have completed.
Step 3. For the $k$-th IMF, do:
    Step 3.1. Concatenate all the k-th IMF along the $l$ axis, constructing a sub-dataset $\dot{\chi}^{(k)}$.
    Step 3.2. Average the values in overlapping regions of $\dot{\chi}^{(k)}$, obtaining $\chi^{(k)}$.
End of STAGE-1. K sub-datasets with each of size $[M, L]$ are generated.
For simplicity, the sub-dataset at near-zero frequency is named after '$DC$', the remaining sub-datasets are named after '$AC[1], AC[2], \dots, AC[K-1]$' respectively.

STAGE-2: Batching
Given batch size $B$, length of input $seq\_len$.
Step 1. Use Python's Dataset package to split the sub-datasets from Stage 1 into training/validating/testing set.
Step 2. Use Python's Dataloader package to compress the consecutive $B$ samples into a batch.
End of STAGE-2. The unit input tensor $x \in R^{B\times M\times seq\_len}$ for model training purposes is created.

STAGE-3: dwTCN for DC
Given $x_{(DC)} \in R^{B\times M\times seq\_len}$ the DC's unit input tensor:
Step 1. ReVIN to normalize.
Step 2. Given $P$(patch size), $D$ (expanded patch size), $S$ (patch stride), obtain number of patches $N$ and apply patching by expanding a dimension and to $x_{(DC)}$ and applying a linear transform, with tensor shape changed $[B, M, seq\_len]$ -> $[B, M, D, N]$.
Step 3. TCN layer I
Given $ks$ the kernel size:
    Step 3.1. Reshape the patched tensor, $[B, M, D, N]$ -> $[B, M*D, N]$.
    Step 3.2. Do deepwise 1D convolution along $N$ axis with groups, followed by a BatchNorm.
Step 4. TCN layer II
Given $hiddn\_dim$ (bottleneck channels):
    Step 4.1. Do 1st pointwise ($1\times 1$) convolution along $N$ axis with groups. $[B, M*D, N]$ -> $[B, M*hiddn\_dim, N]$, followed by a nonlinear activation (GeLU) and dropout.
    Step 4.2. Do 2nd pointwise ($1\times 1$) convolution along N axis with groups. $[B, M*hiddn\_dim, N]$ -> $[B, M*D, N]$.
    Step 4.3. Reshape the convoluted tensor $[B, M*D, N]$ -> $[B, M, D, N]$, followed by a dropout and residual connection.
Step 5. Head
Given $pred\_len$ (length of output sequence):
    **Step 5.1.** Flattening the last dimension, $[B, M, D, N]$ -> $[B, M, D * N]$.
    **Step 5.2.** Do batchwise linear transform to map $D * N$ to $pred\_len$, $[B, M, D * N]$ -> $[B, M, pred\_len]$, followed by a dropout.
    **Step 5.3.** Do inverse ReVIN to de-normalize and generate output.
**End of STAGE-3.** DC's unit output $\hat{y}_{DC} \in R^{B \times M \times pred\_len}$ is generated.

**STAGE-4: dwFCN for each AC[·]**
Given $x_{AC[k]} \in R^{B \times M \times seq\_len}$ the AC$[k]$'s unit input tensor, $k \in \{1, 2, \dots, K-1\}$:
**Step 1.** ReVIN to normalize.
**Step 2. DFT layer:** Apply FFT to generate frequency domain representation of input $\mathcal{H} = a + bj$.
**Step 3. Concat layer:** $conat(a, b)$ along $M$ axis to generate $ri_{(a,b)}$.
**Step 4. FCN layer I**
Given $ks$ the conv1d kernel size, do deepwise 1D convolution to $ri_{(a,b)}$ along $freq\_len$ axis with groups.
**Step 5. FCN layer II**
Given $hidden\_dim$ (bottleneck channels):
    **Step 5.1.** 1st pointwise ($1 \times 1$) convolution along $freq\_len$ axis with groups, $[B, 2M, freq\_len]$ -> $[B, hiddn\_dim, freq\_len]$, followed by a nonlinear activation (GeLU) and dropout.
    **Step 5.2.** 2nd pointwise ($1 \times 1$) convolution along $freq\_len$ axis with groups, $[B, hiddn_{dim}, freq_{len}]$ -> $[B, 2M, freq\_len]$ followed by a dropout.
    **Step 5.3.** Apply a dropout, generating the convoluted tensor $h_{(fusion)}$.
**Step 6. iDFT**
    **Step 6.1.** Split $h_{(fusion)}$ along $2M$ axis for updated real and imaginary parts to reconstruct complex valued tensor $\mathcal{H}_{(fusion)}$.
    **Step 6.2.** Do inverse FFT to $\mathcal{H}_{(fusion)}$ to obtain its temporal domain representation $x_{(fusion)} \in R^{B \times M \times seq\_len}$.
    **Step 6.3.** Do residual connection.
**Step 7. Head**
**Given** $hidden\_len$, and $pred\_len$:
    **Step 7.1.** Do batchwise linear transform to flatten $seq\_len$ to $hidden\_len$, $[B, M, seq\_len]$ -> $[B, M, hidden\_len]$.
    **Step 7.2.** Do batchwise linear transform to switch $hidden\_len$ to $pred\_len$, $[B, M, hidden\_len]$ -> $[B, M, pred\_len]$.
    **Step 7.3.** Do dropout and inverse ReVIN to de-normalize and generate output.
**End of STAGE-4.** For each AC$[k]$, its unit output $\hat{y}_{AC[k]} \in R^{B \times M \times pred\_len}$ is generated.

**STAGE-5: Loss calculation and model training**
**Step 1.** Summing up unit outputs to generate forecast output $\hat{y} = \hat{y}_{(DC)} + \sum_{k=1}^{K-1} \hat{y}_{(AC[k])}$.
**Step 2.** Calculate MSE losses between $\hat{y}$ and ground truth $y$.
**Step 3.** Use Adam optimizer to update model's parameters to minimize loss with the training set.
**End of STAGE-5.** Training for the unit input finished, repeat **STAGE-3 to STAGE-5** for the next unit input in the training set.
**Repeat training for the next epoch until the stopping criteria is reached.**
```

3. Case study

3.1. Dataset description

The dataset is obtained via highway ETC gantries located in southern Jiangsu, China, from the Jiangsu provincial highway tolling system. The regional network covers national-level and provincial-level highways with a total of 2350 km. There are 1203 ETC gantries available for traffic flow monitoring. Traffic flow is recorded over 8 prefecture-level cities: Nanjing, Zhenjiang, Changzhou, Wuxi, Suzhou, Yangzhou, Taizhou, Nantong. The study region is located at the round-Shanghai metropolitan zone, covering 15 counties and 37 districts. Among the 1203 gantries, 737 are bi-directional mainline gantries, and there are 466 toll station gantries on inbound and outbound directions. Certain routes, including G4221 and G40, lack flow data due to ongoing expansion projects. There are also 70 intersections within the region. The geographical overview of the region is illustrated in Figure 6.

The selected flow dataset spans from April 1, 2023, to June 30, 2023, with records every five minutes. The overall traffic flow is aggregated by summing flows of 17 vehicle categories. The dataset also includes gantry name, gantry ID, timestamp, and detailed vehicle flow counts per category (K1–K4 for passenger vehicles, H1–H6 for cargo vehicles, Z1–Z6 for special vehicle types, and an "Other" category for unrecognized vehicles). A detailed data overview is summarized in Table 1.

