ImpGPT: Adaptive Patch-Based GPT-2 Transformer for Multivariate Time Series Imputation

Shuhui Ye[1], Guo Li[1,2*], Yirui Cheng[1], Chenxi Dong[1]
[1]School of Artificial Intelligence and Software Engineering, Nanyang Normal University, Nanyang, Henan, 473061, China
[2]Collaborative Innovation Center of Intelligent Explosion-proof Equipment，Henan Province，Nanyang, Henan, 473061, China
E-mail: airforce1980@126.com
*Corresponding author

Keywords: Multivariate time series, missing value imputation, adaptive patch, gpt-2, generative transformer

Received: August 27, 2025

Missing value imputation for multivariate time series data is a critical issue in time series analysis, with wide applications in industrial monitoring, sensor data recovery, and intelligent transportation systems. Traditional imputation methods often perform poorly under high missing rates, struggling to recover lost data trends. This paper proposes an adaptive Patch partitioning-based multivariate time series missing value imputation model—ImpGPT. This model combines an adaptive Patch partitioning mechanism and a Box encoding module. Through the adaptive Patch partitioning mechanism, time series data are dynamically divided into multiple Patches to capture local feature changes in the sequence. The size and position of Patch partitioning are dynamically generated and adjusted in real time by a lightweight network, which avoids the loss of temporal information that may be caused by traditional fixed partitioning methods.The Box encoding module encodes the geometric information and missing features of each Patch into structural vectors, explicitly associating missing patterns with temporal structures and enhancing the model's sensitivity to local structural changes.ImpGPT adopts a frozen and fine-tuned GPT-2 generative Transformer model to encode and model Patch sequences. This retains the strong sequence representation capability of the original GPT-2 and further optimizes the model to adapt to specific tasks. Combined with a masked normalization mechanism, it ensures accurate imputation of missing data.To verify the effectiveness of the model, we selected two datasets: one is a real flight sensor record from a certain aviation system, which includes parameters such as pitch angle, roll angle, heading angle and their corresponding angular velocities; the other is the public Electricity dataset.Experimental results show that under various missing rates, ImpGPT significantly outperforms existing benchmark methods in metrics including Mean Squared Error (MSE), Mean Absolute Error (MAE), Symmetric Mean Absolute Percentage Error (SMAPE), and Normalized Root Mean Squared Error (NRMSE). Especially under the high missing rate of 75%:In the aviation sensor dataset, the MSE of ImpGPT is 0.0346, which is 1.7% lower than that of PatchTST (0.0352) and 23.1% lower than that of GPT4TS (0.045).In the Electricity dataset, the MSE of ImpGPT is 0.1253, which is 16.3% lower than that of PatchTST (0.1498) and 7.6% lower than that of GPT4TS (0.1356).These results indicate that ImpGPT still maintains good recovery accuracy even in extreme missing scenarios. Ablation experiments further verify that the adaptive Patch partitioning mechanism and model structure play key roles in improving imputation accuracy. ImpGPT performs excellently in handling high-missing-rate multivariate time series imputation tasks, with strong robustness and broad application potential.

Povzetek: Študija predstavi ImpGPT, ki z adaptivnim razrezom časovnih zaporedij modelira lokalno strukturo in zanesljivo imputira manjkajoče vrednosti v večspremenljivskih časovnih vrstah tudi pri visokih stopnjah manjka.

1 Introduction

In time series data, data missing is a common yet critical issue. Missing data is typically caused by sensor failures, transmission errors, environmental interference, and other factors [1]. During the analysis of multivariate time series, if the impact of missing data is not considered, it will not only weaken the ability to model dependencies between variables but also undermine the effectiveness of subsequent tasks [2]. Therefore, developing effective time series missing value imputation models is of great significance for improving the performance of downstream analysis.

