ImputeLLM: A Prompt-Free Large Language Model Framework for Robust Time-Series Imputation in HVAC Systems

Zehuan Hu[a], Yuan Gao[a,*], Gangwei Cai[b], Mingzhe Liu[c], Yan Ke[e,**], Yingjun Ruan[d]

[a] *The Center for Energy Systems Design (CESD), International Institute for Carbon-Neutral Energy Research (WPI-I2CNER), Kyushu University, Japan*
[b] *Department of Architecture, Graduate School of Engineering, The University of Tokyo, Tokyo, Japan*
[c] *Department of Multidisciplinary Engineering, Texas A&M University, College Station, TX, USA*
[d] *School of Mechanical Engineering, Tongji University, Shanghai 200092, China*
[e] *Department of Industrial Engineering, University of Roma Tor Vergata, Rome, Italy*

Abstract

Missing data poses a critical challenge in the modeling and control of HVAC systems, where reliable time-series information is essential for energy optimization and fault detection. While deep learning has advanced imputation accuracy, existing models often struggle with robustness under high missing rates or require extensive fine-tuning. This study introduces ImputeLLM, a novel imputation framework that integrates a frozen large language model (LLM) encoder with a Transformer-based decoder and an adaptive hybrid loss function. Without any fine-tuning or prompt engineering, the model efficiently encodes masked time-series data into semantic embeddings and reconstructs missing values with high accuracy. The framework is validated using real-world monitoring data from a central cooling plant in Qingdao, China, under both MCAR and MAR masking conditions. Compared to conventional methods, the proposed approach achieves up to 37.7% improvement in MAPE over linear interpolation and shows 16.5% gain over traditional MAE-based losses. Furthermore,

*Corresponding author
**Corresponding author2
Email addresses: yuangao1120@gmail.com (Yuan Gao), yanke@mail.zjgsu.edu.cn (Yan Ke)

it demonstrates strong generalization across varying missing rates and feature observability levels. These results highlight the potential of LLM-based architectures for practical deployment in energy systems with noisy or incomplete data.
Keywords: Large language model; Time Series Imputation; Missing Data Handling; HVAC Systems

---

1. Introduction

1.1. Background

Missing data is a pervasive issue across a wide range of applications, arising from human errors, data processing problems, or malfunctions in measurement devices [1]. Incomplete datasets present significant challenges for data analysis and model development. For example, in Heating, Ventilation, and Air Conditioning (HVAC) systems, historical data plays a crucial role in optimizing control strategies [2]. Similarly, energy management systems rely on comprehensive datasets to balance energy demand and supply [3]. However, the presence of missing values impairs these tasks, making effective imputation of missing data a critical research topic [4].

1.2. Current Status of Imputation

In general, three types of missing data mechanisms are recognized in the literature: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR) [5]. MCAR assumes that the probability of a value being missing is independent of both observed and unobserved variables. Owing to its simplicity, it serves as a common starting point for imputation studies. MAR refers to the scenario where the missingness is conditionally random given the observed variables, and it is more prevalent in real-world data. Most statistical imputation methods rely on the MAR assumption. In contrast, MNAR describes a situation where the probability of missingness depends on the unobserved value itself, which is more difficult to handle [6].

In practical applications, such as energy monitoring, HVAC system control, and indoor environmental sensing, MCAR is the most commonly encountered mechanism, while MNAR cases are relatively rare. MAR also frequently appears in certain special scenarios [7]. Therefore, this study focuses primarily on imputation methods tailored for MCAR and MAR conditions.

To address the problem of missing data, a wide range of strategies have been proposed. Existing approaches can be broadly categorized into three groups: statistical methods, machine learning models, and deep learning-based techniques [8]. Traditional statistical methods, such as mean/median imputation and linear interpolation, are simple and widely used, but they are often limited to single imputation and do not account for the uncertainty associated with the missing values [9]. Machine learning-based techniques, such as linear regression, decision trees, and support vector machines, offer improved flexibility but often fail to fully capture complex dependencies across multiple features or long-term temporal relationships [10].

In recent years, the development of deep learning has facilitated the emergence of more powerful imputation models [11, 12]. These include architectures such as multilayer perceptron (MLP), recurrent neural networks (RNN), convolutional neural networks (CNN), and graph neural networks (GNN), as well as attention-based models that have demonstrated strong performance in time-series tasks. More recently, generative adversarial networks (GAN) have also been explored for their potential in modeling complex missing patterns. These models are capable of learning intricate feature interactions and temporal dynamics from large-scale datasets. A summary of representative deep learning-based imputation methods is provided in Table 1.

Table 1: Literature review for probabilistic forecasting.

Ref.	Datasets	Methods	Missing Mechanism
[13]	Clinical datasets	RNN	MCAR
[14]	AQI; Health-care data; Human activity data	RNN	MCAR
[15]	Medical datasets	RNN	MCAR
[16]	ETT; Electricity; Weather	CNN	MCAR
[17]	Air quality(AQI); Traffic; Smart grids	GNN	MCAR; MAR
[18]	PeMS-BAY; METR-LA; AQI	GNN, Attention	MCAR; MAR
[19]	PhysioNet; AQI; Electricity; ETT	Attention	MCAR
[20]	AQI; Chlorine; Gas; Climate; Electricity; Temperature; MeteoSwiss; BAFU, JanataHack, Walmart M5	CNN, Attention	MCAR
[21]	Traffic flow; Solar energy; Smart meters; AQI	Attention	MCAR
[22]	AQI; METR-LA; PEMS-BAY	GNN, Attention	MCAR
[23]	Activity; PhysioNet; KDD	GAN, RNN	MCAR
[24]	AQI; PeMS-BAY; Electricity	GNN	MCAR
[25]	KDD; Guangzhou; PhysioNet	Diffusion, Attention	MCAR; MAR; NMAR
[26]	Electricity demand	Attention	MCAR; MAR; NMAR

1.3. Recent Advances in Large Language Models (LLMs)

Driven by advances in generative artificial intelligence, large language models (LLMs) have gained significant attention in both academic and industrial communities [27]. Many recent studies have explored the application potential of LLMs beyond natural language processing, including in domains such as healthcare, finance, and time-series forecasting [28]. Owing to the inherent sequential nature of language data, LLMs are naturally suited to capture temporal patterns, making them promising tools for forecasting tasks [29, 30].

Recent efforts to apply LLMs to time-series forecasting tasks can be broadly categorized into two approaches. The first category focuses on prompt-based reprogramming, where carefully designed prompts are embedded into the raw time-series input to guide the frozen LLMs to generate accurate predictions without altering the model parameters [31]. These approaches leverage the powerful pattern recognition and reasoning capabilities of LLMs, exploiting their intrinsic ability to understand temporal structures [32].

The second category involves explicitly fine-tuning LLMs on target time-series datasets. For instance, Jin et al. [33] proposed an innovative reprogramming framework that aligns frozen LLMs with time-series inputs using prompt-based conditioning, significantly enhancing performance in few-shot and zero-shot settings. Lin et al. [34] and Xia et al.[35] further demonstrated the benefits of domain-specific fine-tuning on temporal data. Chang et al. [36] introduced a two-stage fine-tuning strategy, in which the model is first aligned with time-aware representations, followed by fine-tuning a lightweight prediction head tailored for time-series forecasting.

These studies underscore the versatility and potential of LLMs for sequence modeling tasks. However, to the best of our knowledge, their application to the task of missing data imputation has not yet been systematically explored—a gap that this study aims to address.

1.4. Research contribution

Based on the above analysis, several research gaps can be identified:

1) Traditional imputation methods often suffer from low accuracy and fail to simultaneously capture both inter-feature correlations and long-term temporal dependencies.

2) Existing deep learning-based imputation models are usually validated only on standard public datasets and lack verification in more complex, real-world energy system scenarios.

3) While LLMs have demonstrated strong performance in time-series forecasting tasks, their use in imputation tasks has yet to be systematically studied.

To address these gaps, the main contributions of this study are as follows:

1) We propose a novel imputation model that integrates a frozen LLM encoder with a Transformer-based decoder. The proposed framework achieves high accuracy without the need for fine-tuning and operates with minimal memory overhead, making it suitable for practical deployment.

2) We introduce a new loss function specifically designed for the imputation task. This loss function improves imputation accuracy across a range of baseline models.

3) We validate the proposed model using real-world monitoring data collected from a central cooling plant in Qingdao, China. The results demonstrate the model's strong performance under practical operating conditions.

2. Methodology

2.1. Proposed Model: ImputeLLM

2.1.1. Motivation and Architecture Overview

Unlike traditional deep learning models that rely solely on numerical feature extraction, large language models (LLM) have demonstrated the ability to learn hierarchical contextual dependencies from vast sequences of data. These learned

representations can implicitly encode domain-relevant semantics, such as the causal relationships between temperature and energy demand, by capturing semantic-level temporal patterns [37]. By reprogramming numerical sequences into tokenized embeddings, the LLM encoder generates high-level abstractions of temporal dynamics, which subsequently guide the decoder in achieving fine-grained temporal alignment. This dual-stage design enhances both deterministic accuracy and probabilistic calibration.