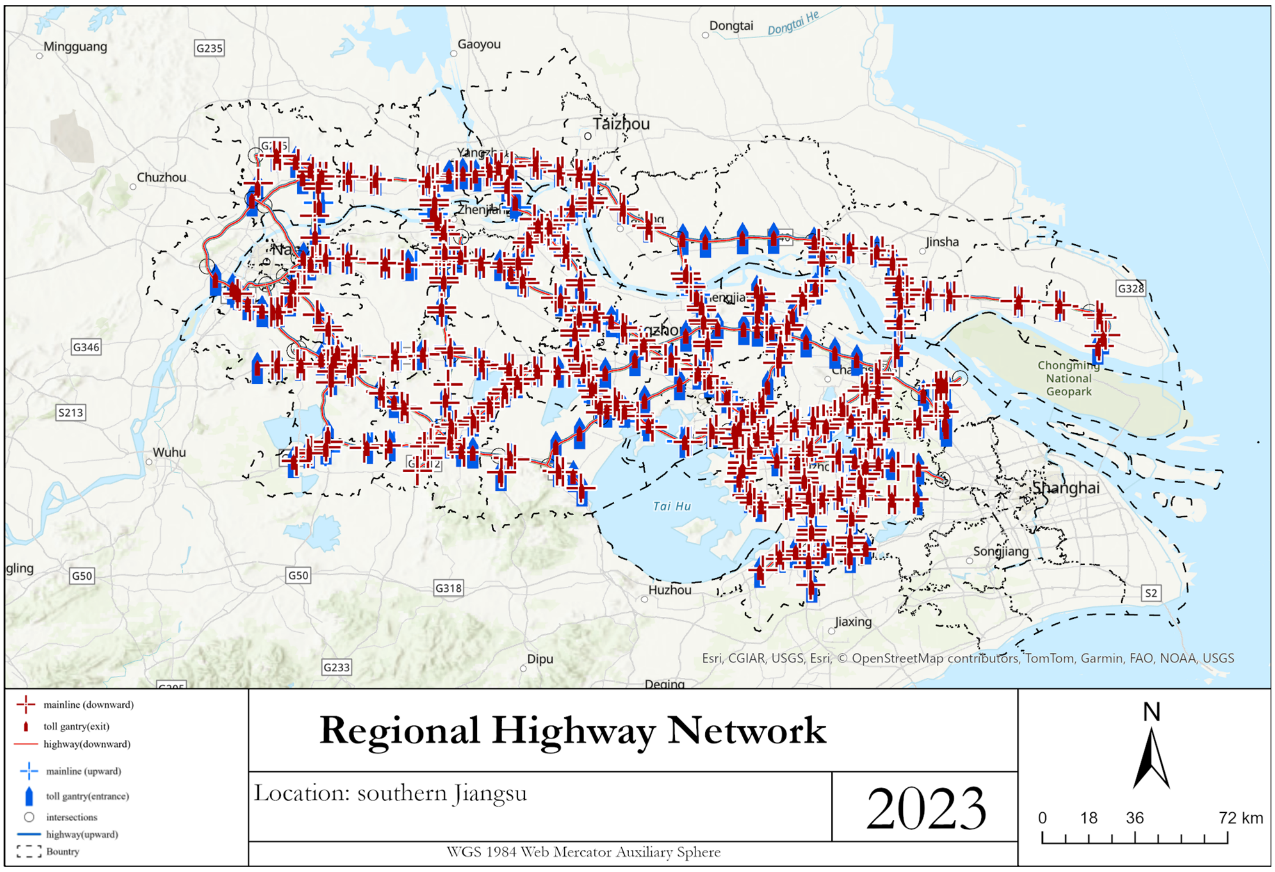


Figure 6. Region of study.

Table 1. Data description.

Column	Description	Sample	Datatype
Gantry name	Name of gantry	Zhengyi Intersection to Sutong bridge	String
Gantry id	ID of gantry	G000232002000110010;	String
Datetime	Flow recording time	2023-04-01 00:00:00	Datetime
K1	Flow of passenger vehicle type 1	9.0	Float
K2	Flow of passenger vehicle type 2	3.0	Float
K3	Flow of passenger vehicle type 3	2.0	Float
K4	Flow of passenger vehicle type 4	1.0	Float
H1	Flow of cargo vehicle type 1	6.0	Float
H2	Flow of cargo vehicle type 2	4.0	Float
H3	Flow of cargo vehicle type 3	3.0	Float
H4	Flow of cargo vehicle type 4	1.0	Float
H5	Flow of cargo vehicle type 5	1.0	Float
H6	Flow of cargo vehicle type 6	0.0	Float
Z1	Flow of special type 1 vehicle	1.0	Float
Z2	Flow of special type 2 vehicle	0.0	Float
Z3	Flow of special type 3 vehicle	0.0	Float
Z4	Flow of special type 4 vehicle	0.0	Float
Z5	Flow of special type 5 vehicle	0.0	Float
Z6	Flow of special type 6 vehicle	0.0	Float
Other	Flow of unrecognized vehicle	0.0	Float

3.2. Baseline models for comparison

Several competitive forecasting baselines that were applied in recent traffic studies are selected. They cover a variety of methodological categories, computational complexity, and key learning components as summarized in Table 2. All models are evaluated under forecast step of {3, 6, 9, 12, 24}.

Table 2. Baseline description.

Models	Key learning component	Estimated complexity regarding variate ($\boldsymbol{M}$) and temporal length ($\boldsymbol{L}$)
STAEformer	Temporal self-attention Spatial self-attention	$O(M \cdot L^2 + M^2 \cdot L)$
MTGNN	Temporal 1D convolution (multiple kernel) Graph diffusion (wavenet) and spatial self-attention	$O(M^2 + M^2 \cdot L)$
PatchTST	Patching temporal self-attention	$O(M \cdot L^2)$
PatchMixer	Patching temporal 1D convolution (deepwise)	$O(M \cdot L)$
Dlinear	Temporal multilayer perception moving average	$O(M \cdot L)$
TCN	Temporal 1D convolution	$O(M^2 \cdot L)$

STAEformer [47]: A spatial-temporal attention-based model utilizing self-adaptive graph.

MTGNN [15]: A spatial-temporal convolution-based model that incorporates multiple kernels, graph diffusion, and an optional self-adaptive graph.

PatchTST [17]: An attention-based model utilizing patching strategy and temporal domain self-attention mechanism.

PatchMixer [16]: A pure convolution-based model utilizing patching strategy and deepwise convolution operations.

DLinear [10]: An MLP-based model stacking multiple linear layers and employing moving average decomposition on input sequences.

TCN [12]: A temporal convolution-based model utilizing typical 1D convolution kernel with dilation.

3.3. Experimental setup

Experiments were conducted under a computer environment with Intel(R) Core (TM) i7-10700F CPU @ 2.90 GHz and NVIDIA GeForce RTX 4070 Ti SUPER.

The model hyperparameters are configured as follows: number of IMFs K is set to 4, the coverage tolerance ϵ is set to 1×10^{-6}, segment length l is set to 288 to cover a full day and leverage low traffic flow during midnight hours, and overlapping p is set to 2. To further accelerate s-VMD computation, parallel processing is enabled using Joblib tool in Python 3.9. In dwTCN, patch size P is set to 4, patch stride S is set to 2, kernel size ks is set to 7, number of blocks is set to 1, and hidden dimension is set to 64. In dwFCN, the frequency kernel size ks is set to 3, number of blocks is set to 1, and the hidden dimension is set to four times the input sequence length. Evaluation metrics include mean absolute error (MAE), mean squared error (MSE), and mean absolute percentage error (MAPE). Missing values are excluded from the evaluation process. The model is trained using the adaptive moment estimation (Adam) optimizer.