Deep learning is developing rapidly. Recently, a growing number of studies have leveraged pre-trained language models for time series representation learning.For example, GPT4TS [3], by freezing and fine-tuning pre-trained language models, has its own advantages in handling incomplete multivariate time series. However, these models divide time series into patches, which can then be treated as tokens in natural language processing to be input into GPT2 for learning. Although this method has achieved certain results, it only treats time series as text and aligns them with the model. While this method well fits the input characteristics of GPT2, it ignores the temporal information correlation characteristics of the sequence. As shown in the upper part of Figure 1 this method breaks the continuity of the sequence and the coherent contextual information. When GPT2 is learning, it cannot fully understand the semantic information of the sequence, which has a great impact on the time series missing value imputation model. To solve this problem, this paper proposes a new multivariate time series missing value imputation model—ImpGPT, which introduces a new patch division method to retain the integrity of local sequence information as much as possible. In figure 1, from the black circle to the red circle, the originally disconnected complete wave is retained under our division method. ImpGPT adopts an adaptive Patch division mechanism, enabling time series data to perform effective local feature extraction while maintaining temporal continuity. This method can dynamically adjust the data partitioning method according to the actual missing pattern of the data, thereby maximizing the retention of the structural information of the time series data. On this basis, ImpGPT combines the frozen pre-trained GPT-2 generative Transformer model, making full use of its powerful contextual modeling ability to capture the global dependencies in the data, thus significantly improving the imputation accuracy and the robustness of the model.

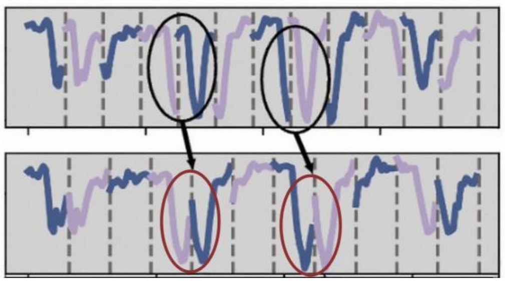
Figure 1: Comparison of different patch partitioning methods.

2 Research status

In recent years, machine learning-based approaches to handling missing data have garnered growing attention from researchers [4]. These imputation methods adopt a predictive paradigm, constructing forecasting models based on originally observed data to predict values at missing positions. Interpolation methods aim to fit a "smooth curve" to the observed results, thereby reconstructing missing values through local interpolation [5] [6].

In recent years, deep imputation models have shown a diversified evolutionary context, and their technical routes have formed a multi-branch development pattern, specifically including: diffusion-based probabilistic modeling, non-autoregressive multi-resolution frameworks, graph neural network (GNN) architectures for spatial dependency modeling, self-attention mechanisms integrated with generative adversarial networks,autoregressive reconstruction based on masked self-attention (SAITS), and bidirectional recurrent neural network schemes for real-time imputation scenarios [7-12]. The development of these technical branches not only adapts to the practical needs of the continuous complication of missing patterns in time series data but also reflects the differences in the understanding of missing mechanisms among different modeling paradigms.

Transformer-based time-series modeling techniques have continuously broken through performance bottlenecks in multiple dimensions and also demonstrated advantages in multivariate time-series missing value imputation tasks. At the infrastructure level, the ProbSparse mechanism proposed by Informer effectively reduces computational complexity while maintaining the ability to model long-range dependencies,However, it mainly focuses on global attention sparsity and has insufficient capability to capture short-term local missing information [13]. Reformer improves the efficiency of approximate attention computation through Locality-Sensitive Hashing (LSH),but its ability to perceive the structure of missing positions is limited, leading to a decline in recovery performance when there are a large number of missing values in local segments [14]. Autoformer strengthens the modeling capability for periodic structures, yet it tends to cause overfitting in imputation when processing non-periodic or mixed-structure data, which affects the accuracy of results [15]. FEDformer combines Fourier transform with attention mechanisms, and its frequency-domain modeling characteristic determines that it responds slowly to random missing values and mutation positions, limiting the flexibility of imputation [16]. ETSformer introduces a season-trend decomposition module, but it does not explicitly model dynamic structural dependencies between multiple variables [17]. DLinear as a linear modeling baseline, performs well when the data distribution is relatively stable. However, it lacks the ability to model nonlinear interactions, resulting in limited accuracy when facing dynamically changing complex sequence structures [18]. The static partitioning model PatchTST enhances the model's expressive ability for local structures and short-term patterns. Nevertheless, it does not explicitly model the continuity between patches and the interaction structure between variables, leading to performance bottlenecks in long-range dependency missing imputation scenarios [19].

