Long-Term Series Forecasting via Conditional Variational Autoencoder Modeling Non-Stationarity

Baowen Xu, Song Gao*, and Hongxin Meng

National Meteorological Center, Beijing, China
Email: xubaowen21@mails.ucas.ac.cn (B.X.); 15154555342@163.com (S.G.); 1992433943@qq.com (H.X.)
*Corresponding author
Manuscript received July 13; accepted September 25, 2025; published November 28, 2025

***Abstract*—Accurate long-term time series forecasting is crucial for public resource allocation, industrial optimization, and epidemic prevention. Yet, the inherent non-stationarity and periodicity of time series make this task highly challenging. To address these issues, we propose a Conditional Variational Autoencoder (CVAE) framework tailored for non-stationary time series. By introducing noise, CVAE enhances adaptability to distribution shifts. A Mixture of Experts encodes features to capture periodic patterns, while the decoder incorporates seasonal-trend disentanglement, adaptive Nadaraya-Watson regression, difference–cumsum operations, and hypergraph convolution to model non-stationarity. Furthermore, sequence stabilization reintroduces non-stationary information into predictions to alleviate distribution drift. Extensive experiments on real-world industrial datasets and public benchmarks verify that our approach effectively improves long-term forecasting accuracy.**

***Keywords*—long-term prediction, non-stationary, distribution drift, conditional variational autoencoder**

I. Introduction

Time series prediction is fundamental in diverse domains such as finance, weather forecasting, epidemic monitoring, energy consumption, and traffic planning. Extending the prediction horizon enables long-term planning and early warning [1]. For instance, long-term predictions of battery temperature fields are crucial for electric vehicle safety, while in steel production, forecasting temperature variables in coke ovens supports stable output. However, prediction accuracy typically degrades as the forecasting horizon extends, making long-term forecasting highly challenging.

Deep neural networks [2–9], particularly Transformer-based models [1, 10, 11], have advanced long-term forecasting by capturing long-range dependencies. Yet, their high computational and memory costs hinder scalability to practical settings, especially with limited data. Recent studies [2, 5] further suggest that simpler linear models can sometimes outperform Transformers, raising doubts about their efficacy. In real-world scenarios, most time series are inherently non-stationary, with shifting data distributions over time (distribution or concept drift). As shown in Fig. 1, the mean and variance of CO and CO_2 in different data splits vary significantly, highlighting the drift problem that undermines forecasting accuracy.

To address these challenges, we propose a Conditional Variational Autoencoder (CVAE) framework explicitly designed for non-stationary time series. By introducing noise, the model adapts to varying input distributions, reducing drift-induced errors. The encoder employs a Mixture of Experts to capture multi-scale periodic patterns, while the decoder integrates: (1) a Seasonal-Trend disentangler for decomposition, (2) hypergraph convolution for channel-wise correlations, (3) adaptive Nadaraya-Watson regression for noise filtering, and (4) difference–cumsum operations for non-stationary modeling. Sequence stabilization further reintroduces original non-stationary information, alleviating distribution drift.

The key contributions of this work are:

- We propose a CVAE-based framework for long-term time series forecasting, explicitly modeling non-stationarity and mitigating distribution drift to improve accuracy.
- Our model integrates Mixture of Experts for periodicity modeling and a carefully designed decoder combining Seasonal-Trend disentanglement, adaptive regression, difference–cumsum operations, and hypergraph convolution, alongside sequence stabilization.
- Extensive experiments on real-world industrial datasets and public benchmarks demonstrate state-of-the-art performance and strong cross-domain generalization.

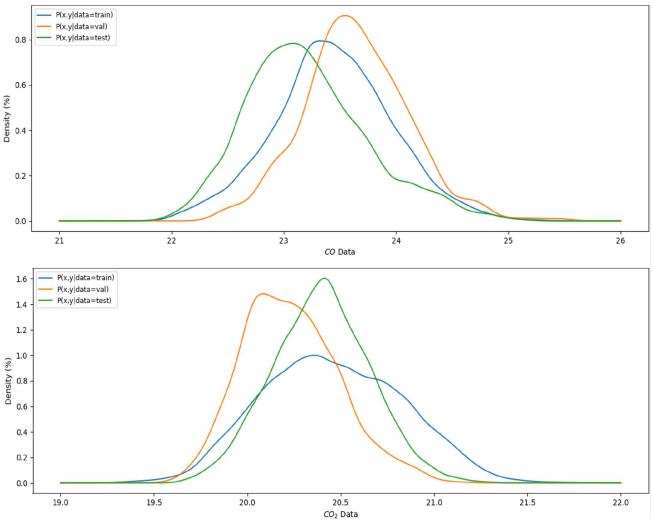

Fig. 1. Illustration of temporal distribution shifts in blast furnace gas composition data from real-world steel manufacturing processes.

II. Methodology

A. Problem Formulation

We formulate the time series prediction task within a sliding window framework. Given fixed input and output window lengths, denoted by I and T respectively, we define the input sequence at time t as $X_t = (x_{t-I+1}, \ldots, x_t)^\top \in \mathbb{R}^{I \times d_x}$. The corresponding ground truth

future sequence is $Y_t = (x_{t+1}, \ldots, x_{t+T})^\top \in \mathbb{R}^{T \times d_y}$, and the model's prediction is $\hat{Y}_t = (\hat{x}_{t+1}, \ldots, \hat{x}_{t+T})^\top \in \mathbb{R}^{T \times d_y}$. Here, each $x_t \in \mathbb{R}^C$ represents the state at timestamp t with C variables, and typically $d_x = d_y = C$. The model's objective is to learn the target predictive distribution $p_{\hat{Y}_t}(\{\hat{x}_{t+1:t+T}\}|\{x_{t-I+1:t}\})$, from which predictions can be sampled via $\hat{Y}_t \sim p_{\hat{Y}_t}(\cdot|X_t)$.

B. Overall Architecture

To improve long-term prediction performance for non-stationary time series, we propose a novel model based on a Conditional Variational Autoencoder (CVAE). This approach is designed to capture the underlying data distribution and mitigate distributional drift, leading to more accurate long-term forecasts. The overall architecture is illustrated in Fig. 2.

Our CVAE framework is composed of three main neural network modules [12–14]:

- A **conditional prior network** $p_\theta(Z|X_t)$, which models the latent variable Z based on the input sequence X_t.
- A **recognition network** $q_\phi(Z|X_t, Y_t)$, which approximates the posterior distribution of Z by incorporating information from the ground truth sequence Y_t during training.
- A **prediction generation network (Decoder)** $p_\psi(\hat{Y}_t|X_t, Z)$, which generates the final prediction $\hat{Y}_t$ conditioned on both the input X_t and the latent variable Z.

Furthermore, to handle the non-stationarity inherent in time series data, we integrate Reversible Instance Normalization (RevIN) [15, 16] as a preprocessing and postprocessing step.