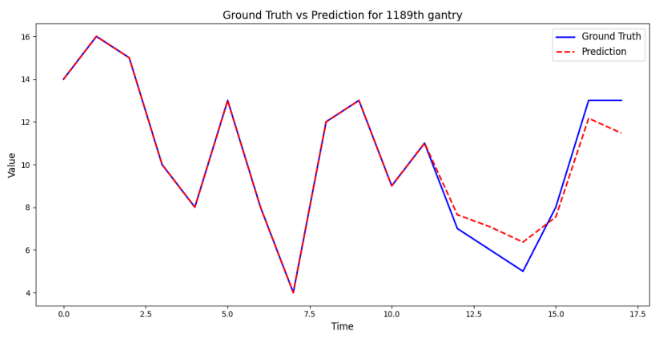
3.4. Results

The experimental results are summarized in Table 3. The proposed model achieves an overall MAE reduction of 8.67% to 26.7% and a MAPE reduction of 9.4%–29.9% at all steps. Figure 7 illustrates model's adaptivity at different forecast steps.

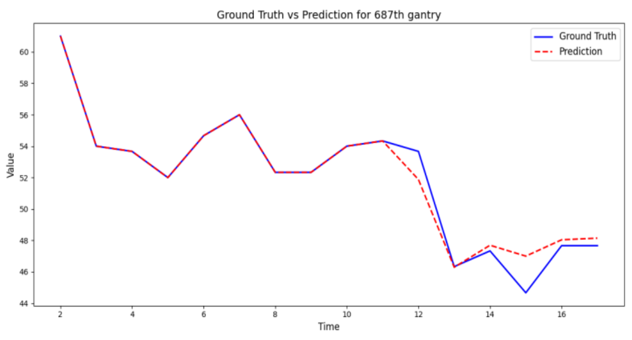
The observed improvement can be attributed to several factors. The segmenting strategy preserves sufficient resolution for VMD's effective separation to raw signals, thereby facilitating diverse model designs. The integration of both temporal and frequency domain learning backbone enhances the model's capability in handling diverse data patterns. Addressing variate independence mitigates over-smoothing.

To evaluate time efficiency, both training time per epoch and overall training time are compared under the same training epochs and early stopping patience. Early stopping allows earlier jump out if the validation loss ceases to decrease for a consecutive number of epochs, a widely used strategy adopted to output models' optimal parameters. As shown in Table 4, despite the proposed model that requires longer training per epoch (mainly attributed to multiple model initializations), it also benefits from the decomposition-guided multi-tasking learning that each model processes simpler signal, leading to its quicker convergence. The final factors contributing to this efficiency are both the decreased learning difficulties and the avoidance of deep-stacking learning layers. Figure 8 further illustrates the decreased difficulties starting with the least training loss of the first epoch. As a result, the proposed model achieves the second highest training efficiency among all baselines, surpassed only by DLinear. Regarding total training time, on average, among steps, it is 17.6%, 47.6%, 33.3%, 70.1%, and 11% faster than PatchMixer, PatchTST, TCN, STAEformer, and MTGNN, respectively.

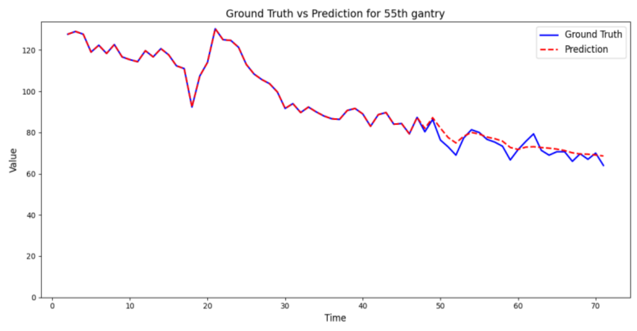
In addition, its growing rate of training time along with the prediction step remains moderate, exhibiting only a 47% increase in training time per epoch from 3-step to 24-step forecasting. This is attributed to all learning blocks and algorithms maintaining linear or log-linear complexity. On contrast, attention-based models such as PatchTST and STAEformer experience nonlinear increases in training time per epoch, at 303% and 607% from 3-step to 24-step forecasting, respectively.



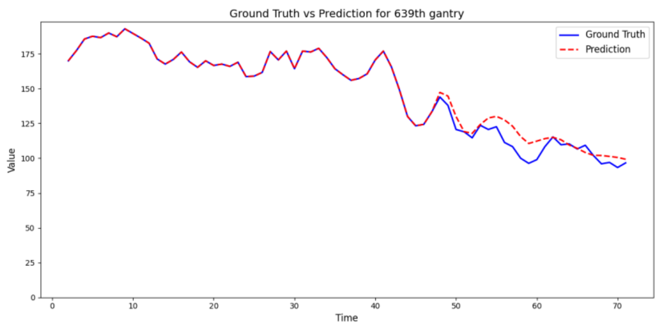
(a)



(b)



(a)



(b)

Figure 7. Forecasted result and ground truth. (a) Prediction and ground truth for toll gantry 1189 (forecast step-6). (b) Prediction and ground truth for toll gantry 687 (forecast step-6). (c) Prediction and ground truth for mainline gantry 1189 (forecast step-24). (d) Prediction and ground truth for mainline gantry 639 (forecast step-24).

Table 3. Performance results.

Models/Steps		3	6	9	12	24
This paper	MAE	6.97	7.14	7.65	8.47	10.64
	MSE	222.12	244.60	269.25	309.10	455.50
	MAPE	25.61	24.46	25.63	28.29	32.20
PatchMixer	MAE	7.95	8.52	9.08	9.73	12.32
	MSE	195.27	233.49	270.17	309.23	496.25
	MAPE	27.41	28.27	28.82	30.41	37.05
PatchTST	MAE	7.96	8.50	9.05	9.65	12.05
	MSE	195.47	232.95	267.67	306.23	480.59
	MAPE	27.29	28.26	28.95	30.21	36.62
Dlinear	MAE	8.01	8.64	9.29	9.99	12.88
	MSE	195.76	242.77	287.69	334.74	560.23
	MAPE	32.14	31.42	34.86	38.24	45.58
TCN	MAE	8.21	8.21	8.84	9.94	11.14
	MSE	230.51	230.39	284.85	341.92	476.17
	MAPE	35.64	35.67	36.40	40.90	38.61
MTGNN	MAE	7.75	8.13	8.58	9.01	/*
	MSE	202.77	231.34	261.14	277.22	/*
	MAPE	26.13	27.46	28.96	31.59	/*
STAEFormer	MAE	7.35	7.81	8.15	8.40	9.26
	MSE	175.72	211.32	235.44	251.64	302.56
	MAPE	30.88	32.84	33.10	34.07	36.46

* Note: not supported by the original code.

Table 4. Computational efficiency results.