Moreover, freezing the LLM encoder ensures model stability and prevents catastrophic forgetting while retaining its pre-trained knowledge of sequential dependencies [38]. This design enables generalizable sequence modeling without requiring costly fine-tuning or manually engineered prompts, making it particularly suitable for cross-domain and low-resource scenarios.

To address the challenges associated with prompt design and task-specific fine-tuning in LLM-based models, we propose a novel architecture named ImputeLLM, as illustrated in Fig. 1. ImputeLLM adopts a frozen pre-trained LLM as an encoder and employs a Transformer-based decoder to generate imputed values. Specifically, the encoder transforms the masked input time-series into tokenized representations via multi-head attention and projection layers, enabling seamless integration with the LLM. This avoids the need for handcrafted prompts or task-specific embeddings. The decoder then processes the encoder's output using a stack of Transformer layers and generates the final imputed data through an output projection head.

Importantly, our method maintains the generalization capability of the pre-trained LLM by entirely bypassing its fine-tuning and prompt engineering. This lightweight and modular design facilitates high-accuracy imputation while minimizing computational cost and memory usage.

2.1.2. Model Components and Mathematical Formulation

(1) Word Projection Layer

Standard LLMs contain large-scale vocabularies with many tokens irrelevant to time-series tasks. Directly utilizing such embeddings introduces

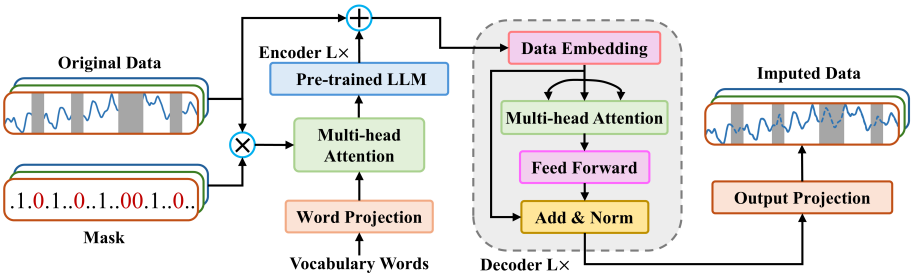


Figure 1: Proposed ImputeLLM architecture (The model consists of five main components: (1) Word projection layer; (2) Multi-head attention module; (3) Frozen pretrained LLM; (4) Transformer-based decoder).

significant memory and computation overhead. To address this, we introduce a learnable projection layer that maps the original pre-trained word embeddings into a reduced set of latent text prototypes, denoted as $\boldsymbol{P} \in \mathbb{R}^{N_p \times d}$, where $N_p \ll V$, which is original vocabulary size of LLM, and d is the embedding dimension.

Let $\boldsymbol{E} \in \mathbb{R}^{V \times d}$ be the original pre-trained word embedding matrix. We define a projection matrix $\boldsymbol{W}_p \in \mathbb{R}^{V \times N_p}$, which produces the compressed text prototype set:

$$\boldsymbol{P} = \boldsymbol{W}_p^\top \boldsymbol{E} \tag{1}$$

This operation significantly reduces memory usage and enables fast downstream attention computations.

(2) Multi-Head Attention for Sequence Reprogramming

Given an input time-series sequence $\boldsymbol{X} \in \mathbb{R}^{T \times D}$ and the corresponding binary mask $\boldsymbol{M} \in \mathbb{R}^{T \times D}$, we first apply element-wise masking to preserve only observed values:

$$\boldsymbol{M}_t^d = \begin{cases} 1, & \text{if } \boldsymbol{X}_t^d \text{ is obeserved} \\ 0, & \text{if } \boldsymbol{X}_t^d \text{ is missing} \end{cases} \tag{2}$$

$$\boldsymbol{X}_{\text{masked}} = \boldsymbol{X} \odot \boldsymbol{M} \tag{3}$$

We then compute the reprogrammed representation using multi-head attention between the masked sequence and the projected text prototypes $\boldsymbol{P}$. The standard multi-head attention mechanism is defined as:

$$\text{MHAtt}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, ..., \text{head}_h)\boldsymbol{W}^O \tag{4}$$

$$\text{head}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^L, \boldsymbol{V}\boldsymbol{W}_i^V), \text{ for } i = 1, ..., h \tag{5}$$

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{6}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ indicate the query, key, and value matrices respectively; $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v \in \mathbb{R}^{f \times d}$, represent the learnable weights. In this study, query is $\boldsymbol{X}_{\text{masked}}$, key and value are $\boldsymbol{P}$.

(3) Frozen Pre-trained LLM Encoder

The LLM encoder processes the tokenized representations without any parameter updates. Let $\mathcal{F}_{\text{LLM}}$ denote the frozen pre-trained LLM. The encoded sequence is given by:

$$\mathbf{H}_{\text{LLM}} = \mathcal{F}_{\text{LLM}}(\text{MHA}_{\text{tt}}(\mathbf{X}_{\text{masked}}, \mathbf{P}, \mathbf{P})) \tag{7}$$

This stage retains the rich pre-trained sequential knowledge of the LLM while integrating contextual information from the input time series.

(4) Transformer-based Decoder

To convert the LLM outputs back into numerical imputation predictions, we employ a Transformer decoder composed of standard modules: data embedding, multi-head attention, feed-forward network, and residual normalization layers.

The decoder input is initialized as:

$$\boldsymbol{Z}_0 = \text{Embed}(\boldsymbol{D}) \tag{8}$$

Each decoder layer updates the hidden state as follows:

$$\mathbf{Z}_\ell = \text{LayerNorm}\left(\text{FFN}\left(\text{MHAtt}(\mathbf{Z}_{\ell-1}, \mathbf{Z}_{\ell-1}, \mathbf{Z}_{\ell-1})\right) + \mathbf{Z}_{\ell-1}\right) \quad \text{for } \ell = 1, \ldots, L \tag{9}$$

Finally, the output projection layer maps the last decoder hidden state $\boldsymbol{Z}_L$ to the imputed time-series values:

$$\hat{\mathbf{X}} = \mathbf{Z}_L \mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}} \tag{10}$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{T \times D}$ represents the final imputed sequence.

In addition, inspired by residual network design [39], we do not feed the LLM output directly into the Transformer-based decoder. Instead, we first merge the LLM-generated values for the missing positions with the observed values from the original input sequence, forming a complete and informative representation. This approach ensures that the decoder leverages accurate observed information while benefiting from the LLM's semantically enriched imputation, rather than relying on zeros or simple placeholders for missing values. Then, the input of this layer $\boldsymbol{D}$ can be calculated as:

$$\boldsymbol{D} = \boldsymbol{M} \odot \boldsymbol{X} + (1 - \boldsymbol{M}) \odot \mathbf{H}_{\text{LLM}} \tag{11}$$

This architecture ensures modular design, efficient memory usage, and compatibility with a variety of missing patterns—all while leveraging the representational power of pre-trained LLMs without fine-tuning.

2.2. Adaptive Hybrid Loss Function

In conventional imputation model training, missing values are typically simulated by artificially masking complete data. The model is then trained to reconstruct only the artificially masked regions by minimizing a loss function defined over these segments. While effective in controlled settings, this approach assumes full observation of the training data—a condition often unmet in real-world applications, where missing values naturally exist in the raw input and vary in position and duration across samples.

Moreover, since prior loss functions ignore naturally missing values and observed (non-masked) values during training, they fail to leverage the full available information. This can lead to suboptimal performance, especially in scenarios with high or irregular missingness. Therefore, a loss design that can

adaptively learn from both observed and artificially masked data is necessary for robust model generalization.

To address these challenges, we propose an adaptive hybrid loss function tailored for imputation tasks, as illustrated in Fig. 2. The loss function consists of two components:

- Reconstruction Loss ($\mathcal{L}_{\text{rec}}$) — computed over the observed parts of the input sequence that were not artificially masked.
- Mask Loss ($\mathcal{L}_{\text{mask}}$) — computed over the artificially masked positions used to simulate missingness.

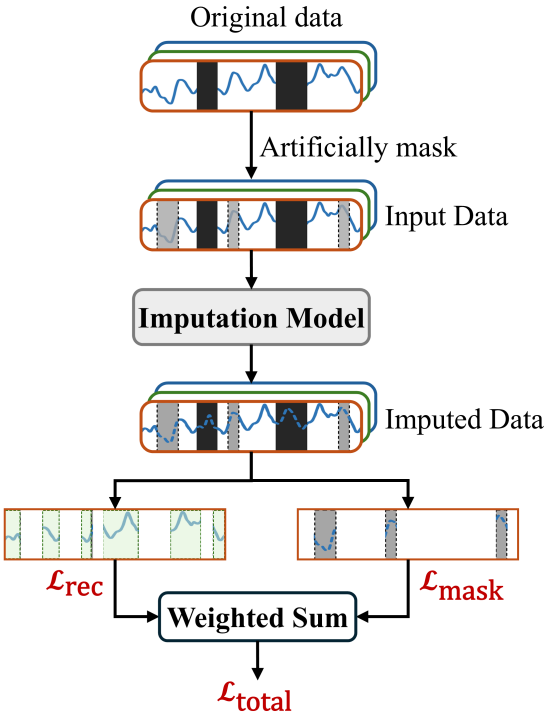


Figure 2: Proposed adaptive hybrid loss function.