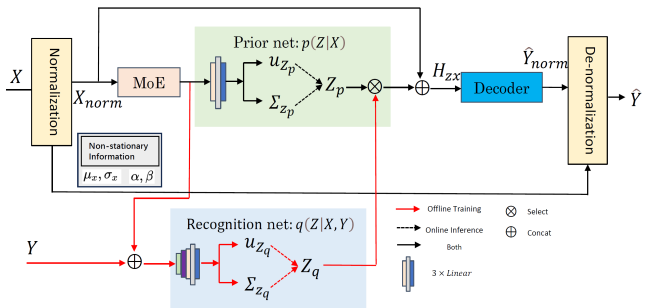

Fig. 2. Overall framework of the proposed CVAE model.

C. Series Stationarization via RevIN

Non-stationarity in time series, where statistical properties like mean and variance change over time, often causes distribution drift ($p(\hat{Y}_t|X_t) \neq p(\hat{Y}_t^r|X_t)$), degrading model performance. While traditional methods stationarize data by removing trends or seasonality, they risk obscuring the original data distribution. To address this, we employ Reversible Instance Normalization (RevIN) [16], a technique that normalizes the input sequence to stabilize its distribution for the model and later restores the original statistical properties to the output, thus avoiding over-stationarization.

Normalization Module. For each input instance X_t, we compute its mean μ_t and variance σ_t^2. The data is then normalized as follows:

$$\begin{cases} \mu_t = \frac{1}{I}\sum_{j=1}^{I} x_{t-I+j} \\ \sigma_t^2 = \frac{1}{I}\sum_{j=1}^{I} (x_{t-I+j} - \mu_t)^2 \\ X_{norm,t} = \frac{X_t - \mu_t}{\sqrt{\sigma_t^2 + \varepsilon}} \cdot \alpha + \beta \end{cases} \quad (1)$$

where $\mu_t, \sigma_t^2 \in \mathbb{R}^{d_x \times 1}$ are the instance-wise statistics, $\alpha, \beta \in \mathbb{R}^{d_x}$ are learnable affine parameters, and ε is a small constant for numerical stability.

De-normalization Module. After the core neural network $f_{model}(\cdot)$ processes the normalized input $X_{norm,t}$ to produce a normalized prediction $\hat{Y}_{norm,t} = f_{model}(X_{norm,t})$, the de-normalization module restores the original scale and shift using the stored statistics μ_t and σ_t^2:

$$\hat{Y}_t = \sqrt{\sigma_t^2 + \varepsilon} \cdot \left(\frac{\hat{Y}_{norm,t} - \beta}{\alpha}\right) + \mu_t \quad (2)$$

This ensures that the final output $\hat{Y}_t$ resides in the original data distribution.

D. CVAE Encoder

The encoder is responsible for learning a latent representation Z of the time series dynamics. It consists of the prior and recognition networks, which share a common feature extraction backbone based on a Mixture of Experts (MoE).

1) Mixture of experts backbone

To capture diverse temporal patterns, we use a Mixture of Experts (MoE) [17] architecture as our primary feature extractor. As shown in Fig. 3, the MoE module consists of n parallel expert networks (ψ_h^i) and a gating network (ψ_m) that adaptively weights and combines the experts' outputs. Let X be the input time series and X_{emb} be its timestamp embedding. The MoE output F_{en} is computed as:

$$X_i = \psi_h^i(X), W = \psi_m(X_{emb}), \quad F_{en} = \sum_{i=1}^{n} W_{:,i} X_i \quad (3)$$

where $\psi_h^i: \mathbb{R}^{I \times d_x} \to \mathbb{R}^{I \times c}$ is the i-th expert network, $\psi_m: \mathbb{R}^t \to \mathbb{R}^{n \times c}$ is the router, n is the number of experts, and c is the embedding dimension. Both ψ_h^i and ψ_m are implemented as MLPs.

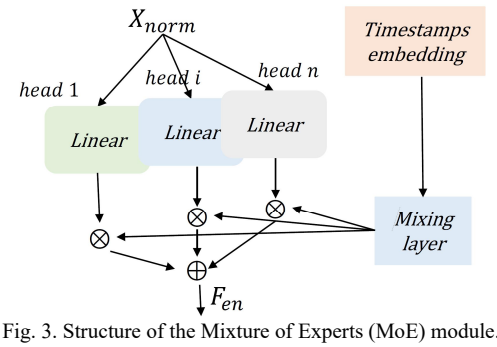

Fig. 3. Structure of the Mixture of Experts (MoE) module.

2) Conditional prior network

The prior network $p_\theta(Z|X_t)$ models the distribution of the latent variable Z conditioned solely on the input sequence X_t. It is defined as a Gaussian distribution whose parameters are derived from the MoE features F_{en}:

$$p_\theta(Z_p|X_t) := \mathcal{N}(\mu_{Z_p}, \Sigma_{Z_p})$$
$$\text{with } [\mu_{Z_p}, \Sigma_{Z_p}] = g_{Z_p}(F_{en}) \quad (4)$$

where $g_{Z_p}(\cdot)$ is a multi-layer feed-forward network. During inference, Z_p can be sampled from distribution $p_\theta(\cdot|X_t)$, i.e., $Z_p \sim p_\theta(\cdot|X_t)$.

3) Recognition network

The recognition network $q_\phi(Z|X_t, Y_t)$ is used during training to guide the learning of the latent space by incorporating information from the ground truth future Y_t. It models the approximate posterior distribution, also as a Gaussian:

$$q_\phi(Z_q|X_t, Y_t) := \mathcal{N}(\mu_{Z_q}, \Sigma_{Z_q})$$
$$\text{with } [\mu_{Z_q}, \Sigma_{Z_q}] = h_{Z_q}(\text{Concat}[F_{en}, Y_t]) \quad (5)$$

where $h_{Z_q}(\cdot)$ is a three-layer fully-connected network that takes the concatenation of MoE features and the ground truth as input. During training, Z_q can be sampled from distribution $q_\phi(\cdot|X_t, Y_t)$, i.e., $Z_q \sim q_\phi(\cdot|X_t, Y_t)$.

E. CVAE Decoder (Prediction Generation Network)

The decoder, parameterized by ψ, acts as the prediction generation network $p_\psi(\hat{Y}_t|X_t, Z)$. It takes the normalized input X_{norm} and a sampled latent variable Z to produce the final forecast. As illustrated in Fig. 4, the decoder has a parallel architecture with B identical modules whose outputs are aggregated. Each module performs a sophisticated sequence of operations to model spatial and temporal dependencies.