Models/steps		3	6	9	12	24
This paper	per epoch (s)	51.45	56.36	49.46	64.25	76.02
	total (s)	565.95	507.24	445.14	1863.25	2432.64
PatchTST	per epoch (s)	28.84	55.27	34.61	61.94	87.45
	total (s)	346.08	2210.80	1592.06	3778.34	3148.20
PatchMixer	per epoch (s)	29.26	23.21	20.99	19.61	24.10
	total (s)	965.58	1044.45	1784.15	1137.38	2120.80
Dlinear	per epoch (s)	9.69	10.85	10.90	11.34	14.91
	total (s)	969.34	390.60	484.21	408.24	820.05
TCN	per epoch (s)	19.58	20.63	22.78	30.50	36.73
	total (s)	1958.49	2042.37	1230.12	1372.50	1983.42
MTGNN	per epoch (s)	80.18	81.60	81.84	82.73	/
	total (s)	3207.22	2366.47	4583.04	4798.34	/
STAEFormer	per epoch (s)	90.06	146.83	225.27	290.91	550.35
	total (s)	2431.52	3083.43	5406.56	7272.66	13208.46

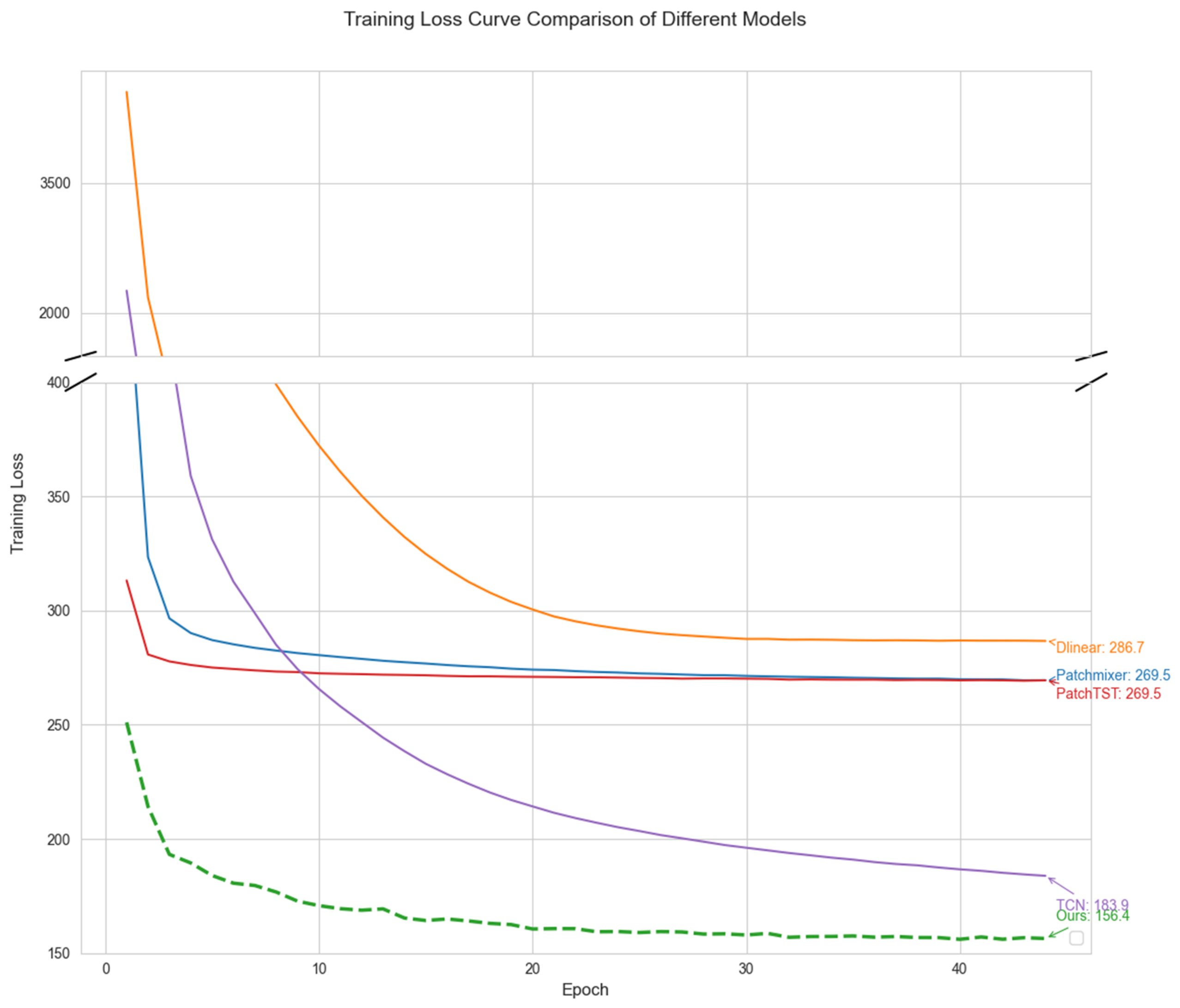


Figure 8. Training loss curve.

4. Ablation study

To assess the effectiveness of the proposed dwFCN, two model variants are constructed for the ablation study.

1) Replacing dwFCN by freMLP: A frequency domain MLP-based block employed by freDF [29]. freMLP operates complex-valued linear transform with shared matrix across the variate axis.

2) Replacing dwFCN by the proposed dwTCN, thereby completely disabling frequency learning.

As shown in Table 5, replacing dwFCN with either freMLP or dwTCN degrades forecasting performance at all steps. Specifically, it leads to a MAE increase by up to 13.2%, a MSE increase by up to 16.7%, and a MAPE increase up to 17.3%. As shown in Table 6, substituting dwFCN also results in longer training time per epoch. These findings highlight the critical role of both frequency learning and variate independence.

Table 5. Results of performance replacing dwFCN.

Models/steps		3	6	9	12	24
dwFCN (original)	MAE	6.97	7.14	7.65	8.47	10.64
	MSE	222.12	244.60	269.25	309.10	455.50
	MAPE	25.61	24.46	25.63	28.29	32.20
Replacing dwFCN by freMLP	MAE	7.49	7.83	8.66	9.30	10.94
	MSE	244.12	269.01	314.50	345.25	468.50
	MAPE	27.46	28.00	29.96	33.11	33.79
Replacing dwFCN by dwTCN	MAE	7.18	7.93	8.07	8.97	10.61
	MSE	232.38	275.01	296.25	344.75	459.04
	MAPE	26.44	28.08	27.53	29.66	32.67

Table 6. Results of training time (step = 9) replacing dwFCN.

Models	Training time per epoch (s)
dwFCN (original)	51.05
Replacing dwFCN by freMLP	52.52
Replacing dwFCN by dwTCN	87.50

5. Extended discussion

5.1. Interpretability of decomposed signal

We further discuss the challenge of model's interpretability for highway operators. Since providing solutions to model explainability is beyond the scope of this study, we measure interpretability by examining the extent to which different traffic patterns embedded in the raw signal are separated distinctly and whether the resulting sub-signals are easily distinguishable. To evaluate VMD's contribution to improved interpretability, a frequency-amplitude analysis on signal components is conducted. For comparison, two decomposition algorithms, moving average (MA) and seasonal-trend decomposition using Loess (STL), are also included.

Figure 9 illustrates a one-day sample of a univariate time series before and after VMD. The left panel shows the common temporal representation of time series (value at time step), while the right panel displays the frequency representation of signal's amplitude (how strong the fluctuation is) and regularity under normalized positive frequency (how fixed the fluctuating pattern is). As illustrated in the temporal panel, the decomposed DC captures the dominant stable trend of daily traffic, while $AC[1]$, $AC[2], and\ AC[3]$ capture minor and short traffic changes. The four sub-signals on the temporal panel can be summed up to fully reconstruct the raw signal. Compared to directly learning from raw signal without separation, VMD proactively guides the process in a manner aligning with intuition (using DC for long-term trends and ACs for short-term dynamics), thereby potentially reducing operator's comprehending difficulty. In the frequency panel, the single-centered frequency features of DC (≈ 0), $AC[1]$ (≈ 0.1), $AC[2]$ (≈ 0.32), and $AC[3]$ (≈ 0.39), together with a shallow frequency interval, are observed, indicating they are steady and unique sub-signals; in other words, they are more "purified" patterns, thereby facilitating simpler downstream models.