During training, we apply an artificial mask to the input data regardless of whether the original sequence contains real missing values. After obtaining the imputed sequence from the model, we compute $\mathcal{L}_{\text{rec}}$ using the discrepancy

between the reconstructed values and the ground truth at originally observed, unmasked positions. Simultaneously, $\mathcal{L}_{\text{mask}}$ is computed using the positions corresponding to the artificial mask. The reconstruction loss and mask loss are defined as:

$$\mathcal{L}_{\text{rec}} = \frac{1}{|\Omega_{\text{rec}}|} \sum_{(t,d)\in\Omega_{\text{rec}}} (\hat{x}_{t,d} - x_{t,d})^2 \tag{12}$$

$$\mathcal{L}_{\text{mask}} = \frac{1}{|\Omega_{\text{mask}}|} \sum_{(t,d)\in\Omega_{\text{mask}}} (\hat{x}_{t,d} - x_{t,d})^2 \tag{13}$$

where $x_{t,d}$ and $\hat{x}_{t,d}$ denote the ground truth and imputed value at time t, dimension d; Ω_{rec} is the set of originally observed, unmasked positions; Ω_{mask} is the set of artificially masked positions.

To balance the contribution of these two losses—especially under different masking patterns and varying sequence lengths—we introduce a learnable weighting parameter λ. The final objective function is defined as:

$$\mathcal{L}_{\text{total}} = \lambda \cdot \mathcal{L}_{\text{rec}} + (1 - \lambda) \cdot \mathcal{L}_{\text{mask}} \tag{14}$$

To ensure stability during training, λ is initialized as 0.5 and optimized jointly with the model parameters via backpropagation. This dynamic weighting allows the model to adaptively emphasize either reconstruction or masking loss depending on the data conditions and training stage.

This hybrid loss framework enables effective learning even when training data contains natural missingness and eliminates the dependency on fully observed datasets, making the model more applicable in real-world deployment scenarios.

2.3. Benchmark Models

To comprehensively evaluate the effectiveness of our proposed model and loss function, we benchmark against a range of classical and deep learning-based imputation methods. Specifically, we include two conventional approaches—linear interpolation and XGBoost—as well as seven deep learning models that span diverse architectural paradigms.

The deep learning baselines consist of three fundamental architectures: LSTM, DLinear, and Transformer, representing recurrent, MLP-based, and self-attention-based models respectively. In addition, we evaluate three advanced Transformer variants—Informer, iTransformer, and PatchTST—which have been tailored for long-sequence forecasting. We also include TimesNet, a recent model that achieves state-of-the-art performance across various time-series tasks by leveraging temporal decomposition. Each benchmark model is briefly introduced below.

2.3.1. Conventional Methods

Linear Interpolation is one of the most widely used techniques for handling missing values due to its simplicity and computational efficiency [40]. It estimates missing entries by assuming linearity between adjacent observed values. However, it fails to exploit cross-feature dependencies and often performs poorly when consecutive missing segments occur.

XGBoost is a powerful tree-based machine learning method widely applied in time-series forecasting [41]. Its strengths lie in its robustness, ease of implementation, and minimal need for hyperparameter tuning. Despite being non-sequential in nature, its performance in many imputation tasks is competitive, making it a strong conventional baseline.

2.3.2. LSTM

The Long Short-Term Memory (LSTM) network is a RNN variant designed to capture long-term temporal dependencies by incorporating gating mechanisms [42]. It has been extensively used in time-series modeling and imputation tasks due to its ability to retain information over long sequences.

2.3.3. DLinear

DLinear is a lightweight deep learning model based on multi-layer perceptrons (MLP), which directly maps historical inputs to future values without relying on complex sequence modeling [43]. Its architecture emphasizes simplic-

ity and efficiency, and it has been shown to outperform many Transformer-based models on short-horizon tasks.

2.3.4. Transformer

Transformer is a foundational architecture based on the self-attention mechanism, originally developed for natural language processing but later widely adopted in time-series forecasting [44]. It models global temporal dependencies and scales well with large datasets, though it can be memory-intensive and prone to overfitting in small-data regimes.

2.3.5. Informer

Informer improves upon the standard Transformer by introducing a ProbSparse attention mechanism, which significantly reduces computational complexity for long sequences [45]. Additionally, it adopts a generative-style decoder to better handle multi-step forecasting, making it suitable for imputation over extended gaps.

2.3.6. iTransformer

iTransformer is a recent variant of Transformer that introduces instance-level normalization and self-attention adaptations to enhance model stability and performance in imputation and forecasting scenarios [46]. It aims to preserve instance-specific temporal patterns while reducing overfitting.

2.3.7. PatchTST

PatchTST is a Transformer-based model that applies patch embedding to time-series inputs, akin to the vision transformer architecture [47]. By segmenting inputs into temporal patches, it captures local and global patterns more effectively and has demonstrated strong results in both short-term and long-term forecasting benchmarks.

2.3.8. TimesNet

TimesNet is a state-of-the-art time-series model that incorporates multi-scale decomposition of temporal signals [16]. By disentangling and modeling seasonal,

trend, and residual components, it achieves high accuracy across diverse forecasting and imputation tasks. It has been shown to consistently outperform traditional Transformer-based models in benchmark evaluations.

2.4. Model setup

All benchmark models are configured using the optimal hyperparameter settings as reported in their original studies. The choice of parameters balances model complexity and computational feasibility, ensuring efficient training and high-quality imputation performance within available hardware constraints. Furthermore, model-specific configurations are adjusted to align with the characteristics of the time-series data used in this study, including input dimensionality and sampling frequency, to better match the nature of the forecasting task. Due to limited equipment, we chose ChatGPT2 [48], which has shown good performance in previous studies, as the pre-trained LLM model in this experiment.

Table 2 summarizes the hyperparameters used for each model, including the number of layers, hidden dimensions, number of attention heads, and other key architectural settings.

Table 2: Configuration of hyper-parameters for benchmarks and ImputeLLM (d_{model}: Dimension of model; d_{ff}: Dimension of feed-forward layer; l_{encoder}: Number of encoder layers; l_{decoder}: Number of decoder layers; f_{attn}: Attention factor).

Model	**Model hyper-parameter**				
	d_{model}	d_{ff}	l_{encoder}	l_{decoder}	f_{attn}
LSTM	256		2		
Transformer	256	1024	2	1	3
Informer	512	2048	4	1	5
iTransformer	512	512	3	1	3
PatchTST	256	256	3	1	3
TimesNet	32	32	4	1	
ImputeLLM	32	64	6	2	3

To ensure a fair comparison, all models are trained under a unified training protocol. Specifically, the Adam optimizer is used for all models with an initial learning rate of 0.001. The learning rate is decayed by a factor of 0.95 at the end of each epoch to promote stable convergence. Each model is trained for 20 epochs. To prevent overfitting, we adopt an early stopping strategy. This strategy terminates the training process when the accuracy on the validation dataset ceases to improve (or begins to degrade) after a certain number of iterations. Such an approach accelerates training and improves hyperparameter tuning efficiency. The "patience" parameter in early stopping defines the maximum number of epochs the model is allowed to continue training without improvement. In this study, the patience is set to 6.

All experiments are conducted in a consistent computational environment to guarantee reproducibility. Models are implemented in PyTorch and trained on a single GPU node equipped with an NVIDIA GeForce RTX 5090 with 32 GB of memory.

3. Case study

3.1. Case Study Description

The case study is conducted on a central cooling plant located in Qingdao, China. The plant mainly serves the cooling demand of office buildings, production workshops, and process air-conditioning in the area. The facility operates a secondary-pump variable-flow centralized air-conditioning system, as illustrated in Fig. 3. The plant provides year-round cooling, where the process cooling system serves production lines and workshop equipment, while the air-conditioning system supports comfort cooling and constant-temperature–constant-humidity (CTCH) demands.

The system is configured in a one-to-one series connection between each chiller unit and its corresponding pump, enabling redundancy and flexible operation. The major equipment and operational parameters of the chiller plant are summarized in Table 3. The dataset used in this study covers approximately 17

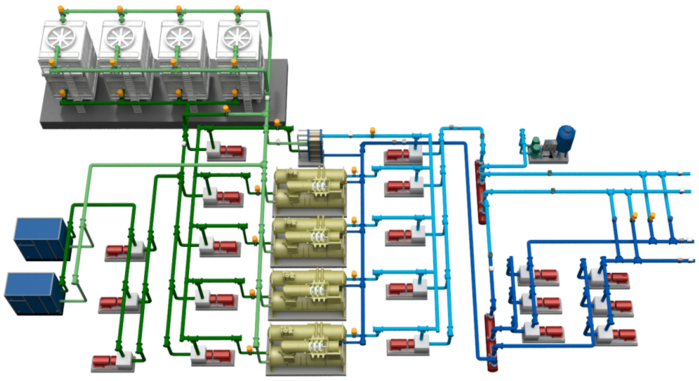


Figure 3: Schematic diagram of the secondary-pump variable-flow centralized cooling system in the Qingdao case study.

months of operation, spanning from March 8, 2023 to August 19, 2024, with a sampling resolution of 15 minutes.