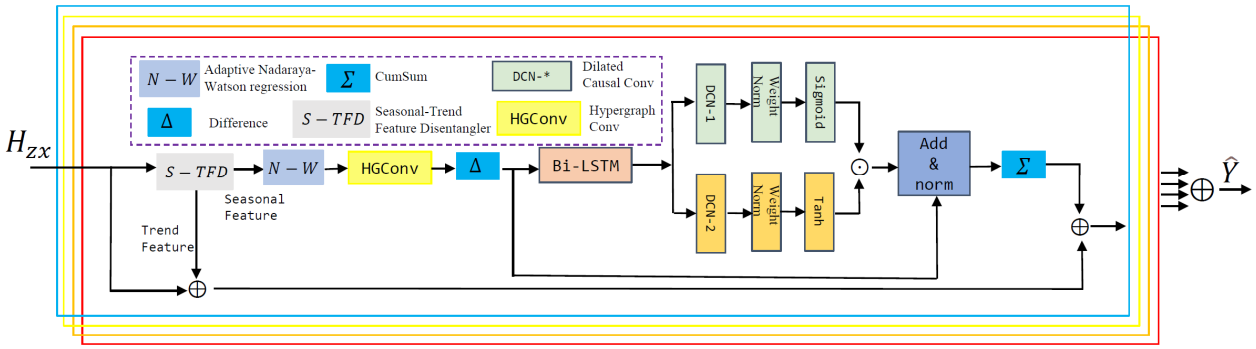

Fig. 4. Framework of the decoder. It consists of B parallel modules that process the concatenated input H_{zx} and aggregate their outputs.

1) Decoder module pipeline

The input to each of the B modules is the concatenation $H_{zx} = \text{Concat}([Z, X_{norm}])$. Within each module, the following steps are executed:

a) Seasonal-Trend Feature Disentangler (S-TFD)

We first disentangle the input features into trend and seasonal components. The trend component H^t_{zx} is extracted using an averaging pool [1], and the seasonal component H^s_{zx} is obtained by subtraction:

$$\begin{cases} H^t_{zx} = \text{AvgPool}(\text{Padding}(H_{zx})) \\ H^s_{zx} = H_{zx} - H^t_{zx} \end{cases} \quad (6)$$

Subsequent complex modeling is applied primarily to the seasonal component H^s_{zx}, while the trend component is preserved and added back later.

b) Spatial and relational feature extraction

To model complex interactions within the seasonal component, we employ three specialized modules.

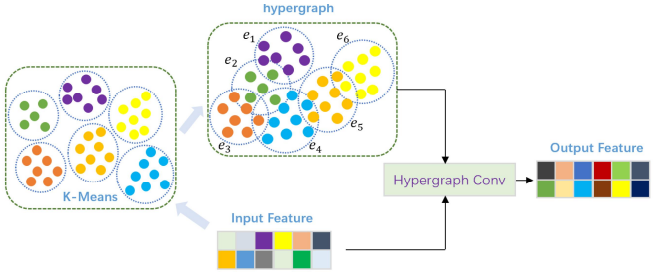

Fig. 5. The architecture of the hypergraph convolutional network.

Hypergraph convolution. Spatial feature extraction is achieved through hypergraph convolution [18], as illustrated in Fig. 5. We encode channel-wise spatial dependencies with a hypergraph $\mathcal{G} = \{\mathcal{V}_h, \mathcal{E}_h, \mathcal{W}_h\}$, where $\mathcal{V}_h$ and $\mathcal{E}_h$ denote vertices and hyperedges, and $\mathcal{W}_h = \text{diag}\{w_e\} \in \mathbb{R}^{|\mathcal{E}_h| \times |\mathcal{E}_h|}$ is the diagonal matrix of nonnegative hyperedge weights. The incidence matrix $\mathcal{H} \in \mathbb{R}^{|\mathcal{V}_h| \times |\mathcal{E}_h|}$ is

$$\mathcal{H}(v, e) = \begin{cases} 1, & v \in e, \\ 0, & v \notin e, \end{cases} \quad (7)$$

with hyperedge degree $d(e_i) = \sum_{v_j \in \mathcal{V}_h} \mathcal{H}(v_j, e_i)$ and vertex degree $d(v_j) = \sum_{e_i \in \mathcal{E}_h} w_i \, \mathcal{H}(v_j, e_i)$. Let $D_e = \text{diag}\{d(e_i)\}$ and $D_v = \text{diag}\{d(v_j)\}$. The hypergraph Laplacian matrix $\mathcal{L}_h$ is

$$\mathcal{L}_h = I - D_v^{-1/2} \mathcal{H} \mathcal{W}_h D_e^{-1} \mathcal{H}^\top D_v^{-1/2}. \quad (8)$$

Given hidden features $\mathcal{X}_h^{(i)}$, one layer of spectral hypergraph convolution [19] computes

$$\mathcal{X}_h^{(i+1)} = \mathcal{L}_h \mathcal{X}_h^{(i)} W_h^{(i)} + b_h^{(i)}, \quad (9)$$

with learnable $W_h^{(i)}$ and $b_h^{(i)}$. We construct $\mathcal{G}$ by applying K-Means [20] to the feature matrix so that each hyperedge corresponds to a cluster centroid, adaptively capturing cross-channel correlations.

Adaptive Nadaraya–Watson Regression. For a univariate reference, classical Nadaraya–Watson (NW) regression [21] estimates

$$\hat{y} = \frac{\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) y_i}{\sum_{j=1}^n K\left(\frac{x - x_j}{h}\right)}, \quad (10)$$

where $K(\cdot)$ integrates to 1. To obtain data-adaptive smoothing for multivariate features $\mathcal{X}_{\text{nw}} = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n\}$ with $\boldsymbol{\xi}_i \in \mathbb{R}^d$, we compute

$$\boldsymbol{\eta}_i = \frac{\sum_{j \neq i} K(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) \odot \boldsymbol{\xi}_j}{\sum_{j \neq i} K(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)}, \quad (11)$$

using a Gaussian kernel with learnable parameters

$$K(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) = \exp\left(\mathbf{w}_i \odot (\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)^2\right), \quad \mathbf{w}_i \in \mathbb{R}^d. \tag{12}$$

This adaptively emphasizes similar items, attenuates random components, and yields more stable representations.