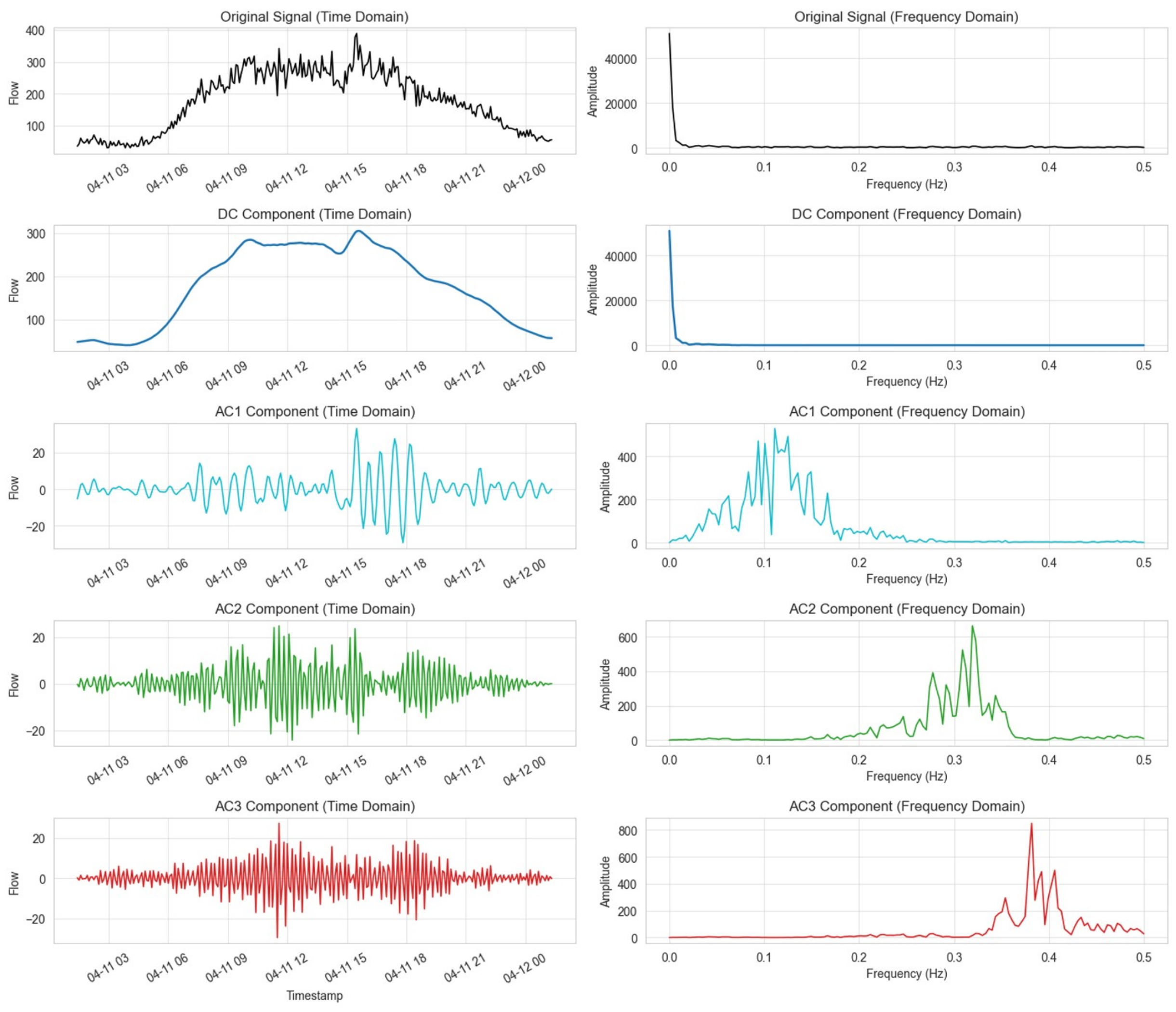


Figure 9. Illustration of VMD results.

Figure 10 illustrates the results using STL. Compared to Figure 9, STL can effectively extract the trend but fail to fully interpret the ideal *seasonality* and *residual*. As the frequency panel reveals, the *seasonality* holds multi-centered frequency (six peaks are observed), implying the existence of multiple unseparated seasoned patterns. Regarding the *residual*, its self-dependency is observed and unable to be treated as noise, thereby requiring extra modeling to ensure no pattern is lost. Consequently, STL offers less interpretability than VMD in our case, potentially requiring subsequent models to learn from entangled patterns.

Figure 11 illustrates the results using MA. MA is quite straightforward compared to VMD and STL, smoothing the raw signal to extract its trend and treating the remainder as *seasonality*. However, as the frequency panel suggests, the *seasonality* reveals a high mixture of patterns, indicating little traffic-related meanings. In our case, MA offers the relative lowest interpretability, where subsequent models may still be presented with entangled and complex data patterns.

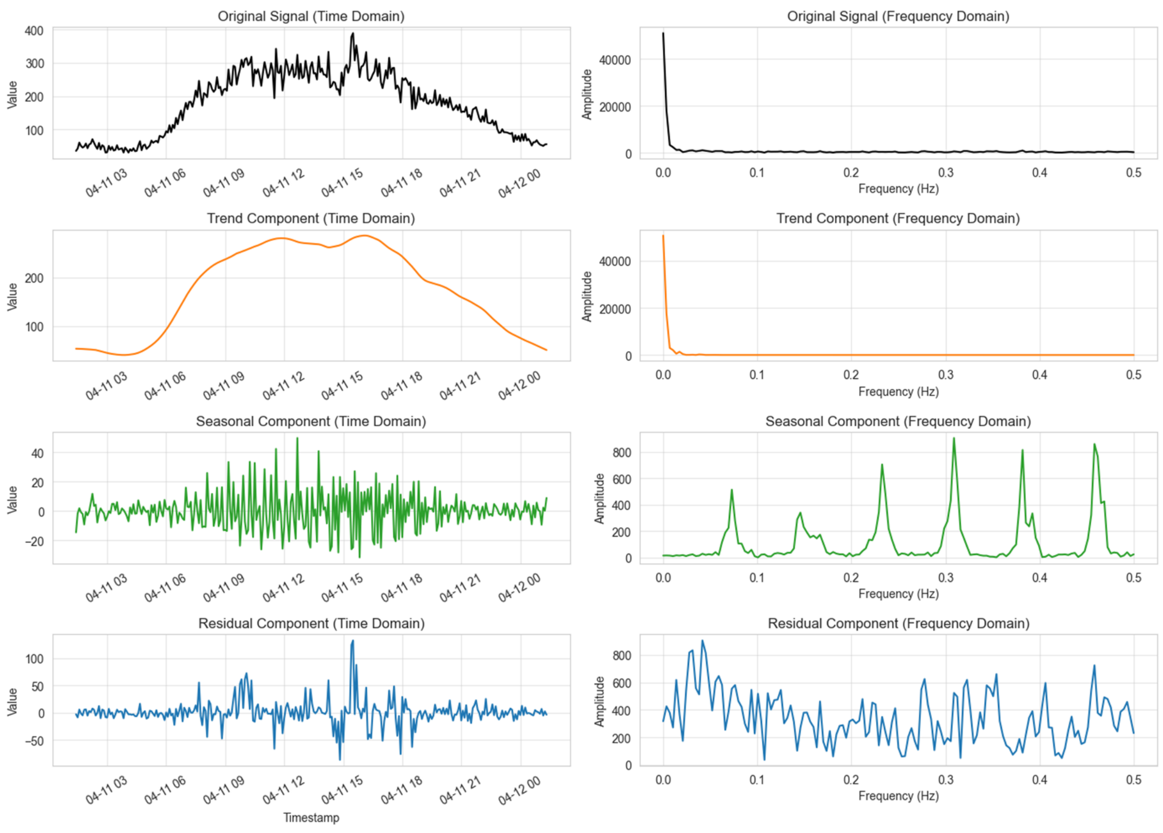


Figure 10. Illustration of STL results.