Table 3: Main equipment and specifications of the central cooling plant (VF: Variable Frequency; fixed: fix frequency control; AC: Air-conditioning loop; Process: Industrial cooling loop).

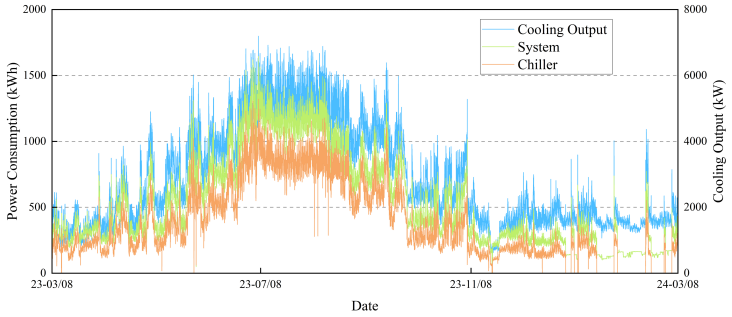
Category	Equipment	Quantity	Power (kW)	Flow Rate (m^3/h)	Head (m)	Speed (rpm)	Notes
Chiller Unit	CVHF06580 (Centrifugal Chiller) - Cooling capacity: 650 RT, COP: 6.47 - Chilled water temp.: 6–12°C - Condenser water temp.: 32–37°C	4	353	—	—	—	VF (fixed)
Pumps	Primary chilled	5	30	350	20	1450	VF (fixed)
	Condenser	5	45	490	22	1450	VF (fixed)
	Secondary chilled (AC)	3	15	175	—	—	VF (fixed)
	Secondary chilled (Process)	3	30	350	—	—	VF (fixed)
Cooling Tower	Cooling tower	4	22	500			VF (PID)

3.2. Feature Analysis

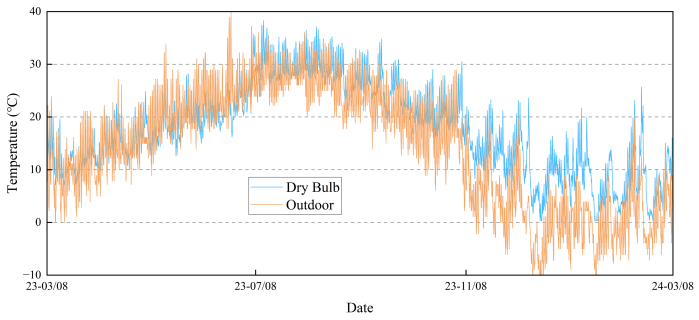
As the monitoring system in this project records a large number of variables, we selected only a subset of key features to validate the imputation performance of the proposed model. In practical HVAC system control scenarios, the most critical features for optimization are typically cooling demand, total energy consumption, and system efficiency.

Accordingly, we designate the following four features as the imputation target variables: Total System Power Consumption, Chiller Power Consumption, System efficient and Total Cooling Output.

Fig. 4 illustrates the time-series patterns of the selected target features over a representative period. To support imputation, additional input features were selected based on their strong physical and operational correlation with the target variables. A detailed list of these input features is provided in Table 4.



(a) HVAC system performance metrics



(b) Ambient temperature data

Figure 4: Time-series trends of selected monitoring data used for imputation.

To ensure stable gradient descent during training and to mitigate issues such as vanishing gradients, all input features were standardized using z-score normalization prior to being fed into the model [49]. Specifically, each feature was normalized by subtracting its mean and dividing by its standard deviation, computed from the training dataset. During evaluation, all model outputs were

Table 4: Input features used for imputation, grouped by category.

Category	Features
Environment	Measured ambient air temperature (°C)
	Dew point temperature (°C)
	Dry bulb air temperature (°C)
	Wet bulb temperature (°C)
System Energy	Total energy consumption of the system (kWh)
and Efficiency	Total cooling output of the system (kW)
	System coefficient of performance
	Overall system energy efficiency
Chiller Subsystem	Power consumption of all chillers (kWh)
	Coefficient of performance of chillers
Cooling Tower Subsystem	Power used by cooling tower fans (kWh)
Cooling Water Loop	Power for condenser water pumps (kWh)
	Return water temperature of condenser loop (°C)
	Supply water temperature of condenser loop (°C)
Chilled Water Loop	Combined flow of chilled water (m^3/h)
	Flow rate in industrial loop (m^3/h)
	Flow rate in air-conditioning loop (m^3/h)
	Power of primary chilled water pumps (kWh)
	Power of secondary pumps for process cooling (kWh)
	Pump efficiency
	Chilled water supply temperature (°C)
	Chilled water return temperature (°C)
	Bypass temperature (°C)
	Supply pressure of chilled water loop (MPa)
	Pressure difference in chilled water loop (MPa)

re-scaled through inverse normalization to restore the original units and enable fair comparison across models and metrics.

3.3. Experimental Setup

3.3.1. Comparison of Loss Functions

To evaluate the effectiveness of the proposed adaptive hybrid loss function, we compare it against four baseline loss functions commonly used in previous imputation studies:

(a) MSE on missing positions only:

This loss function computes the Mean Squared Error (MSE) exclusively over the artificially masked (i.e., missing) positions. It focuses solely on the imputation accuracy of missing values, ignoring the reconstruction of observed segments. MSE can be computed as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{15}$$

where x_i and $\hat{x_i}$ represent the measured value and predicted value respectively; N is the length of sequence.

(b) MAE on missing positions only:

Similar to (a), this variant calculates the Mean Absolute Error (MAE) only over the artificially masked parts. MAE is less sensitive to large outliers than MSE, which may benefit models under noisy data conditions. MAE can be computed as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x}_i| \tag{16}$$

(c) MAE on the entire sequence:

This loss function evaluates MAE across the entire input sequence, including both masked and unmasked positions. While it encourages smoother reconstruction, it may dilute the model's focus on missing-value estimation.

(d) Fixed-weight hybrid loss:

This loss combines both observed and missing positions using a fixed scalar weight, typically chosen manually. Although this approach can partially balance both reconstruction and imputation accuracy, the fixed weight does not adapt to varying missing patterns or sequence lengths, potentially leading to instability during training.

These baselines provide a comprehensive context for assessing the performance improvements introduced by our adaptive hybrid loss, which dynamically balances the two components using a learnable weighting mechanism (as described in Section 2.2).

3.3.2. Masking Strategies

To simulate realistic conditions under which missing data may occur in building energy systems, we evaluate the imputation performance under two representative masking strategies:

(a) MCAR (Missing Completely At Random):

This strategy assumes that missing values occur independently across all features and time steps. It is used to model scenarios where data loss is caused by random sensor faults or unintentional data corruption across the entire monitoring system. All variables have an equal probability of being masked, and we generate the mask for each data sample by drawing from a Bernoulli random variable with a fixed parameter.

(b) MAR (Missing At Random):

This strategy reflects more practical conditions, where certain variables—such as weather data (e.g., outdoor temperature) and energy consumption readings—are typically more reliable or easier to estimate through alternative measurements. In this case, missing values are introduced only to a subset of variables, while others remain fully observed. This pattern better mimics real-world sensor networks where certain channels (e.g., indoor

temperatures or flow rates) may suffer from sporadic dropout, while others (e.g., weather stations or smart meters) are consistently available. For MAR, we first sample a subset of features (columns in X) that will not contain missing values and then we use a logistic model with these non-missing columns as input to determine the missing values of the remaining columns and we employ line search of the bias term to get the desired proportion of missing values.

These masking patterns are used during both training and evaluation to ensure that the models are tested under diverse and application-relevant missing data scenarios.

3.4. Evaluation Metrics

To comprehensively assess the imputation performance across multiple target variables, we employ three evaluation metrics: MAE, Mean Absolute Percentage Error (MAPE), and the Coefficient of Determination (R^2). MAE provides a straightforward, dimensioned measure of the average absolute deviation between the imputed values and the ground truth. To capture the relative performance across variables with different magnitudes, MAPE is used to reflect the proportional imputation error. Additionally, R^2 is included to quantify how well the imputed sequence preserves the overall trend and variance of the original data. MAPE and R^2 metrics are computed as follows

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{x_i - \hat{x}_i}{x_i + \epsilon} \right| \tag{17}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{\sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{18}$$

where x_i, $\hat{x_i}$, and $\bar{x}$ represent the measured value, predicted value, and mean of measured value, respectively; N is the length of sequence; ϵ is a small constant (e.g., 10^{-5}) added to the denominator in MAPE to prevent division by zero.