Difference and CumSum Operators. To highlight short-horizon changes in seasonal patterns, we insert temporal modules (Bi-LSTM and Gated-TCN) between discrete Difference and CumSum operators. For $M = [\mathbf{m}_1, \dots, \mathbf{m}_I]^\top \in \mathbb{R}^{I \times m}$,

$$\Delta\mathbf{M} = [\Delta\mathbf{m}_1, \dots, \Delta\mathbf{m}_I]^\top, \quad \Delta\mathbf{m}_i = \mathbf{m}_{i+1} - \mathbf{m}_i, \ i \in [1, I), \tag{13}$$

and we pad $\Delta\mathbf{m}_I$ by $\Delta\mathbf{m}_{I-1}$ to keep length. The cumulative-sum operator along time is

$$\Sigma\mathbf{M} = [\Sigma\mathbf{m}_1, \dots, \Sigma\mathbf{m}_I]^\top, \tag{14}$$

so that temporal blocks act on $\Delta\mathbf{M}$ to capture multiscale, non-stationary variations, after which $\Sigma(\cdot)$ restores the sequence scale.

c) Temporal feature extraction

To model temporal dependencies, we process the features through a block designed to capture non-stationary patterns at multiple scales. This block is framed by Difference and CumSum operations. The differencing operation $\Delta\mathbf{M}$ first computes changes between adjacent time steps, focusing the model on variations in the seasonal patterns. The resulting features are then processed by a Bidirectional LSTM (Bi-LSTM) [22] and a Gated Temporal Convolutional Network (Gated-TCN) to capture both local and global temporal correlations. Finally, the CumSum operation Σ reintegrates the processed changes. This entire sequence models the temporal dynamics effectively.

The core temporal processing layers are:

$$H^{\text{lstm}} = \psi_{\text{lstm}}(H^s_{zx}) \tag{15}$$

$$\begin{aligned} g_1 = \psi_{\text{tcn}_1}(H^{\text{lstm}}), \quad g_2 = \psi_{\text{tcn}_2}(H^{\text{lstm}}) \\ g_{\text{tcn}} = \sigma(g_1) \odot \tanh(g_2) \end{aligned} \tag{16}$$

where $\psi_{lstm}(\cdot)$ is a Bi-LSTM and $\psi_{tcn_*}(\cdot)$ are 1D dilated convolutions.

d) Output aggregation

The complete processing pipeline for the b-th decoder module can be summarized as:

$$\begin{cases} H^{(b)}_{\text{spatial}} = \psi_{hgcn}(\psi_{an-w}(H^s_{zx})) \\ H^{(b)}_{\Delta} = \Delta(H^{(b)}_{\text{spatial}}) \\ H^{(b)} = \Sigma(\psi_{g-tcn}(\psi_{lstm}(H^{(b)}_{\Delta})) + H^{(b)}_{\Delta}) + H^t_{zx} \end{cases} \tag{17}$$

The final output of the entire decoder network is the sum of the outputs from all B parallel modules: $\psi_\psi(H_{zx}) = \sum_{b=1}^{B} H^{(b)}$. This aggregated feature map is then used to parameterize the final predictive distribution $\mathcal{N}(\mu_{de}, \Sigma_{de})$, from which the normalized prediction $\hat{Y}_{norm,t}$ is sampled.

F. Training Objective

During inference, the model generates predictions by sampling from the prior distribution:

$$p(\hat{Y}_t|X_t) = \int_Z p_\psi(\hat{Y}_t|X_t, Z) \cdot p_\theta(Z|X_t) dZ \tag{18}$$

During training, we optimize the model parameters by maximizing the Evidence Lower Bound (ELBO) of the data log-likelihood. Instead of using the log-likelihood for the reconstruction term, we use the Mean Absolute Error (MAE), which is more robust to outliers. The final optimization objective is to minimize the following loss function:

$$\begin{aligned} \mathcal{L} = \min_{\theta,\phi,\psi} \mathbb{E}_{\substack{\hat{Y}_t \sim p_\psi(\cdot|X_t,Z) \\ Z \sim q_\phi(\cdot|X_t,Y_t)}} [\ell_{mae}(\hat{Y}_t, Y_t)] \\ +\lambda \cdot \mathbb{KL}[q_\phi(Z_q|X_t, Y_t)||p_\theta(Z_p|X_t)] \end{aligned} \tag{19}$$

where the first term is the reconstruction loss and the second term is the Kullback-Leibler (KL) divergence, which regularizes the latent space. The hyperparameter λ balances the two terms and is set to 1 in our experiments. The complete training procedure is outlined in Algorithm 1. The training flowchart is shown in Fig. 6.

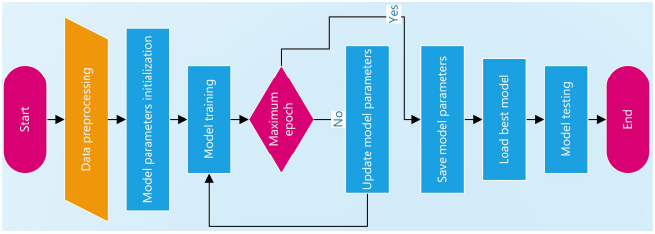

Fig. 6. Flowchart.

Algorithm 1: Training of CVAE for Time Series Prediction

1. **Initialize:** Model parameters $\theta, \phi, \psi, \alpha, \beta$; optimizer (Adam).
2. **Input:** Training set $\{(X_t, Y_t)\}$, number of epochs E, input length I, output length T.
3. **For** epoch $= 1$ **to** E:
   a. **For each batch** $(X_{\text{batch}}, Y_{\text{batch}})$:
      i. **Step 1 — Normalization:** For each $X_t \in X_{\text{batch}}$, compute μ_t, σ_t^2 and normalize to obtain $X_{norm,t}$.
      ii. **Step 2 — Encoder forward pass:**
         1. $F_{en} \leftarrow \text{MoE}(X_{norm,\text{batch}})$.
         2. $(\mu_{Z_p}, \Sigma_{Z_p}) \leftarrow g_{Z_p}(F_{en})$.
         3. $(\mu_{Z_q}, \Sigma_{Z_q}) \leftarrow h_{Z_q}(\text{Concat}[F_{en}, Y_{\text{batch}}])$.
         4. Sample $Z_q \sim \mathcal{N}(\mu_{Z_q}, \Sigma_{Z_q})$.
      iii. **Step 3 — Decoder forward pass:**
         1. $H_{zx} \leftarrow \text{Concat}([Z_q, X_{norm,\text{batch}}])$.
         2. $\hat{Y}_{norm} \leftarrow \text{Decoder}(H_{zx})$. *(Aggregate outputs of B parallel modules if applicable.)*
      iv. **Step 4 — De-normalization:** $\hat{Y}_t \leftarrow \text{RevIN\_denorm}(\hat{Y}_{norm})$ using stored μ_t, σ_t^2.
      v. **Step 5 — Loss and optimization:**
         1. Compute total loss (e.g., use Eq. (loss) in the paper) on $(\hat{Y}_t, Y_t)$.
         2. Backpropagate and update parameters $\{\theta, \phi, \psi, \alpha, \beta\}$.
   4. **End for**

III. Results and Discussion

A. Datasets

We separately validate the proposed model on a set of industrial real-world time-series datasets and a set of publicly available benchmark time-series datasets. The descriptions of the datasets used in this experiment are as follows: (1) *Blast furnace gas* (*BFG*) dataset is a practical industrial dataset collected by a steel factory in China. This dataset is not open, as it involves key technologies in the industrial manufacturing process. The dataset consists of two target variables (**CO** and **CO_2**), sampled every 10 seconds since December 2018, accumulating over 60,000 samples. The input variables are listed in Table 1. (2) **ETT** dataset, **Electricity** dataset, **Exchange** dataset, **Traffic** dataset, **Weather** dataset and **ILI** dataset are currently open benchmark time-series datasets [1].