In terms of pre-training technology, researchers are attempting to migrate the capabilities of large models that have achieved breakthroughs in the field of natural language processing to time series modeling tasks. Timer has constructed a general temporal Transformer pre-training framework to improve the efficiency of time series modeling [20].In addition to Timer, there are other general temporal Transformer pre-training frameworks. TimeNet laid the foundation for early temporal pre-training by extracting transferable temporal representations through large-scale training of stacked RNN structures [21]. GPT4TS successfully migrated the GPT-2 model to temporal tasks, pioneering a new paradigm for the application of large models in temporal modeling [3]. However, existing models often ignore the structural nature of missing information itself during the sequence normalization and feature extraction stages, thereby weakening the model's ability to perceive missing mechanisms.

To address the limitations of existing methods, ImpGPT proposes an innovative Adaptive Patch Partitioning Mechanism and Box Encoding Technology. Unlike traditional sliding window methods and graph-based encoders, ImpGPT can dynamically adjust the size and position of patches, flexibly adapt to different missing patterns, effectively capture local changes, and avoid the information loss caused by static partitioning. Meanwhile, through encoding the geometric properties of patches, the Box Encoding Technology enables the model to better capture local structures and global dependencies in time series data, reducing reliance on complex external graph structures. In time series data processing tasks, GPT-2 has significant advantages over traditional Transformers in capturing long-term dependencies and complex patterns. Its decoder-based autoregressive mechanism makes it more suitable for handling long-term dependencies in time series data. Unlike the original GPT model, which is mainly designed for language tasks, GPT-2 exhibits stronger context modeling capability, especially for complex time series prediction and imputation tasks. To highlight the differences between various methods in terms of adaptability, robustness to missing structures, and context modeling capability, representative models are compared in Table 1.

Table 1: Model comparison and limitation analysis

Model	Key Technologies	Limitations
PatchTST	Fixed-length patch partitioning	Cannot flexibly handle dynamic missing patterns and lacks modeling of interactions between variables.
FEDformer	Fourier/wavelet frequency-domain modeling	Responds slowly to sudden missing values and struggles to handle irregular missing patterns.
SAITS	Dual-stream self-attention, masked reconstruction	Insufficient capture of long-range dependencies and difficulty adapting to high missing rates and irregular time intervals.
DLinear	Trend/residual decomposition	Only applicable to data with stable distributions and lacks the ability to model dynamic changes.
GNN	Graph convolution + temporal recursion	Relies on fixed graph structures and lacks flexible adaptation to missing patterns in time series data.
GPT4TS	Migration of GPT-2 to time series tasks	Lacks the ability to explicitly utilize missing structures and struggles to capture complex missing patterns.

3 Model design

The ImpGPT model designed in this study is intended to address the multivariate time series imputation problem under high missing rate conditions. Firstly, the Adaptive Patch partitioning and the pre-trained capability of GPT-2 are adopted to improve the imputation accuracy when facing missing data. Secondly, by freezing most weights of GPT-2 and only fine-tuning task-specific layers, the computational efficiency is optimized and the training time is reduced. Meanwhile, the model's robustness and generalization ability are enhanced to ensure that it can effectively perform imputation under different missing patterns and data types.

3.1 Overall framework of the model

ImpGPT processes inputs using a masked normalization mechanism, calculating statistics solely based on valid observations to maintain data stability. It dynamically generates Patches through adaptive Patch partitioning, adjusting sampling positions and sizes. Box encoding maps geometric attributes into structural vectors, which are linearly projected into the feature space via feature embedding. For global modeling, a frozen and fine-tuned GPT-2 is introduced: the attention layers and feed-forward network layers are frozen to retain pre-trained knowledge, while positional encoding and LayerNorm layers are fine-tuned to adapt to temporal structures and capture long-range dependencies across time steps. The output layer predicts missing values through a fully connected layer and restores them to the original physical scale via denormalization, ensuring the physical consistency of imputation results. The complete framework is shown in Figure 2.

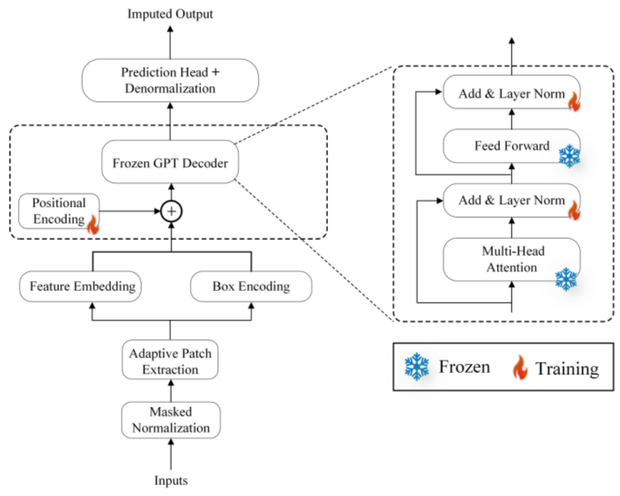

Figure 2: Framework diagram of the ImpGPT model.