4. Results and discussions

4.1. Comparison of Loss Functions

Firstly, we investigate how the input sequence length affects the imputation performance of different models. We evaluate sequence lengths of 12, 24, 48, 96, 192, 288, 480, and 672 time steps, as illustrated in Fig. 5. The results show that, except for Informer, iTransformer, and PatchTST, most models improve as the sequence length increases, reaching optimal performance around 48 steps (12 hours), after which the accuracy declines. As the sequence length exceeds 288 steps (3 days), the degradation slows down. Unlike conventional HVAC systems in residential or office buildings where daily periodicity is apparent, this central cooling plant also serves large industrial zones, which exhibit less pronounced temporal cycles. As a result, shorter sequences fail to provide sufficient context, while excessively long sequences introduce noise from distant time points. Based on this observation, we adopt an input sequence length of 48 steps for all subsequent experiments.

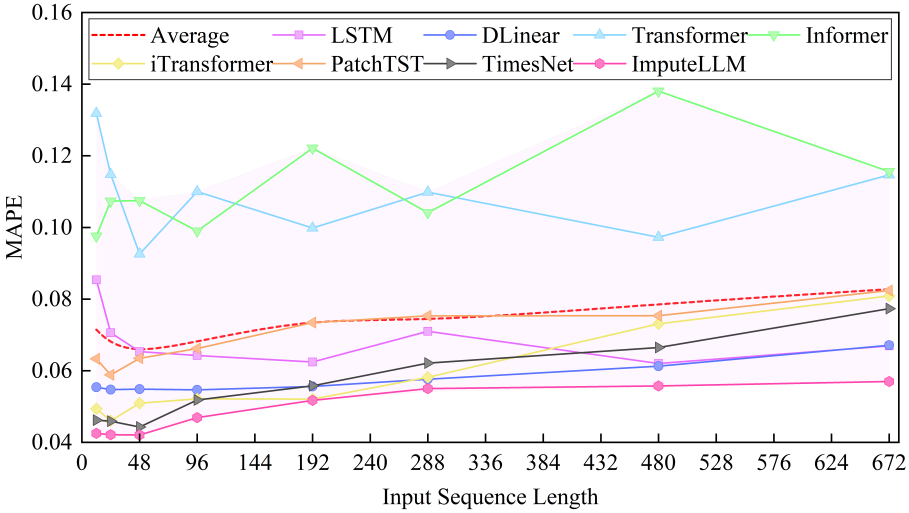
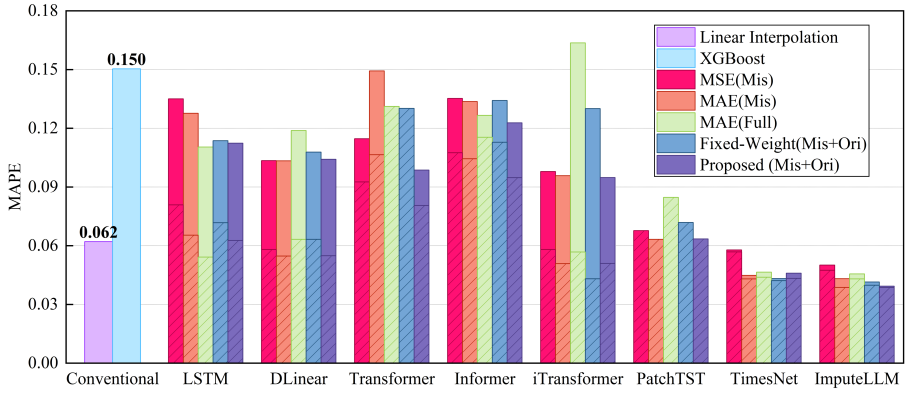


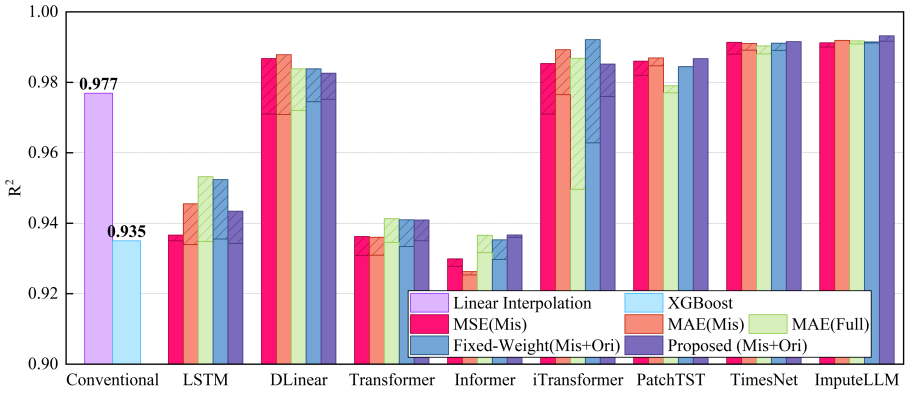
Figure 5: Performance of different models under varying input sequence length, evaluated by MAPE (mask rate = 0.3; missing mechanisms: MCAR).

The performance of each model under different loss functions is presented in

Fig. 6, evaluated using two unitless metrics: MAPE and R^2. MAPE quantifies the relative error of the imputed values, while R^2 reflects the consistency of the imputed sequence with the overall trend of the original data.



(a) MAPE



(b) R^2

Figure 6: Comparison of different loss functions across models using MAPE and R^2 metrics (shaded (diagonally filled) bars represent experiments where missing input values were first filled using linear interpolation before being passed into the model; all results are averaged over 10 random seeds with mask rates in {0.1, 0.2, 0.3, 0.4, 0.5}; missing mechanisms: MCAR).

The results show that across all model types, the proposed adaptive hybrid loss consistently achieves superior performance compared to traditional loss functions, including both single-term losses (MSE or MAE on masked positions

only) and fixed-weight hybrid losses. Among all methods, XGBoost yields the poorest performance, indicating that conventional machine learning algorithms are ill-suited for complex imputation tasks without further adaptation.

A key observation is that when missing inputs are filled with zeros, most deep learning models—even strong architectures such as Transformer, Informer, and DLinear—perform worse than simple linear interpolation, with the notable exceptions of TimesNet and the proposed ImputeLLM model. This highlights a critical limitation: standard time-series forecasting models, when trained with conventional loss functions, are not directly transferable to imputation tasks without special treatment of missing inputs.

On the other hand, when input sequences are preprocessed using linear interpolation to replace missing values before feeding into the model, all architectures show substantial improvements. With the exception of Transformer and Informer, every model outperforms the linear interpolation baseline under this setting. Notably, LSTM and iTransformer achieve average MAPE reductions of 44.2% and 53.2%, respectively, when trained with our proposed loss formulation. These findings indicate that even standard forecasting models can be readily adapted for imputation tasks using our proposed approach—without requiring any architectural modification or fine-tuning.

4.1.1. Performance Across Different Target Variables

Due to the poor performance of XGBoost observed in previous experiments, we focus our analysis in this section on the deep learning models and the commonly used linear interpolation baseline. When the missing rate is set to 0.3, the imputation results for the four selected target variables are summarized in Table 5. The proposed ImputeLLM model achieves the best performance across all targets, outperforming even the state-of-the-art TimesNet. In contrast, models such as DLinear, Informer, and iTransformer exhibit the weakest performance.

For the three relatively stable variables—total energy consumption, chiller power, and system energy efficiency—all models perform reasonably well, with average MAPE values of 0.052, 0.048, and 0.052, respectively. However, for cool-

Table 5: Imputation performance of different models on each target variable (mask rate = 0.3; missing mechanisms: MCAR; LI: Linear Interpolation).