Table 1. Input variables of blast furnace gas prediction model

Input variable	Symbol	Unit
Wind flow	P-BV10S	m^3/min
Wind pressure	P-BP10S-COOL, P-BP	kPa
Wind temperature	P-BT10S-COOL, P-BT1, P-BT2	℃
Oxygen enrichment flow	P-BVO	m^3/h
Pulverized coal injection	P-WINJ	t/h
Top temperature	P-TTOP1, P-TTOP2, P-TTOP3, P-TTOP4	℃
Top pressure	P-TP1, P-TP2, P-TP3, P-TP4	kPa
CO	P-CO	%
CO_2	P-CO2	%
Temperature measurement center point	P-CROSS1-0, P-CROSS1-1, P-CROSS2-1, P-CROSS3-1, P-CROSS4-1	℃
Temperature measurement edge point	P-CROSS1-5, P-CROSS2-5, P-CROSS3-5, P-CROSS4-5	℃
Mechanical probe	P-SLIDE1, P-SLIDE2	mm

B. Comparison Models

Transformer-based models have demonstrated significant advantages in long-term time series forecasting. We compare proposed CVAE with the state-of-the-art (sota) Transformer-based prediction models in this experiment. The baseline models selected for this experiment include six sota Transformer-based models: FEDformer [10], NS Transformer [15], Autoformer [1], Informer [11], Pyraformer [23], LogTrans [24], and Reformer [25]

C. Settings

As in previous work, we use two metrics: mean absolute error (MAE), and mean squared error (MSE). All results are run on a NVIDIA GeForce 3090 32 GB GPU using PyTorch version 2.01. All baselines are implemented according to the official open source code. Our model is optimized by Adam optimizer for up to 150 epochs. Metrics on the test set is based on the model with the best performance on the validation set. The original learning rate is 0.001, and the learning rate decays with a ratio of 0.9 after 30 and 60 epochs.

The prediction horizon for long-term time series forecasting tasks often extends far into the future. While there is no distinct boundary separating long-term and short-term predictions, a common practice is to set the minimum prediction length equal to the input length.

D. Analysis of Blast Furnace Gas Data Prediction Results

To strictly evaluate the effectiveness of the proposed model, a comprehensive analysis was conducted using real-world industrial data. The following section is organized into three parts: first, the characteristics of the blast furnace gas data are introduced; second, the specific data preprocessing techniques aimed at improving data quality are detailed; and finally, the prediction results are compared and analyzed to validate the model's performance.

1) Blast furnace gas data description

The blast furnace gas dataset used in this section is based on actual industrial data collected from a 1280 m^3 blast furnace in China. This blast furnace ironmaking system consists of the blast furnace body, charging system, hot air system, pulverized coal injection system, gas treatment system, and slag system. The blast furnace body is the smelting facility that converts iron ore into molten iron. The charging system allocates ironmaking materials to the furnace throat according to preset blast furnace distribution rules, determining the distribution of charge materials in the furnace. The hot air system primarily generates and delivers hot air to the blast furnace's rotary zone to provide the necessary heat for furnace thermal balance and tuyere fuel combustion. The pulverized coal injection system evenly and stably injects pulverized coal into the blast furnace's rotary zone, substituting cheaper coal for more expensive coke, thus reducing coke usage in the furnace. The gas treatment system mainly purifies blast furnace gas, such as CO, CO_2, and sulfur compounds, through desulfurization, denitrification, and dust removal to prevent the emission of harmful gases into the atmosphere. The slag system primarily handles the molten iron and slag produced by the blast furnace.

The specific operational process [26] of blast furnace ironmaking is as follows: Ironmaking raw materials, including sinter, coke, and flux, are charged into the blast furnace from the top through the charging system. The hot air system injects high-temperature air into the furnace, while the pulverized coal injection system blows pulverized coal into the furnace from the lower tuyere. Under the high-temperature conditions within the furnace, hydrogen and carbon monoxide, among other reducing gases, are generated. As the smelting progresses, the descending charge materials come into contact with the ascending high-temperature gas stream, which undergoes reduction, softening, melting, and decarburization to form molten iron. This molten iron, along with slag, is eventually discharged

from the iron outlet. In summary, the contents of CO and CO_2 in the blast furnace gas significantly affect steel production, making accurate long-term prediction of CO content and CO_2 content in the blast furnace gas data highly significant.

2) Data preprocessing

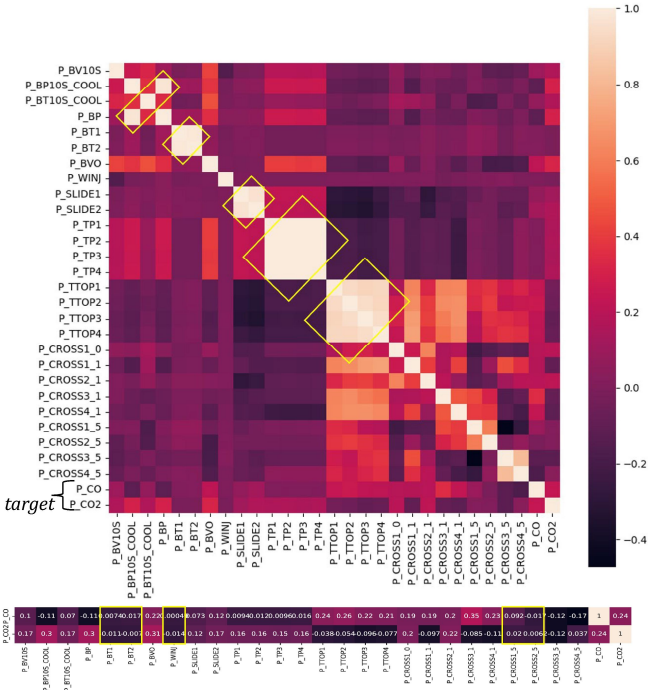

Fig. 7. Correlation matrix of different variables in the blast furnace gas dataset.