3.2 Input normalization and mask processing

The evaluation of deep learning models for imputation tasks on time series datasets heavily relies on masking practices, where certain data points are intentionally designated as missing to simulate incomplete data conditions [22]. Therefore, ImpGPT introduces a masked normalization mechanism in the input stage to avoid missing values interfering with the statistical calculation of variables.

The original multivariate time series input is $X \in R^{B\times L\times M}$, where B is the batch size, L is the sequence length, and M is the variable dimension. The missing mask is $M \in \{0,1\}^{B\times L\times M}$. To ensure that the normalization process is based solely on valid observations, the model calculates the mean μ_m and standard deviation σ_m for each variable dimension as follows:

$$\mu_m = \frac{\sum_t X_{t,m} \cdot M_{t,m}}{\sum_t M_{t,m}} \quad (1)$$

$$\sigma_m = \sqrt{\frac{\sum_t \left(X_{t,m} - \mu_m\right)^2 \cdot M_{t,m}}{\sum_t M_{t,m}} + \varepsilon} \quad (2)$$

where ε is a small positive constant to improve numerical stability. The normalized input is:

$$\hat{X}_{t,m} = \frac{\left(X_{t,m} - \mu_m\right)}{\sigma_m} \cdot M_{t,m} \quad (3)$$

The masked normalization mechanism serves as a fundamental step in the encoding stage of ImpGPT, standardizing the numerical distribution of inputs to facilitate subsequent Patch extraction and context modeling.

3.3 Adaptive patch partitioning mechanism

To dynamically capture local structural changes in multivariate time series, ImpGPT designs a learnable Patch partitioning mechanism. This mechanism dynamically generates the center and width parameters of each Patch through length-scalable Patches, enabling segmented modeling of local regions in the sequence, as shown in Figure 3.

After normalization, the sequence $\hat{X} \in R^{B\times L\times M}$ is processed by a lightweight linear network f_{shift}. The objective of this network is to dynamically calculate the center offset $\Delta x \in R^{B\times N\times 1}$ and the Patch length extension $\Delta\omega \in R^{B\times N\times 2}$ for each Patch, where N denotes the number of Patches.The f_{shift}network computes these two parameters through linear layers and activation functions, thereby achieving fine-grained control over local regions. Its calculation formulas are as follows:

$$\Delta x = tanh\left(W_{offset} \cdot \hat{X}\right) \quad (4)$$

$$\Delta\omega = ReLU\left(tanh\left(W_{extend} \cdot \hat{X}\right)\right) \quad (5)$$

Among them, W_{offset} and W_{extend} are two key weight matrices in the network, which are used to learn the center offset and length extension of each Patch, respectively. The center offset Δx is transformed using the *tanh* activation function, which restricts its value within the range of [-1, 1] to avoid excessive offsets and ensure model stability. The Patch length extension $\Delta\omega$ is transformed using the *ReLU* activation function, ensuring its value is positive and has an adjustable range.

The generated center offset Δxand Patch length extension Δω are used to calculate the center position and boundary range of each Patch. For the i-th Patch, its center position c_i and boundary interval $[\, s_i, e_i,]$ can be expressed as:

$$c_i = x_i + \Delta x_i \quad (6)$$

$$s_i = c_i - \frac{\Delta\omega_i}{2} \quad (7)$$

$$e_i = c_i + \frac{\Delta\omega_i}{2} \quad (8)$$

The sampling region of each Patch interval $[s_i, e_i,]$ is uniformly divided into D time points, whose positions are:

$$pos_{i,k} = s_i + \frac{k}{D-1} \cdot (e_i - s_i), \quad k = 0,1,\dots,D-1 \quad (9)$$