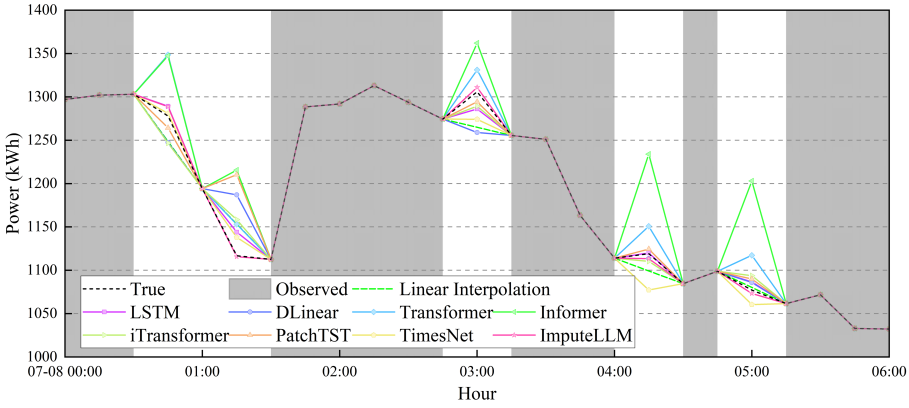
Target variable	Metrics	Model								
		LI	LSTM	DLinear	Transformer	Informer	iTransformer	PatchTST	TimesNet	**ImputeLLM**
Total Power (System)	MAE	22	21	23	32	39	21	27	17	**13**
	MAPE	0.051	0.043	0.054	0.069	0.084	0.047	0.057	0.034	**0.026**
	R^2	0.996	0.995	0.996	0.988	0.984	0.997	0.995	**0.999**	0.998
Total Power (Chiller)	MAE	17	24	18	27	33	16	22	14	**11**
	MAPE	0.042	0.057	0.044	0.065	0.067	0.040	0.051	0.035	**0.028**
	R^2	0.996	0.985	0.995	0.983	0.979	0.996	0.994	0.998	**0.998**
Energy Efficient (System)	MAE	0.058	0.114	0.063	0.123	0.126	0.061	0.065	0.054	**0.047**
	MAPE	0.054	0.048	0.058	0.048	0.050	0.058	0.063	0.048	**0.040**
	R^2	0.970	0.801	0.966	0.774	0.766	0.968	0.964	0.965	**0.980**
Cooling Output	MAE	117	**73**	119	121	167	109	137	92	74
	MAPE	0.058	0.113	0.064	0.188	0.229	0.059	0.083	0.060	**0.053**
	R^2	0.995	0.998	0.995	0.995	0.991	0.996	0.994	**0.998**	0.997

ing output, which is characterized by larger fluctuations and higher variance, all models show significantly reduced imputation accuracy, with an average MAPE of 0.101 across all methods.

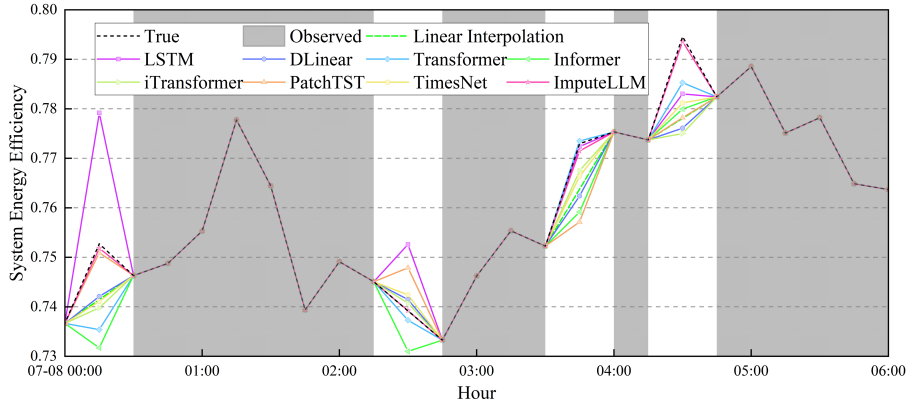
To better understand these results, Fig. 7 presents selected time-series imputation examples for three key targets. The comparison reveals that linear interpolation provides robust performance when the missing segments are short and isolated. However, as the length of the missing segments increases and values fluctuate more drastically, its accuracy declines sharply. In contrast, deep learning models are able to maintain relatively consistent performance under such conditions. Notably, the ImputeLLM model consistently achieves the best accuracy regardless of the location, length, or dynamics of the missing segments, demonstrating both flexibility and generalization capability.

4.2. Model Performance under Varying Mask Rates

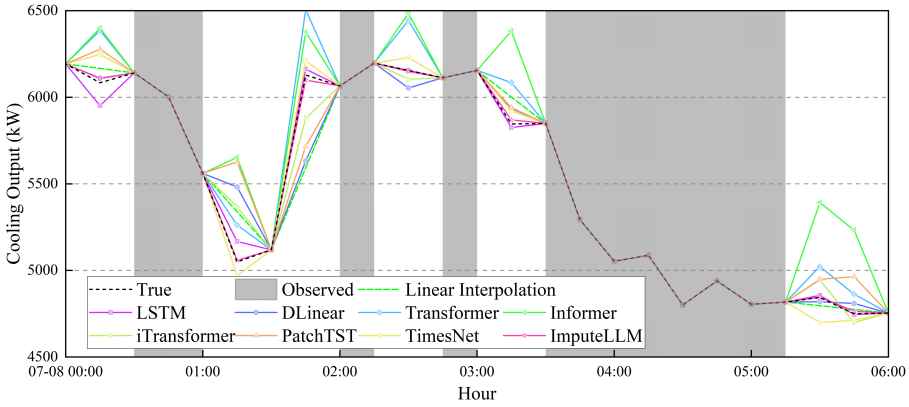
As the mask rate increases, the imputation performance of all models deteriorates, as shown in Fig. 8. Among the models, Transformer and Informer exhibit the most rapid decline in both MAPE and R^2, indicating limited robustness to high levels of missing data. In contrast, the proposed ImputeLLM consistently achieves the lowest error and the highest trend fidelity across all



(a) Total energy consumption of the system



(b) System energy efficient



(c) Total cooling output of the system

Figure 7: Detailed imputation results for three representative target variables (mask rate = 0.3; missing mechanisms: MCAR).

tested mask rates, demonstrating superior stability and generalization capability under challenging data conditions.

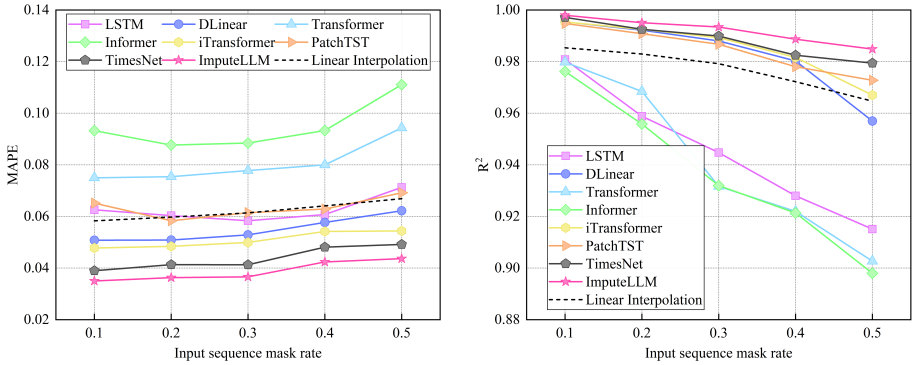


Figure 8: Performance of different models under varying input mask rates, evaluated by MAPE and R^2 (missing mechanisms: MCAR).

The training efficiency of all deep learning models is summarized in Fig. 9, which presents a trade-off between training time, memory consumption, and imputation accuracy. The ImputeLLM model achieves the best overall performance on all target variables, while maintaining a training time less than twice that of other models and a memory usage below three times that of SOTA baselines such as PatchTST or TimesNet. This highlights that the proposed architecture balances performance and efficiency, making it a practical solution for real-world deployment even in resource-constrained environments.

4.3. Zero-shot Imputation Performance

4.3.1. MCAR Setting

To evaluate model generalization under distribution shifts in missing patterns, we assess all models in a zero-shot setting, where the models are trained using a fixed mask rate of 0.3 and tested across a range of mask rates from 0.1 to 0.7. The results under the MCAR setting are shown in Fig. 10.

Overall, all models exhibit relatively stable performance when tested at mask rates lower than the training value (i.e., < 0.3). However, when the mask rate

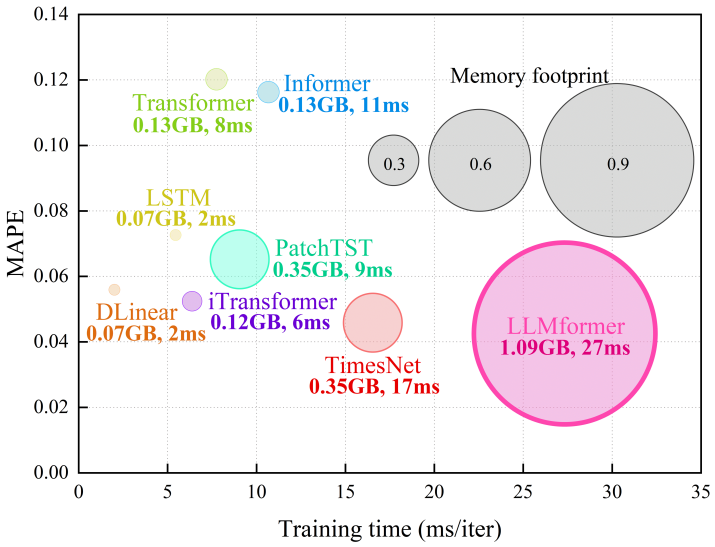


Figure 9: Training efficiency of models in terms of time per iteration and memory usage (mask rate = 0.3; missing mechanisms: MCAR).