Since the data for predicting blast furnace gas content is collected in an industrial real-world setting, further detailed preprocessing is required. As shown in Table 1, there are a large number of input variables, and not all features are important. We use the correlation matrix between variables to filter out redundant feature variables. Fig. 7 depicts the correlation matrix of the relationships between variables. From Fig. 7, it is evident that variables P-BT1, P-BT2, P-WINJ, P-CROSS1-5, and P-CROSS2-5 are almost uncorrelated with the target variables P-CO and P-CO2. Therefore, these feature variables should be removed. From Fig. 7, it can also be observed that variables P-BP10S-COOL and P-BP are highly correlated, variables P-BT1 and P-BT1 are highly correlated, variables P-SLIDE1 and P-SLIDE2 are highly correlated, variables P-TP1, P-TP2, P-TP3, and P-TP4 are highly correlated, and variables P-TTOP1, P-TTOP2, P-TTOP3, and P-TTOP4 are highly correlated. We can merge highly correlated variables using the $max(\cdot,\ldots,\cdot)$ function, for example, P-BT=max(P-BT1, P-BT2).

The blast furnace gas data, generated during the ironmaking process, contains some noise, missing values, and outliers. In this section, outliers are removed using boxplot method, and missing values are filled using spline interpolation. Due to the large differences in the magnitude of different features, input features are normalized to the range [0,1].

3) Comparison and analysis

Table 2 displays the evaluation results of different models on the blast furnace gas dataset for two target variables, CO and CO_2 content, under various prediction length settings. From Table 2, it can be observed that the proposed CVAE model achieves state-of-the-art performance on the CO_2 content target variable. Additionally, for the CO content target variable, the CVAE model reaches sota on the prediction horizon of 96 and 192, and also competes with Transformer-based models on the prediction horizon of 336 and 720. Fig. 8 and Fig. 9 visualize the comparative results of different models for CO and CO_2 content predictions against the ground truth, with an input/output setting of 92/92. In summary, on the blast furnace gas dataset, the proposed CVAE model demonstrates effectiveness.

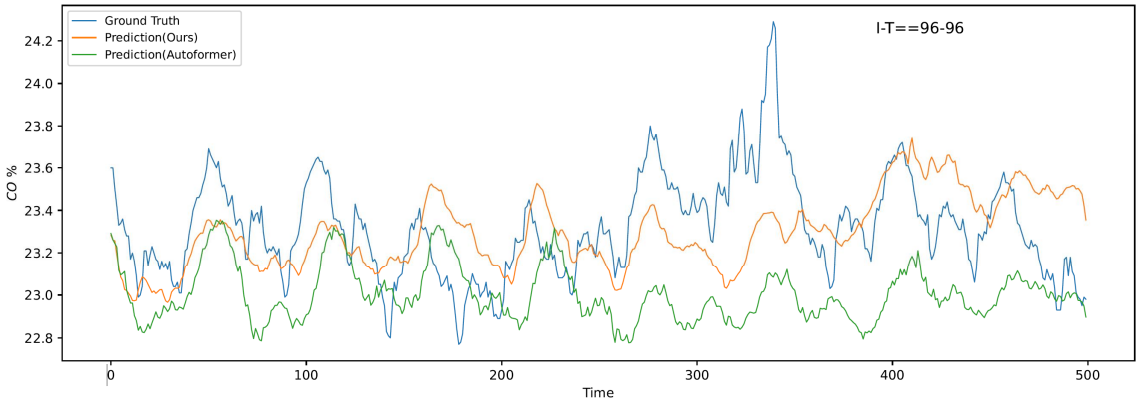

Fig. 8. Visualization of CO Content Prediction.

Table 3 presents the computational efficiency metrics on CO data from blast furnace gas in real industrial scenarios, including training time, inference time, training FLOPs, and GPU memory consumption. As shown in Table 3, the proposed CVAE demonstrates a clear competitive advantage in terms of computational efficiency compared with other baseline models, and further indicates that the method can fully meet the efficiency requirements for practical applications in real blast furnace gas industrial settings.

E. Analysis of Benchmark Time Series Data Prediction Results

The evaluation results in Table 2 of the benchmark time series dataset indicate that the predicted outcomes of the proposed CVAE are close to or even superior to the state-of-the-art Transformer-based baseline. Across the *Exchange*, *Weather*, and *ILI* datasets, the proposed CVAE outperforms the leading comparative models at all prediction time steps. On the remaining three datasets, the

proposed CVAE performs partially better than the baseline at all time steps, and in some evaluation metrics where it falls short of the baseline, the results are only slightly inferior. Overall, the model proposed in this paper proves to be effective on publicly available benchmark time series datasets.

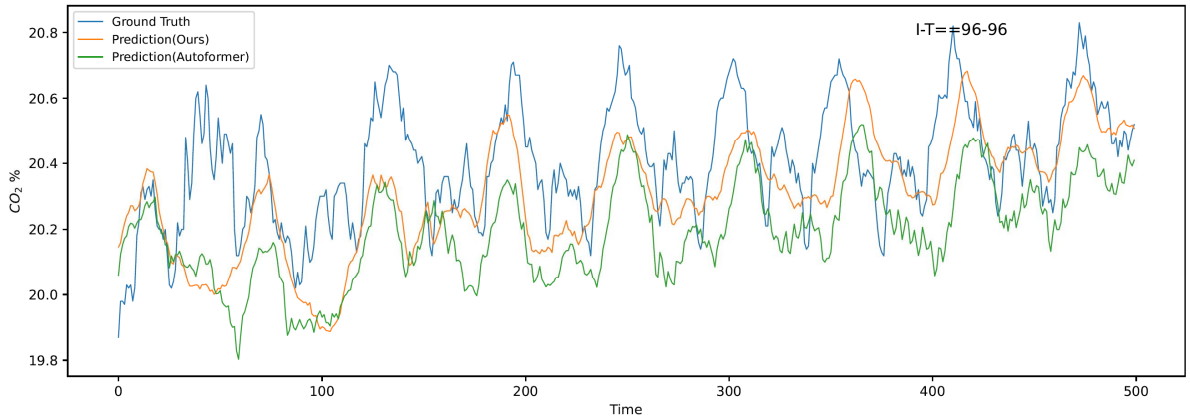

Fig. 9. Visualization of CO_2 Content Prediction.