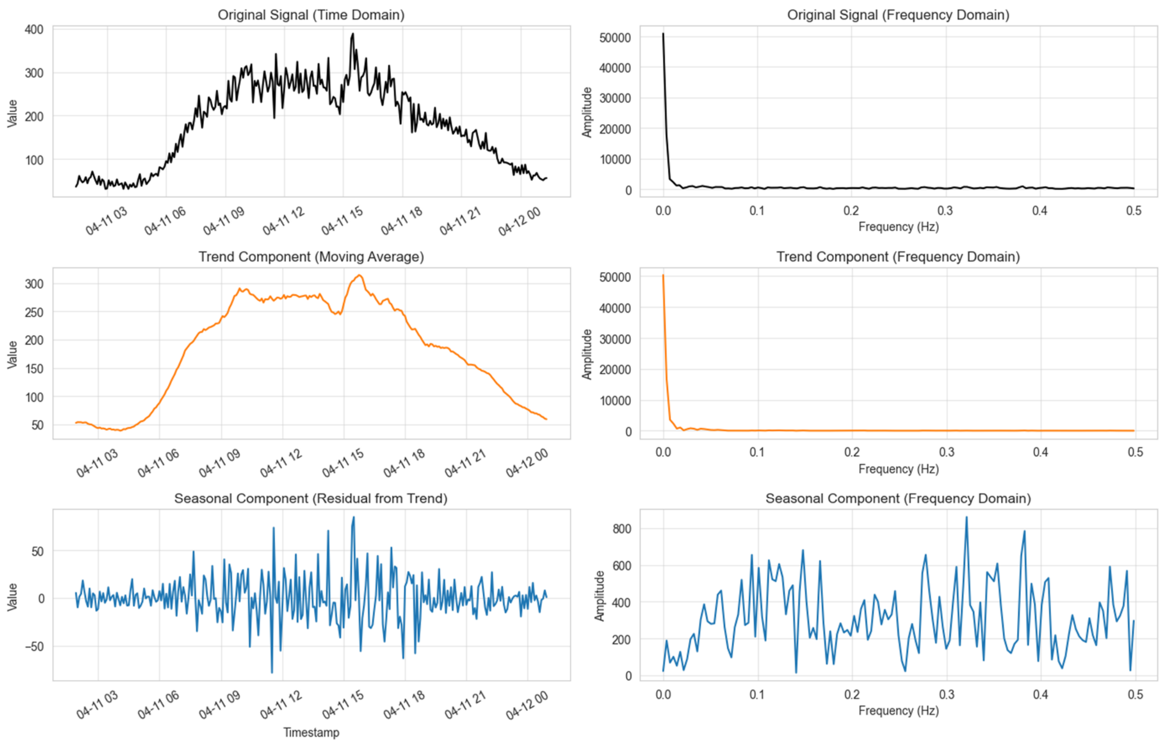


Figure 11. Illustration of MA results.

5.2. Variate heterogeneity

To further investigate inter-variate heterogeneity embedded in highway traffic flow, an empirical analysis is conducted using gantries around the Yinshan Intersection, Suzhou (for the location, see Figure 12). The fast dynamic time warping (FastDTW) algorithm is employed to evaluate time series similarity between variates.

As shown in Figure 12, geographically adjacent gantries, such as 10,296, 10,297, and 10,977, do not consistently exhibit similar temporal patterns. The FastDTW between gantries varies significantly across different months and weeks, reflecting the stochastic nature of short-term traffic flow. These variations are primarily driven by factors such as road incidents and localized unique demand patterns.

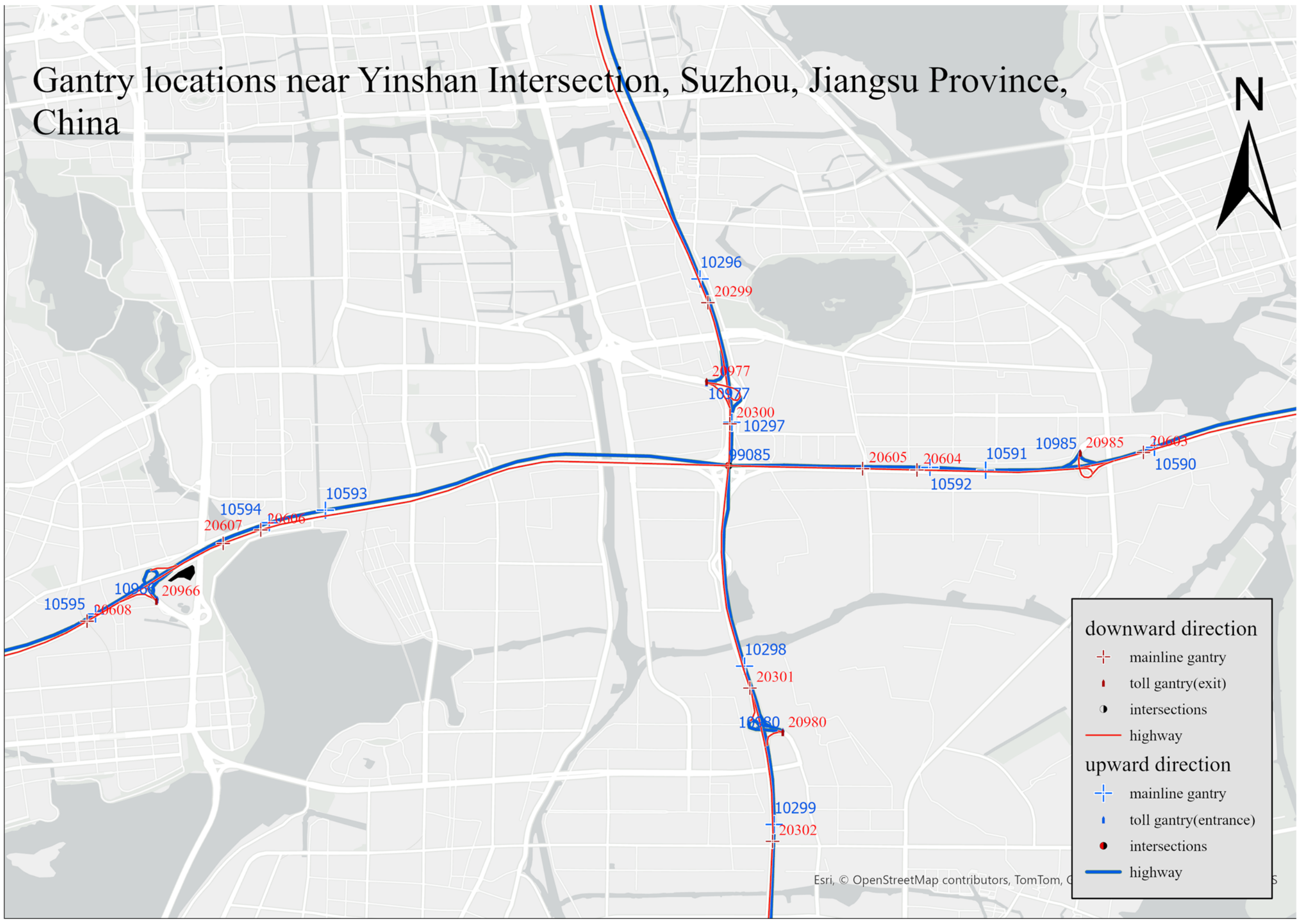
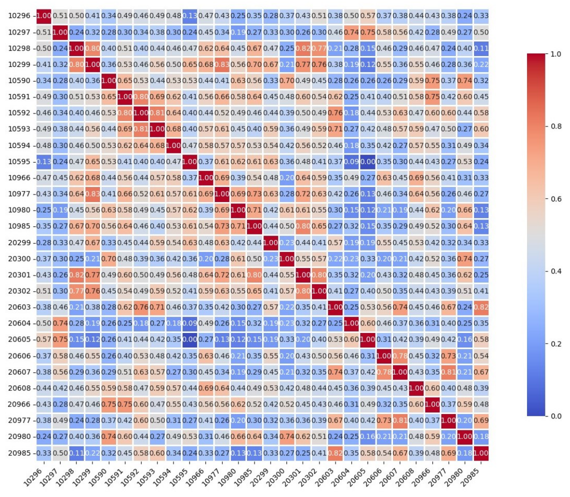
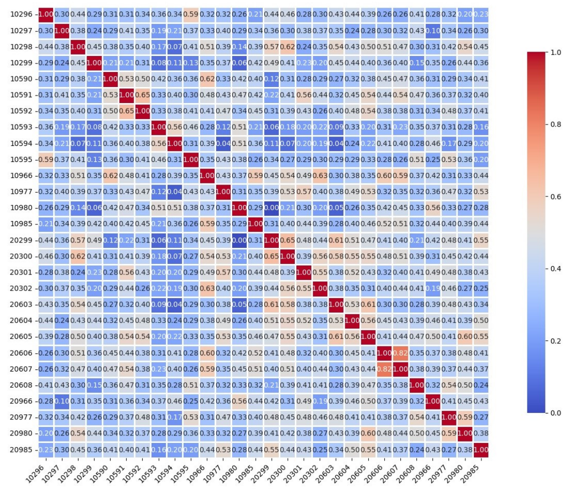


Figure 12. Network topology of Yinshan Intersection, Suzhou.