Since these positions are continuous real-number coordinates, they cannot directly index sequence values. To overcome the limitations of traditional interpolation methods such as nearest-neighbor interpolation and spline interpolation under high missing rate conditions, this study adopts grid sampling technology for interpolation. Traditional methods typically rely on fixed neighborhoods to impute missing values, which may fail to accurately recover data trends, especially in complex time series. In contrast, grid sampling can more flexibly capture changes in the data by dynamically adjusting sampling positions and sizes, effectively avoiding the loss of temporal fidelity [1] [2]. The model employs the *grid_sample* function to perform bilinear interpolation on $\hat{X}$, extracting features corresponding to the Patch from the input. By generating non-integer coordinates for interpolation calculations on the input sequence, local features are obtained. In the case of *1D* time series, the input sequence is rearranged into a *2D* format, and then interpolation is performed via the *grid_sample* function to ensure smooth feature extraction. Specifically, the grid_sample function uses target sampling positions to extract local features from the input sequence. The sampling position of each Patch is dynamically calculated and can be automatically adjusted according to its local structure. The interpolated value $\hat{X}_{i,k}$ can be expressed as:

$$\hat{X}_{i,k} = \sum_{j\in\mathcal{N}(pos\ i,k)} G\left(pos_{i,k}, j\right) \cdot \hat{X}_j \quad (10)$$

Where $\mathcal{N}\left(pos_{i,k}\right)$ denotes the two integer indices adjacent to $\mathcal{N}\left(pos_{i,k}\right)$, To avoid artifacts caused by temporal aliasing, the model uses a smooth interpolation weight function $G(\cdot)$, which is defined as follows:

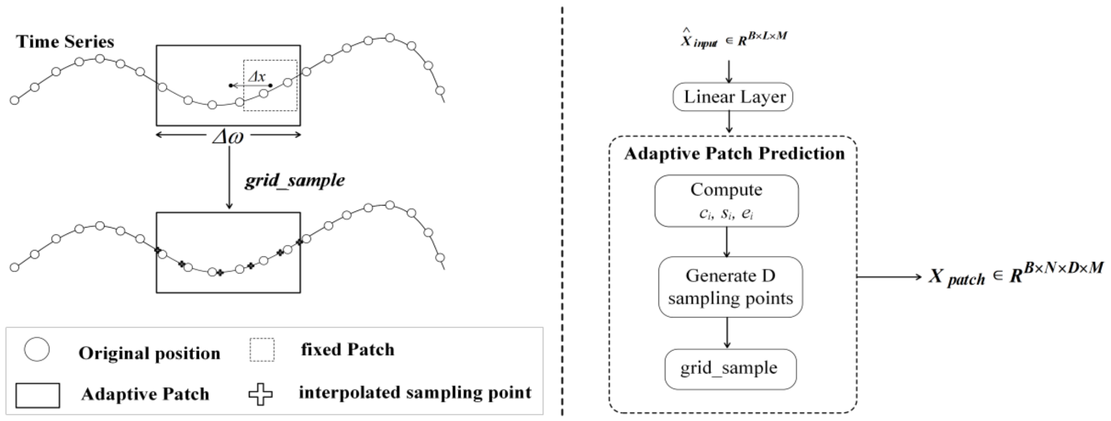
Time Series
Δx
Δω
grid_sample
Original position
fixed Patch
Adaptive Patch
interpolated sampling point
$\hat{X}_{input} \in R^{B\times L\times M}$
Linear Layer
Adaptive Patch Prediction
Compute c_i, s_i, e_i
Generate D sampling points
grid_sample
$X_{patch} \in R^{B\times N\times D\times M}$

Figure 3: Adaptive patch generation.

$$G(i,j) = max(0,1-|i-j|) \quad (11)$$

This function ensures linear fusion between non-integer positions and their two nearest observation points, enabling a smooth and differentiable feature extraction process. It avoids the loss of low-frequency information and high-frequency noise, thereby reducing artifacts caused by temporal aliasing. By applying this operation to all D positions within each Patch, Patch features are formed:

$$X_{patch} = grid\_sample(\hat{X}, pos) \quad (12)$$

Where $pos \in R^{B\times N\times D}$ represents the Patch sampling coordinates. Before interpolation sampling, the input sequence is expanded into a *2D* format, and linear interpolation is performed along the time axis via *grid_sample* this ensures smooth extraction of features within each Patch.

The overall workflow of this module is shown on the right side of Figure 3: the input sequence first passes through a linear network to predict sampling position parameters, which are further mapped to sampling coordinates on the continuous time axis. The model then performs interpolation sampling on the normalized input sequence based on these coordinates to construct Patch features with local integrity, ultimately obtaining an adaptive Patch tensor of dimension $X_{patch} \in R^{B\times N\times D\times M}$.