Table 2. Performance Comparison on BFG and Benchmark Time-Series Datasets. Historical horizon: 36 for ILI, 96 for others.Prediction lengths: {96, 192, 336, 720} (24/36/48/60 for ILI).Bold: SOTA; Underline: Second-best.}

Dataset / Len		CVAE		FEDformer		NS Trans.		Reformer		Pyraformer		Autoformer		Informer		LogTrans	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
CO	96	**0.237**	**0.096**	0.300	0.166	0.505	0.372	0.486	0.349	0.410	0.281	0.444	0.298	0.558	0.366	0.487	0.321
	192	**0.345**	**0.183**	0.366	0.208	0.488	0.311	0.470	0.292	0.397	0.273	0.396	0.266	0.561	0.310	0.499	0.341
	336	0.422	**0.263**	0.423	0.268	0.510	0.319	0.496	0.293	0.426	0.279	**0.420**	0.264	0.533	0.322	0.512	0.378
	720	0.484	0.343	0.477	0.340	0.533	0.332	0.526	0.375	0.549	0.483	**0.465**	**0.339**	0.583	0.496	0.554	0.390
CO_2	96	**0.139**	**0.033**	0.198	0.066	0.292	0.123	0.267	0.118	0.209	0.071	0.219	0.081	0.299	0.134	0.311	0.141
	192	**0.170**	**0.047**	0.255	0.102	0.310	0.211	0.299	0.187	0.249	0.099	0.255	0.110	0.321	0.220	0.332	0.162
	336	**0.201**	**0.065**	0.231	0.099	0.315	0.223	0.309	0.209	0.220	0.076	0.261	0.123	0.344	0.231	0.376	0.188
	720	**0.247**	**0.100**	0.256	0.100	0.333	0.237	0.312	0.211	0.257	0.105	0.277	0.133	0.355	0.265	0.390	0.201
Electricity	96	0.199	**0.303**	**0.193**	0.308	0.169	0.273	0.312	0.402	0.386	0.449	0.201	0.317	0.274	0.368	0.258	0.357
	192	0.212	0.316	**0.201**	**0.315**	0.182	0.286	0.348	0.433	0.378	0.443	0.222	0.334	0.296	0.386	0.266	0.368
	336	0.224	**0.325**	**0.214**	0.329	0.200	0.304	0.350	0.433	0.376	0.443	0.231	0.338	0.300	0.394	0.280	0.380
	720	0.260	**0.351**	**0.246**	0.355	0.222	0.321	0.340	0.420	0.376	0.445	0.254	0.361	0.373	0.439	0.283	0.376
Traffic	96	0.610	**0.337**	**0.587**	0.366	0.612	0.338	0.732	0.423	0.867	0.468	0.613	0.388	0.719	0.391	0.684	0.384
	192	0.638	**0.353**	**0.604**	0.373	0.613	0.340	0.733	0.420	0.869	0.467	0.616	0.382	0.696	0.379	0.685	0.390
	336	0.661	**0.357**	**0.621**	0.383	0.618	0.328	0.742	0.420	0.881	0.469	0.622	0.387	0.777	0.420	0.734	0.408
	720	0.680	**0.364**	**0.626**	0.382	0.653	0.355	0.755	0.423	0.896	0.473	0.660	0.408	0.864	0.472	0.717	0.396
Exchange	96	**0.122**	**0.252**	0.148	0.278	0.111	0.237	1.065	0.829	1.748	1.105	0.197	0.323	0.847	0.752	0.968	0.812
	192	**0.222**	**0.339**	0.271	0.380	0.219	0.335	1.188	0.906	1.874	1.151	0.300	0.369	1.204	0.895	1.040	0.851
	336	**0.395**	**0.464**	0.460	0.500	0.421	0.476	1.357	0.976	1.943	1.172	0.509	0.524	1.672	1.036	1.659	1.081
	720	**1.050**	**0.781**	1.195	0.841	1.092	0.769	1.510	1.016	2.085	1.206	1.447	0.941	2.478	1.310	1.941	1.127
Weather	96	**0.201**	**0.252**	0.217	0.296	0.173	0.223	0.689	0.596	0.622	0.556	0.266	0.336	0.300	0.384	0.458	0.490
	192	**0.256**	**0.296**	0.276	0.336	0.245	0.285	0.752	0.638	0.739	0.624	0.307	0.367	0.598	0.544	0.658	0.589
	336	**0.309**	**0.332**	0.339	0.380	0.321	0.338	0.639	0.596	1.004	0.753	0.359	0.395	0.578	0.523	0.797	0.652
	720	**0.382**	**0.379**	0.403	0.428	0.414	0.410	1.130	0.792	1.420	0.934	0.419	0.428	1.059	0.741	0.869	0.675
ILI	24	**2.205**	**0.941**	3.228	1.260	2.294	0.945	4.400	1.382	7.394	2.012	3.483	1.287	5.764	1.677	4.480	1.444
	36	**1.787**	**0.871**	2.679	1.080	1.825	0.848	4.783	1.448	7.551	2.031	3.103	1.148	4.755	1.467	4.799	1.467
	48	**1.849**	**0.886**	2.622	1.078	2.010	0.900	4.832	1.465	7.662	2.057	2.669	1.085	4.763	1.469	4.800	1.468
	60	**2.366**	**0.971**	2.857	1.157	2.178	0.963	4.882	1.483	7.931	2.100	2.770	1.125	5.264	1.564	5.278	1.560
ETTm2	96	**0.198**	**0.278**	0.203	0.287	0.192	0.274	0.658	0.619	0.435	0.507	0.255	0.339	0.365	0.453	0.768	0.642
	192	**0.267**	**0.321**	0.269	0.328	0.280	0.339	1.078	0.827	0.730	0.673	0.281	0.340	0.533	0.563	0.989	0.757
	336	0.344	0.371	0.325	0.366	0.334	0.361	1.549	0.972	1.201	0.845	0.339	0.372	1.363	0.887	1.334	0.872
	720	0.461	0.436	**0.421**	**0.415**	0.417	0.413	2.631	1.242	3.625	1.451	0.422	0.419	3.379	1.388	3.048	1.328

Table 3. comparison of computational efficiency metrics across models on the co dataset

Model	Training Time (s/epoch)	Inference Time (ms/sample)	Training FLOPs (G)	GPU Memory (GB)
CVAE	120.5 ± 3.1	0.45 ± 0.02	35.2	10.5
Autoformer	150.3 ± 4.5	0.60 ± 0.03	42.1	14.8
Informer	110.7 ± 2.8	0.52 ± 0.02	30.4	9.8
FEDformer	135.8 ± 3.9	0.58 ± 0.03	38.7	13.2
TimesNet	142.9 ± 4.1	0.63 ± 0.04	40.2	12.1
iTransformer	138.6 ± 4.0	0.55 ± 0.03	37.8	9.2

Note: All experiments were conducted on a single NVIDIA RTX 4090 GPU with CUDA 12.0 and PyTorch 2.1. Results are averaged over 5 independent runs, and standard deviations are reported. The batch size was set to 32, and the input sequence length was 720. FLOPs are measured in billions (10^9) and represent the computational cost for a single forward and backward pass for a batch. Inference time is measured on a per-sample basis. GPU Memory indicates the peak memory usage during training.