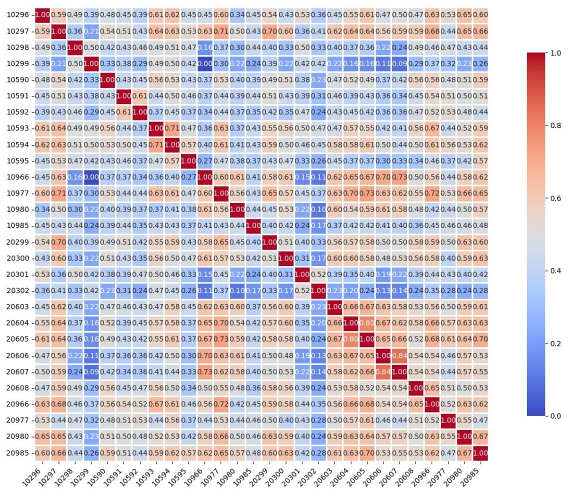
An extra finding in Figure 12 is that different gantry types and locations would complicate the highway network representation, as direction changes and turning back in-place are allowed at intersections and toll stations. Also, real-time changes of network relations may also occur due to accidents, bad weather, and reacting actions such as road and ramp closure. Operators are suggested to carefully establish the regional highway network structure, with timely updates to spatial relations and recordings of incidents such as road closures.



(a). FastDTW matrix of 2023-04-01 8:00 to 8:30.



(b). FastDTW matrix of 2023-05-03 15:30 to 16:30.



(c). FastDTW matrix of 2023-06-01 7:15 to 9:15.

Figure 13. Inter-variate similarity during different time periods.

6. Conclusions

This paper addresses the problem of large-scale highway traffic flow forecasting with challenges of variate heterogeneity for engineering practicality. To achieve performance improvement without increased computational burden, we innovatively introduce VMD and establish a multi-tasking deep-learning architecture. We propose a novel temporal-frequency pure convolutional network to improve variate-specific pattern learning and mitigate negative effect of cross-variate mixing. To verify the proposed model, we conduct a case study on a regional network in Jiangsu Province China. By comparison with competitive baselines, an MAE reduction of up to 26.7% and a MAPE reduction up to 29.9% is observed, while also achieving fast training with improved interpretability.

Based on our findings, addressing variate independence during learning is beneficial for real-world traffic flow forecasting. Since decomposition methods remain relatively underexplored in practical deployment, many effective algorithms like VMD can be further applied with deep learning. When appropriately designed, low-complexity models rooted in frequency learning and convolutional networks can also perform with high competitiveness.

Based on our study results, we suggest the following measures aimed at highway operators:

• A greater attention on decomposition methods during decision making, as they hold potential for improving forecasting and comprehension and provide versatility toward future upgrades.

• A tighter link between forecast outputs and downstream actions. Long-term forecasts should inform resource allocation for upcoming congestion and potential incidents. Short-term forecasts, on the other hand, should trigger real-time actions, for example, when predicted flow exceeds a threshold, predefined actions such as ramp metering, congestion information broadcast, re-routing suggestions to autonomous electric vehicles [46], and dynamic hard-shoulder running can be immediately activated.

• Well-planned maintenance plan for forecasting models, including error monitoring and necessary retraining around major events and holidays. Equally important is recording the full cycle of forecast–observation–decision–feedback, which ensures traceability and essential data for future improvements.

These managerial actions will further strengthen the utility of forecasting models in practical scenarios.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This study is supported by the Youth Program of National Natural Science Foundation of China (No. 72301066), Fundamental Research Funds for the Central Universities, China (No. 2242024RCB0048), Natural Science Foundation of Jiangsu Province (No. SBK2024021394), Wuxi Science and Technology Development Fund Project (No. K20231014).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. K. Huang, Z. Zhang, X. Wang, Y. Tao, Z. Liu, Life-cycle carbon emissions of autonomous electric vehicles in varying traffic situations, *Transp. Res. Part D Transp. Environ.*, **146** (2025), 104871. https://doi.org/10.1016/j.trd.2025.104871
2. X. Zhang, R. R. Chowdhury, R. K. Gupta, J. Shang, Large language models for time series: A survey, in *the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, (2024), 8335–8343. https://doi.org/10.24963/ijcai.2024/921
3. Z. Liu, Z. Zhou, Z. Gu, S. Liu, P. Liu, Y. Zhang, et al., TRIP: Transport reasoning with intelligence progression — a foundation framework, *Transp. Res. Part C Emerging Technol.*, **179** (2025), 105260. https://doi.org/10.1016/j.trc.2025.105260
4. B. M. Williams, Multivariate vehicular traffic flow prediction: Evaluation of arimax modeling, *Transp. Res. Rec.*, **1776** (2001), 194–200. https://doi.org/10.3141/1776-25
5. N. G. Polson, V. O. Sokolov, Deep learning for short-term traffic flow prediction, *Transp. Res. Part C Emerging Technol.*, **79** (2017), 1–17. https://doi.org/10.1016/j.trc.2017.02.024
6. Z. Liu, C. Lyu, Z. Wang, S. Wang, P. Liu, Q. Meng, A Gaussian-process-based data-driven traffic flow model and its application in road capacity analysis, *IEEE Trans. Intell. Transp. Syst.*, **24** (2023), 1544–1563. https://doi.org/10.1109/TITS.2022.3223982
7. Z. Liu, C. Lyu, J. Huo, S. Wang, J. Chen, Gaussian process regression for transportation system estimation and prediction problems: The deformation and a Hat Kernel, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 22331–22342. https://doi.org/10.1109/TITS.2022.3155527
8. M. Liu, A. Zeng, Q. Lai, Q. Xu, T-WaveNet: Tree-structured wavelet neural network for sensor-based time series analysis, preprint, arXiv:2012.05456.
9. Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, J. Liu, LSTM network: A deep learning approach for short-term traffic forecast, *IET Intell. Transp. Syst.*, **11** (2017), 68–75. https://doi.org/10.1049/iet-its.2016.0208
10. A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in *the 37th AAAI Conference on Artificial Intelligence (AAAI)*, AAAI Press, (2023), 11121–11128. https://doi.org/10.1609/aaai.v37i9.26317
11. A. Das, W. Kong, A. Leach, S. Mathur, R. Sen, R. Yu, Long-term forecasting with tide: Time-series dense encoder, preprint, arXiv:2304.08424.
12. S. Bai, J. Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, preprint, arXiv:1803.01271.
13. B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in *the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, (2018), 3634–3640, https://doi.org/10.24963/ijcai.2018/505
14. Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in *the 28th International Joint Conference on Artificial Intelligence (IJACI)*, (2019), 1907–1913.
15. Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the Dots: Multivariate time series forecasting with graph neural networks, in *the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2020), 753–763. https://doi.org/10.1145/3394486.3403118

16. Z. Gong, Y. Tang, J. Liang, PatchMixer: A patch-mixing architecture for long-term time series forecasting, preprint, arXiv:2310.00655.
17. Y. Nie, N. H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, in *the 11th International Conference on Learning Representations (ICLR)*, (2023), 1–24.
18. D. Luo, X. Wang, ModernTCN: A modern pure convolution structure for general time series analysis, in *the 12th International Conference on Learning Representations (ICLR)*, (2024), 1–43.
19. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need, in *Advances in Neural Information Processing Systems 30 (NeurlPS)*, (2017), 1–11.
20. Y. Zhang, J. Yan, Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting, in *the 11th International Conference on Learning Representations (ICLR)*, (2023), 1–21.
21. H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, in *Advances in Neural Information Processing Systems 34 (NeurlPS)*, (2021), 22419–22430.
22. H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, et al., Informer: Beyond efficient transformer for long sequence time-series forecasting, in *the 35th AAAI Conference on Artificial Intelligence (AAAI)*, **35** (2021), 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325
23. S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Atention based spatial-temporal graph convolutional networks for traffic flow forecasting, in *the 33th AAAI Conference on Artificial Intelligence (AAAI)*, **33** (2019), 922–929. https://doi.org/10.1609/aaai.v33i01.3301922
24. Q. Wu, C. Yang, W. Zhao, Y. He, D. Wipf, J. Yan, DIFFormer: Scalable (graph) transformers induced by energy constrained diffusion, in *the 11th International Conference on Learning Representations (ICLR)*, (2023), 1–26.
25. Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, et al., iTransformer: Inverted transformers are effective for time series forecasting, in *the 12th International Conference on Learning Representations (ICLR)*, (2024), 1–25.
26. A. Katharopoulos, A. Vyas, N. Pappas, F. Fleuret, Transformers are RNNs: Fast autoregressive transformers with linear attention, in *the 37th International Conference on Machine Learning (ICML)*, (2020), 5156–5165.
27. C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, et al., Do transformers really perform badly for graph representation?, in *Advances in Neural Information Processing Systems 34 (NeurlPS)*, (2021), 28877–28888.
28. H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2d-variation modeling for general time series analysis, in *the 11th International Conference on Learning Representations (ICLR)*, (2023), 1–23.
29. Z. Xu, A. Zeng, Q. Xu, FITS: Modeling time series with *10k* parameters, in *the 12th International Conference on Learning Representations (ICLR)*, (2024), 1–24.
30. H. Wang, L. Pan, Z. Chen, D. Yang, S. Zhang, Y. Yang, et al., FreDF: Learning to forecast in the frequency domain, in *the 13th International Conference on Learning Representations (ICLR)*, (2025), 1–30.

31. Y. He, L. Li, X. Zhu, K. L. Tsui, Multi-graph convolutional-recurrent neural network (MGC-RNN) for short-term forecasting of transit passenger flow, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 18155–18174. https://doi.org/10.1109/TITS.2022.3150600
32. C. Zheng, X. Fan, C. Wen, L. Chen, C. Wang, J. Li, DeepSTD: Mining spatio-temporal disturbances of multiple context factors for citywide traffic flow prediction, *IEEE Trans. Intell. Transp. Syst.*, **21** (2019), 3744–3755. https://doi.org/10.1109/TITS.2019.2932785
33. J. Lu, X. Han, Y. Sun, S. Yang, CATS: Enhancing multivariate time series forecasting by constructing auxiliary time series as exogenous variables, in *the 41st International Conference on Machine Learning (ICML)*, (2024), 1–17.
34. X. Huang, S. Belongie, Arbitrary style transfer in real-time with adaptive instance normalization, in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, (2017), 1510–1519. https://doi.org/10.1109/ICCV.2017.167
35. N. Bjorck, C. P. Gomes, B. Selman, K. Q. Weinberger, Understanding batch normalization, in *Advances in Neural Information Processing Systems 31 (NeurlPS)*, (2018), 1–12.
36. T. Kim, J. Kim, Y. Tae, C. Park, J. H. Choi, J. Choo, Reversible instance normalization for accurate time-series forecasting against distribution shift, in *the 10th International Conference on Learning Representations (ICLR)*, (2022), 1–25.
37. K. Dragomiretskiy, D. Zosso, Variational mode decomposition, *IEEE Trans. Signal Process.*, **62** (2013), 531–544. https://doi.org/10.1109/TSP.2013.2288675.
38. A. Taha, N. Nazih, P. Makeen, Wind speed prediction based on variational mode decomposition and advanced machine learning models in zaafarana, Egypt, *Sci. Rep.*, **15** (2025), 15599. https://doi.org/10.1038/s41598-025-98543-6
39. E. A. Tuncar, S. Saglam, B. Oral, A review of short-term wind power generation forecasting methods in recent technological trends, *Energy Rep.*, **12** (2024), 197–209. https://doi.org/10.1016/j.egyr.2024.06.006
40. G. Liu, Y. Ma, N. Wang, Rolling bearing fault diagnosis based on SABO-VMD and weighted Manhattan-KNN, *Sensors*, **24** (2024), 5003. https://doi.org/10.3390/s24155003
41. N. ur Rehman, H. Aftab, Multivariate variational mode decomposition, *IEEE Trans. Signal Process.*, **67** (2019), 6039–6052. https://doi.org/10.1109/TSP.2019.2951223
42. C. Xu, J. Yang, T. Zhang, K. Li, K. Zhang, Adaptive parameter selection variational mode decomposition based on a novel hybrid entropy and its applications in locomotive bearing diagnosis, *Measurement*, **217** (2023), 113110. https://doi.org/10.1016/j.measurement.2023.113110
43. H. Jia, P. Cao, T. Liang, C. F. Caiafa, Z. Sun, Y. Kushihashi, et al., Short-time variational mode decomposition, *Signal Process.*, **238** (2026), 110203. https://doi.org/10.1016/j.sigpro.2025.110203
44. K. Dragomiretskiy, D. Zosso, Two-dimensional variational mode decomposition, in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, (2015), 197–208. https://doi.org/10.1007/978-3-319-14612-6_15
45. D. P. Bertsekas, Chapter 1 - Introduction, in *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, (1982), 1–94, https://doi.org/10.1016/B978-0-12-093480-5.50005-2
46. Z. Liu, Y. Wang, Q. Cheng, H. Yang, Analysis of the information entropy on traffic flows, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 18012–18023. https://doi.org/10.1109/TITS.2022.3155933

47. H. Liu, Z. Dong, R. Jiang, J. Deng, J. Deng, Q. Chen, et al., Spatio-Temporal adaptive embedding makes vanilla transformer SOTA for traffic forecasting, in *the 32nd ACM International Conference on Information and Knowledge Management (CIKM)*, (2023), 4125–4129. https://doi.org/10.1145/3583780.3615160