3.4 Structure-aware encoding module

ImpGPT introduces a structure-aware encoding module (Box Encoding), which encodes the geometric information and missing features of each Patch into a structural vector.

Specifically, for each Patch, its structural features include the following elements: Interval position (s_i, e_i), length $l_i = e_i - s_i$, missing ratio r_i, and mask statistics $mask_i$. This structural information is concatenated into a vector and input into a lightweight Multi-Layer Perceptron (MLP) to obtain the structural encoding vector of the corresponding Patch:

$$b_i = MLP([s_i, e_i, l_i, r_i, mask_i]) \in R^{d_{model}} \quad (13)$$

This structural vector is input into subsequent modeling modules along with the corresponding Patch content. This design not only provides positional information but also enhances the model's perception of local variation ranges and data integrity, facilitating improved cross-Patch context modeling [21].

3.5 GPT-2 encoder

After Patch feature extraction and structure encoding, ImpGPT employs the pre-trained language model GPT-2 to perform global modeling on the Patch sequence, aiming to capture long-range dependencies and cross-variable contextual information. Compared with traditional time series models such as RNN and TCN, GPT-2, built on a stacked Transformer decoder structure, boasts non-recursive modeling capabilities and stronger expressive power, particularly demonstrating significant advantages in scenarios with high missing rates and non-stationary fluctuations [23][24].

3.5.1 Patch embedding and fusion modeling

Taking the i-th Patch as an example, its content can be represented as a tensor $P_i \in R^{D\times M}$, where D denotes the number of sampling points and M is the variable dimension. ImpGPT first flattens each Patch and maps it linearly to the dimension space required by GPT-2 ($d_{model} = 768$, consistent with the pre-trained dimension of GPT-2):

$$e_i = Linear(Flatten(Patch_i)) \in R^{d_{model}} \quad (14)$$

Subsequently, all Patch embedding vectors form a sequence $E = [e_1; e_2; \dots; e_N] \in R^{N\times d_{model}}$, which are added to their corresponding structural vectors bi to obtain the fused Patch representation:

$$\tilde{e}_i = e_i + b_i \quad (15)$$

The fused Patch sequence is fed into the GPT-2 backbone model for global context modeling, and the output result is:

$$H = GPT2(\tilde{E}) \in R^{N\times d_{model}} \quad (16)$$

Here, H_i represents the representation of the i-th Patch after integrating global semantics, serving as the input for the subsequent reconstruction module.

3.5.2 Freezing and fine-tuning strategy

In ImpGPT, we fine-tune the positional encoding and LayerNorm layers, while freezing the self-attention layers and feed-forward networks (FFN). This strategy aims to balance computational efficiency and imputation accuracy, especially in data scenarios with high missing rates. To handle time series data with high missing rates, ImpGPT primarily enhances its imputation capability through Adaptive Patch Partitioning and Box Encoding Technology. Traditional imputation methods tend to suffer from overfitting or information loss under high missing rates. To overcome this limitation, ImpGPT adopts a dynamically adjustable patch partitioning strategy, which can automatically adjust the size and position of patches according to different missing patterns. This enables more flexible capture of local changes in the data and mitigates the challenges posed by high missing rates.

Since the self-attention layers and FFN have already learned the capabilities of long-range dependency modeling and context information capture during the pre-training phase of the model, we choose to freeze these two components to retain their pre-trained knowledge and avoid wasting computational resources during the fine-tuning phase. During the pre-training of GPT-2, positional encoding is mainly used to capture sequentiality in language; however, the temporal structure of time series data differs from the sequence in natural language. Fine-tuning the positional encoding layer helps the model better grasp the sequential nature of time series data, particularly when dealing with dynamic missing patterns. On the other hand, fine-tuning the LayerNorm layers helps the model adapt to the data distribution under conditions of missing data and high missing rates, thereby improving the model's stability and imputation accuracy.