F. Ablations

The impact of different components of the proposed CVAE on the overall performance of the model at a prediction horizon of 720 for the CO_2 target variable is illustrated in Fig. 10. From Fig. 10, it can be seen that the various components designed in the proposed CVAE

contribute to the overall improvement in model performance. Next, we will quantitatively analyze the impact of each component in the proposed CVAE on model performance. Regarding the MAE evaluation metric, compared to the CVAE, components MoE, $HGConv$, $difference + cumsum$, $N - W$, and $S - TFD$ respectively improve the model's predictive performance by 10.71%, 31.03%, 8.26%, 9.91%, and 2.91%. Among them, the hypergraph convolution component has the greatest impact on model performance improvement, likely because hypergraph convolution can capture the complex nonlinear correlations between different variables in multivariate time series. Conversely, the $S - TFD$ component has the smallest impact on model performance improvement, possibly because the data from industrial sites exhibit strong periodicity and lack significant variability (sequences change cyclically according to each production period), making the trend-seasonal disentanglement operation less effective for this blast furnace gas dataset.

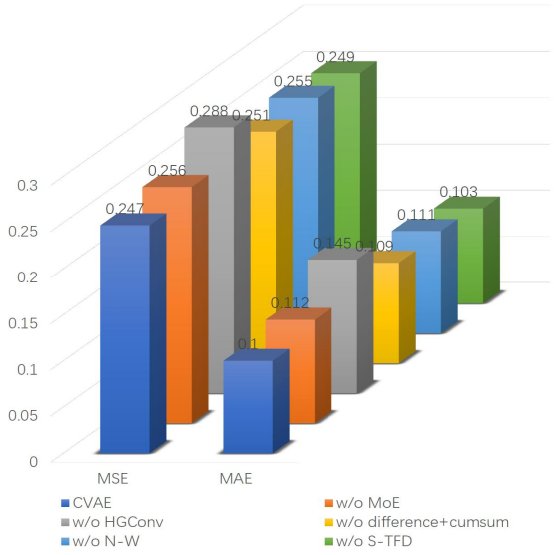

Fig. 10. Impact of different components.

G. Limitations

The proposed model enhances long-term forecasting accuracy by explicitly modeling the non-stationarity of time series. It adopts a variational autoencoder (VAE) framework and introduces stochastic perturbations to improve the robustness of the prediction process, thereby mitigating the effects of noise and distribution shifts to some extent. However, the model is designed for real-world industrial scenarios, where data collection is often constrained by sensor malfunctions, environmental disturbances, and systemic noise. Consequently, the model cannot consistently maintain optimal performance under all practical conditions. A more fundamental limitation lies in the inherent difficulty of long-term time series forecasting: as the prediction horizon extends, uncertainty and error accumulation increase significantly, leading to a decline in predictive accuracy and a potential rise in error rates within industrial decision-making or control processes. To improve practical applicability in industrial settings, future work may incorporate multi-source information, enhance denoising and anomaly detection mechanisms, and introduce causal or physical constraints to strengthen both the stability and interpretability of the model.

IV. Conclusion

This article proposes a novel framework for long-term time series prediction based on a conditional variational autoencoder. Specifically, the framework aims to enhance the performance of long-term time series prediction by modeling non-stationarity. The overall idea of the CVAE framework to model non-stationarity for improved long-term prediction performance are as follows: 1. Introducing randomness through noise to enable the model to better adapt to different input distributions. 2. Utilizing MoE encoding features to capture multi-periodic patterns in time series. 3. Carefully designing components in the decoder, such as seasonal-trend feature disentangler, adaptive Nadaraya-Watson regression, difference and cumsum operations, as well as hypergraph convolution, to model the non-stationarity of time series patterns. 4. Reintroducing non-stationary information from the original sequence into the predicted sequence through sequence stabilization, addressing the issue of distribution drift.

The proposed CVAE framework has been experimentally validated on multiple time-series datasets, achieving state-of-the-art results and demonstrating the effectiveness of the framework.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Baowen Xu conducted the research and drafted the manuscript. Song Gao and Hongxin Meng supervised the work, provided critical review, and refined the manuscript. All authors approved the final version of the paper.

References

[1] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22419–22430, 2021.

[2] Z. Li, S. Qi, Y. Li, and Z. Xu, "Revisiting long-term time series forecasting: An investigation on linear mapping," arXiv preprint arXiv:2305.10721, 2023.

[3] M. H. Liu, A. L. Zeng, M. X. Chen *et al.*, "SCINet: Time series modeling and forecasting with sample convolution and interaction," *Advances in Neural Information Processing Systems*, 2022.

[4] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17804–17815, 2020.

[5] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" arXiv preprint arXiv:2205.13504, 2022.

[6] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," arXiv preprint arXiv:2111.00396, 2021.

[7] Y. Liu, H. Zhang, C. Li, X. Huang, J. Wang, and M. Long, "Timer: Generative pre-trained transformers are large time series models," arXiv preprint arXiv:2402.02368, 2024.

[8] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," arXiv preprint arXiv:2210.02186, 2022.

[9] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "Itransformer: Inverted transformers are effective for time series forecasting," arXiv preprint arXiv:2310.06625, 2023.

[10] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. International Conference on Machine Learning*, 2022, pp. 27268–27286.

[11] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conference on Artificial Intelligence*, 2021, pp. 11106–11115.

[12] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1463–1470, 2021.

[13] P. Xu, J.-B. Hayet, and I. Karamouzas, "SocialVAE: Human trajectory prediction using timewise latents," arXiv preprint arXiv:2203.08207*,* 2022.

[14] B. Xu, X. Wang, S. Li, J. Li, and C. Liu, "Social-CVAE: Pedestrian trajectory prediction using conditional variational auto-encoder," in *Proc. International Conference on Neural Information Processing*, 2023, pp. 476–489.

[15] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Rethinking the stationarity in time series forecasting," arXiv preprint arXiv:2205.14415*,* 2022.

[16] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *Proc. International Conference on Learning Representations*, 2021.

[17] R. Ni, "Mixture-of-linear-experts for long-term time series forecasting," Ph.D. dissertation, Carnegie Mellon University, 2023.

[18] Z. Zhao, G. Shen, J. Zhou, J. Jin, and X. Kong, "Spatial-temporal hypergraph convolutional network for traffic forecasting," *PeerJ Computer Science*, vol. 9, e1450, 2023.

[19] [19] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in Neural Information Processing Systems*, vol. 19, 2006.

[20] T. M. Kodinariya, P. R. Makwana *et al*., "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[21] Y. Yang, J. Zhang, Z. Wang *et al.*, "Long term 5G network traffic forecasting via modeling non-stationarity with deep learning," *Communications Engineering*, vol. 2, no. 1, p. 33, 2023.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proc. International Conference on Learning Representations*, 2022.

[24] R. Chen, Y. Yang, W. Meng *et al.*, "LogTransfer: Cross-system log anomaly detection for software systems with transfer learning," in *Proc. IEEE International Symposium on Software Reliability Engineering*, 2020, pp. 37–47.

[25] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," arXiv preprint arXiv:2001.04451*,* 2020.

[26] C. Liu, J. Tan, J. Li, Y. Li, and H. Wang, "Temporal hypergraph attention network for silicon content prediction in blast furnace," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–13, 2022.