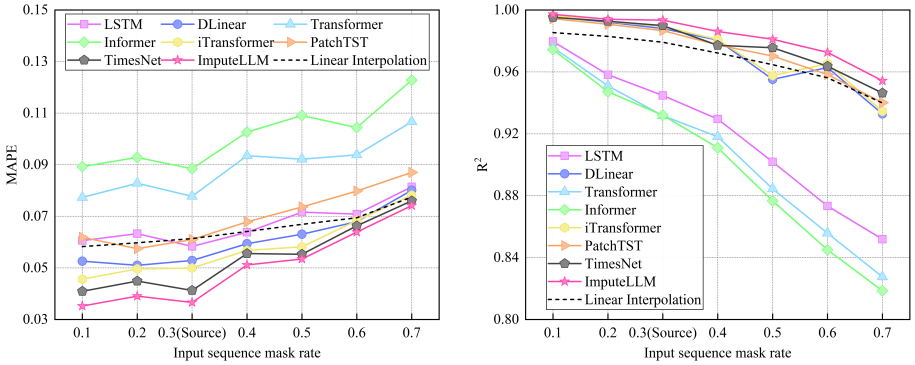


Figure 10: Zero-shot imputation performance of all models under MCAR masking, evaluated on input sequences with varying mask rates (all models are trained at a fixed mask rate of 0.3).

exceeds 0.3, a notable decline in imputation accuracy is observed across all models, with degradation trends roughly mirroring that of the linear interpolation baseline. This highlights the sensitivity of imputation tasks to the quantity of observable input information.

Among all tested models, ImputeLLM consistently outperforms others at all mask rates, showing superior robustness and generalization. Notably, when the mask rate exceeds 0.6, the imputation performance of most models—including LSTM, DLinear, iTransformer, and TimesNet—approaches that of linear interpolation. Only Transformer and Informer perform significantly worse. This observation suggests that when the proportion of missing values is too high, deep models lose access to sufficient contextual input to perform reliable imputation.

It is also worth emphasizing that the mask rate used during training appears to have limited impact on the model's ability to generalize across different masking conditions. This robustness arises from our training strategy, where artificial masks are applied across the entire dataset rather than on a fixed ratio per sequence. As a result, the actual input mask rate varies naturally during training, enabling the model to learn a wide range of missing patterns and improving its generalization in zero-shot settings.

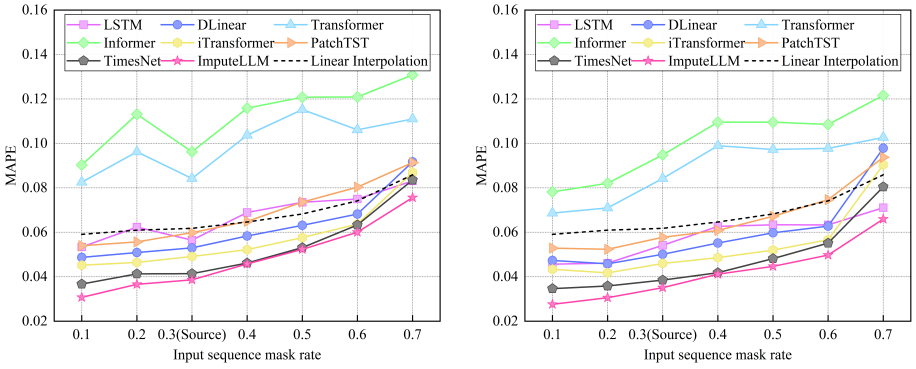
4.3.2. MAR Setting

In this section, we evaluate the robustness of all models under the MAR condition, where different proportions of non-target variables are assumed to be always observable. Specifically, we vary the fraction of fully observed non-target features across five settings: 0% (equivalent to MCAR), 25%, 50%, 75%, and 100%, while applying random masking to the remaining features according to input mask rates between 0.1 and 0.7.

All models are trained under the MCAR setting with a mask rate of 0.3 and then directly tested (zero-shot) under these MAR configurations. The average performance across different levels of partial observability is summarized in Table 6, and Fig. 11 presents the MAPE trends for the two most representative cases: (a) 50% and (b) 100% full observability among non-target features.

Table 6: Zero-shot imputation performance of all models under MAR masking (AOR: Always-Observed Ratio; all models are trained at a fixed mask rate of 0.3 under MCAR masking; all results are averaged with mask rates in {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7}).

AOR	Metrics	Model								
		LSTM	DLinear	Transformer	Informer	iTransformer	PatchTST	TimesNet	ImputeLLM	LI
0%	MAPE	0.077	0.064	0.111	0.125	0.061	0.071	0.057	0.055	0.068
	R^2	0.920	0.969	0.905	0.900	0.972	0.969	0.975	0.981	0.966
25%	MAPE	0.069	0.064	0.115	0.115	0.059	0.072	0.045	0.050	
	R^2	0.921	0.956	0.893	0.892	0.957	0.953	0.970	0.964	
50%	MAPE	0.068	0.062	0.100	0.113	0.057	0.069	0.052	0.049	
	R^2	0.906	0.965	0.884	0.880	0.967	0.965	0.987	0.982	
75%	MAPE	0.063	0.062	0.095	0.117	0.054	0.065	0.048	0.047	
	R^2	0.927	0.952	0.909	0.903	0.955	0.948	0.970	0.959	
100%	MAPE	0.058	0.060	0.089	0.101	0.054	0.066	0.044	0.046	
	R^2	0.920	0.973	0.905	0.899	0.977	0.972	0.977	0.988	



(a) 50% of non-target features are fully observed

(b) 100% of non-target features are fully observed

Figure 11: Zero-shot imputation performance under MAR masking with partial feature observability (all models are trained at a fixed mask rate of 0.3 under MCAR masking).

The results clearly indicate that increasing the number of always-observed features significantly enhances imputation performance. Across all models and mask rates, the average MAPE decreases from 0.078 at 0% observability to 0.065 at 100% observability. This improvement is especially evident under high mask rates (e.g., > 0.5), where contextual information from correlated observed variables plays a critical role in reconstructing missing values in the target features.

These findings highlight the potential benefits of leveraging strong feature correlations and suggest that domain knowledge (e.g., selecting reliable sensor channels) can be used to guide feature prioritization in real-world deployment scenarios.

4.4. Comparison with Existing Imputation Methods

Finally, we compare the proposed ImputeLLM model with several state-of-the-art imputation methods on three widely-used public datasets, as summarized in Table 7. The AQI dataset [50] is a benchmark dataset for urban computing, consisting of air quality readings from 437 monitoring stations across 43 cities in China. The ECL dataset [51] is a publicly released electricity consumption dataset from the University of California, Irvine. The PEMS-BA dataset [52] is a well-known spatiotemporal traffic dataset. Experimental results show that ImputeLLM achieves the best performance on the AQI and PEMS-BA datasets, and the second-best result on the ECL dataset. These results demonstrate that our model generalizes well across diverse datasets and application scenarios.

5. Conclusion

In real-world time series data, missing values frequently occur due to sensor failure, communication loss, or data corruption. Traditional imputation approaches, such as linear interpolation or classical deep learning methods, often struggle to balance accuracy, robustness, and generalization, especially under high missing rates or partial observability. To address these challenges, we propose ImputeLLM, a novel time series imputation framework that leverages a

Table 7: The MAE results of different imputation methods on different datasets (mask rate = 0.25).

Models	Datasets		
	AQI	ECL	PEMS-BA
Transformer[44]	18.13	0.84	0.74
BRITS[14]	20.21	0.93	1.47
GRIN[53]	14.73	0.56	0.67
SAITS[19]	21.33	0.77	1.40
TDM[54]	20.76	0.54	2.28
HSPGNN[24]	12.85	**0.10**	0.79
ImputeLLM	**12.80**	0.52	**0.66**

frozen LLM encoder, a lightweight Transformer decoder, and a unified probabilistic output layer. Our approach avoids prompt engineering and fine-tuning, ensuring both flexibility and efficiency. Based on extensive experiments, we draw the following conclusions:

1) Applying simple linear interpolation to missing inputs significantly improves model performance, enabling even standard forecasting models to outperform traditional interpolation methods. Compared with zero-filling, linear interpolation filling reduces the average MAPE across all models by 24.0%, highlighting its importance for pre-processing in real-world deployments.

2) The adaptive hybrid loss enhances training stability and imputation accuracy by dynamically balancing observed and missing data. Compared with the conventional MAE loss, it lowers the average MAPE across all models by 16.5%, improving generalization across various missing scenarios.

3) ImputeLLM maintains strong performance under high mask rates, showing higher resilience than existing deep learning models. Across all mask

rates, its imputation accuracy improves by 37.7% compared to linear interpolation, confirming its robustness under data scarcity.

4) The model benefits significantly when key input features remain observable, especially under MAR conditions. This finding supports the design of sensor systems that prioritize critical variables to improve imputation reliability.

Despite its promising performance, ImputeLLM has several limitations. First, although the model avoids fine-tuning by using a frozen LLM encoder, this design may constrain its adaptability to certain domain-specific semantics not covered during pretraining. Second, the current implementation relies on GPT-2, which limits model capacity due to memory constraints; future studies could explore more compact or domain-adapted LLMs such as DistilGPT or Time-LLM variants. Third, while our model was validated on a real-world HVAC dataset, broader generalization requires further testing across different domains (e.g., healthcare, finance). Finally, future work may consider incorporating causal inference mechanisms or uncertainty quantification modules to enhance interpretability and risk-aware decision-making in downstream control tasks.

Acknowledgment

This work was supported by project No. 23KJ0766 funded by the Japan Society for The Promotion of Science. All the code of models are open sourced in ImputeLLM GitHub Repository.

References

[1] K. Seu, M.-S. Kang, H. Lee, An intelligent missing data imputation techniques: A review, JOIV: International Journal on Informatics Visualization 6 (1-2) (2022) 278–283.

[2] Z. Hu, Y. Gao, L. Sun, M. Mae, T. Imaizumi, Improved robust model predictive control for residential building air conditioning and photovoltaic

power generation with battery energy storage system under weather forecast uncertainty, Applied Energy 371 (2024) 123652.

[3] Y. Gao, M. Liu, Z. Hu, S. Yamate, J. Otomo, W.-A. Chen, Z. O'Neill, Quantitative analysis of energy justice in demand response: Insights from real residential data in texas, usa, Renewable Energy 242 (2025) 122477.

[4] Y. Sun, J. Li, Y. Xu, T. Zhang, X. Wang, Deep learning versus conventional methods for missing data imputation: A review and comparative study, Expert Systems with Applications 227 (2023) 120201.

[5] R. J. Little, D. B. Rubin, Statistical analysis with missing data, John Wiley & Sons, 2019.

[6] R. J. Little, Missing data assumptions, Annual Review of Statistics and Its Application 8 (1) (2021) 89–107.

[7] P. Wang, D. Chen, X. Meng, R. Liang, Research on collaborative reconstruction method of missing data and abnormal data in air handling units (ahu) system, Journal of Building Engineering 98 (2024) 111414.

[8] S. Jäger, A. Allhorn, F. Bießmann, A benchmark for data imputation methods, Frontiers in big Data 4 (2021) 693674.

[9] D. Adhikari, W. Jiang, J. Zhan, Z. He, D. B. Rawat, U. Aickelin, H. A. Khorshidi, A comprehensive survey on imputation of missing data in internet of things, ACM Computing Surveys 55 (7) (2022) 1–38.

[10] A. Alamoodi, B. Zaidan, A. Zaidan, O. Albahri, J. Chen, M. Chyad, S. Garfan, A. Aleesa, Machine learning-based imputation soft computing approach for large missing scale and non-reference data imputation, Chaos, Solitons & Fractals 151 (2021) 111236.

[11] J. Wang, W. Du, Y. Yang, L. Qian, W. Cao, K. Zhang, W. Wang, Y. Liang, Q. Wen, Deep learning for multivariate time series imputation: A survey, arXiv preprint arXiv:2402.04059 (2024).

[12] M. Kazijevs, M. D. Samad, Deep imputation of missing values in time series health data: A review with benchmarking, Journal of biomedical informatics 144 (2023) 104440.

[13] Z. Che, S. Purushotham, K. Cho, D. Sontag, Y. Liu, Recurrent neural networks for multivariate time series with missing values, Scientific reports 8 (1) (2018) 6085.

[14] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, Y. Li, Brits: Bidirectional recurrent imputation for time series, Advances in neural information processing systems 31 (2018).

[15] J. Yoon, W. R. Zame, M. Van Der Schaar, Estimating missing data in temporal data streams using multi-directional recurrent neural networks, IEEE Transactions on Biomedical Engineering 66 (5) (2018) 1477–1490.

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

[17] A. Cini, I. Marisca, C. Alippi, Multivariate time series imputation by graph neural networks, arXiv preprint arXiv:2108.00298 (2021).

[18] I. Marisca, A. Cini, C. Alippi, Learning to reconstruct missing data from spatiotemporal graphs with sparse observations, Advances in neural information processing systems 35 (2022) 32069–32082.

[19] W. Du, D. Côté, Y. Liu, Saits: Self-attention-based imputation for time series, Expert Systems with Applications 219 (2023) 119619.

[20] P. Bansal, P. Deshpande, S. Sarawagi, Missing value imputation on multidimensional time series, arXiv preprint arXiv:2103.01600 (2021).

[21] T. Nie, G. Qin, W. Ma, Y. Mei, J. Sun, Imputeformer: Low rankness-induced transformers for generalizable spatiotemporal imputation, in: Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining, 2024, pp. 2260–2271.

[22] B. Jing, D. Zhou, K. Ren, C. Yang, Causality-aware spatiotemporal graph neural networks for spatiotemporal time series imputation, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 1027–1037.

[23] X. Miao, Y. Wu, J. Wang, Y. Gao, X. Mao, J. Yin, Generative semi-supervised learning for multivariate time series imputation, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 8983–8991.

[24] G. Liang, P. Tiwari, S. Nowaczyk, S. Byttner, Higher-order spatio-temporal physics-incorporated graph neural network for multivariate time series imputation, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 1356–1366.

[25] X. Yang, Y. Sun, X. Chen, et al., Frequency-aware generative models for multivariate time series imputation, Advances in Neural Information Processing Systems 37 (2024) 52595–52623.

[26] A. Lotfipoor, S. Patidar, D. P. Jenkins, Transformer network for data imputation in electricity demand data, Energy and buildings 300 (2023) 113675.

[27] T. B. Brown, Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020).

[28] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al., A survey on large language models: Applications, challenges, limitations, and practical usage, Authorea Preprints (2023).

[29] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, R. McHardy, Challenges and applications of large language models, arXiv preprint arXiv:2307.10169 (2023).

[30] Z. Hu, Y. Gao, L. Sun, M. Mae, A novel attention-enhanced llm approach for accurate power demand and generation forecasting, Renewable Energy (2025) 123465.

[31] S. Lim, R. Schmälzle, Artificial intelligence for health message generation: an empirical study using a large language model (llm) and prompt engineering, Frontiers in Communication 8 (2023) 1129082.

[32] J. Su, C. Jiang, X. Jin, Y. Qiao, T. Xiao, H. Ma, R. Wei, Z. Jing, J. Xu, J. Lin, Large language models for forecasting and anomaly detection: A systematic literature review, arXiv preprint arXiv:2402.10350 (2024).

[33] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, et al., Time-llm: Time series forecasting by reprogramming large language models, arXiv preprint arXiv:2310.01728 (2023).

[34] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, T.-S. Chua, Data-efficient fine-tuning for llm-based recommendation, in: Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, 2024, pp. 365–374.

[35] Y. Xia, J. Kim, Y. Chen, H. Ye, S. Kundu, C. C. Hao, N. Talati, Understanding the performance and estimating the cost of llm fine-tuning, in: 2024 IEEE International Symposium on Workload Characterization (IISWC), IEEE, 2024, pp. 210–223.

[36] C. Chang, W.-Y. Wang, W.-C. Peng, T.-F. Chen, Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters, arXiv preprint arXiv:2308.08469 (2023).

[37] X. Yu, Z. Chen, Y. Ling, S. Dong, Z. Liu, Y. Lu, Temporal data meets llm–explainable financial time series forecasting, arXiv preprint arXiv:2306.11025 (2023).

[38] Y. Hu, Q. Li, D. Zhang, J. Yan, Y. Chen, Context-alignment: Activating and enhancing llm capabilities in time series, arXiv preprint arXiv:2501.03747 (2025).

[39] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[40] T. Blu, P. Thévenaz, M. Unser, Linear interpolation revitalized, IEEE Transactions on Image Processing 13 (5) (2004) 710–719.

[41] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[42] S. Hochreiter, Long short-term memory, Neural Computation MIT-Press (1997).

[43] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting?, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 37, 2023, pp. 11121–11128.

[44] A. Vaswani, Attention is all you need, Advances in Neural Information Processing Systems (2017).

[45] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 11106–11115.

[46] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, itransformer: Inverted transformers are effective for time series forecasting, arXiv preprint arXiv:2310.06625 (2023).

[47] Y. Nie, N. H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, arXiv preprint arXiv:2211.14730 (2022).

[48] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners (2019).

[49] C. Cheadle, M. P. Vawter, W. J. Freed, K. G. Becker, Analysis of microarray data using z score transformation, The Journal of molecular diagnostics 5 (2) (2003) 73–81.

[50] X. Yi, Y. Zheng, J. Zhang, T. Li, St-mvl: Filling missing values in geo-sensory time series data, in: Proceedings of the 25th international joint conference on artificial intelligence, 2016.

[51] A. Trindade, ElectricityLoadDiagrams20112014, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C58C86 (2015).

[52] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv preprint arXiv:1707.01926 (2017).

[53] A. Cini, I. Marisca, C. Alippi, Filling the g_ap_s: Multivariate time series imputation by graph neural networks, arXiv preprint arXiv:2108.00298 (2021).

[54] H. Zhao, K. Sun, A. Dezfouli, E. V. Bonilla, Transformed distribution matching for missing value imputation, in: International Conference on Machine Learning, PMLR, 2023, pp. 42159–42186.