In environments with high missing rates, the fine-tuned positional encoding layer and LayerNorm layers can enhance the model's ability to handle missing data, avoid overfitting under high missing rates, and enable the model to better recover the lost temporal information.Although the original sinusoidal positional encoding of GPT-2 is designed for natural language, its core feature modeling relative positional relationships through periodic differences of trigonometric functions highly aligns with the requirement of time series for modeling "relative time intervals" [25]. This design is also applicable to time series data. Despite the fact that time steps in time series data have fixed sampling intervals, while the word order in natural language has no physical intervals, the fine-tuned positional encoding has acquired time-aware capability. Through fine-tuning, the relative positional representations of the positional encoding are recalibrated to the physical properties of time intervals; thus, there is no need for additional explicit temporal encoding to transmit temporal structural information. This fine-tuning strategy is regarded as a standard practice, which helps improve the performance of downstream tasks and minimize the workload of fine-tuning [26][27].

3.6 Reconstruction and imputation output

Imputation, the process of identifying missing values and replacing them with substitute values, is also referred to as missing value imputation [28]. After completing contextual semantic modeling, ImpGPT maps high-dimensional representations back to the complete time series space through a structured decoding module, enabling the reconstruction of missing positions and the restoration of the original sequence.

First, the contextual representation $H \in R^{B\times N\times d_{model}}$ output by GPT-2 is flattened into a vector, and decoding mapping is performed via a fully connected network:

$$\tilde{X} = Linear\big(Flatten(H)\big) \in R^{B\times(L\times M)} \qquad (17)$$

To ensure the model performs consistent modeling of the entire sequence structure, ImpGPT adopts an end-to-end supervision mechanism. Instead of only interpolating at missing points, The model is trained to predict the entire sequence, thereby enhancing the continuity and contextual coherence of the reconstructed data.To enhance the numerical consistency between structural information and the original sequence, ImpGPT introduces a mask-aware denormalization strategy in the prediction stage. Combining the mean μ and standard deviation σ, the denormalization is computed as follows:

$$\hat{X}_{final} = \hat{X}_{rec} \cdot \sigma + \mu \qquad (18)$$

This transformation ensures calculations with high observation accuracy, avoids error accumulation, and guarantees the physical rationality and dimensional consistency of predicted values. Finally, ImpGPT outputs $\hat{X}_{final}$ as the imputation result, ensuring the model has efficient recovery capabilities for missing value imputation under multiple missing patterns.

3.7 Loss function design

We adopt a mean squared error (MSE) loss function design based on a mask-guided mechanism. Specifically, in each training iteration, the model randomly generates a missing mask, replaces some positions in the original input with zero values, and feeds the mask information into the model together. After outputting the complete predicted sequence, only the mean squared error (MSE) between the predicted values and the true values at the missing positions is calculated to form the final optimization objective [11][12]. This design effectively avoids the interference of observed values on the gradient direction, allowing the model to focus on the reconstruction task of missing Patches. Specifically, by training on the predicted values of the entire sequence, the model can acquire more contextual information, enabling it to consider broader temporal dependencies and global trends when imputing missing values. This design approach ensures that imputation is not limited to local data points; instead, it improves imputation accuracy and consistency through global modeling. Thus, although the output is the predicted values of the entire sequence, the model's performance in the imputation task is enhanced especially under complex missing patterns, where it can provide more accurate imputation results. The loss function is defined as follows:

$$\mathscr{L}_{MSE} = \frac{1}{|\Omega|}\sum_{(i,j,k)\in\Omega}\left(\hat{x}_{i,j,k} - x_{i,j,k}\right)^2 \qquad (19)$$

$\Omega = \{(i,j,k) \mid mask_{i,j,k} = 0\}$ represents the positions of all missing points in the current training batch. During the model input phase, a binary mask is used to mark missing positions, and missing values are processed via a padding method this ensures the model cannot access the true information of missing values. All evaluation metrics adopt the masking mechanism: only the error at missing positions is calculated, while predictions at non-missing positions are completely excluded. This design avoids data leakage and improves imputation accuracy.

4 Experiments

4.1 Experimental dataset

The experimental data utilized in this study is sourced from a real flight sensor record of a certain aviation system,forming a multivariate time series dataset. This dataset collects key flight status and attitude parameters continuously during flight at a frequency of 1 Hz, including pitch angle (in degrees), roll angle (in degrees), heading angle (in degrees), and their corresponding angular velocities (pitch rate, roll rate, yaw rate, all in degrees per second). In this research, "heading angle (degrees)" is regarded as the core imputation target. To model its periodic characteristics more effectively, this variable is encoded into sine and cosine dual components, which are then used as the final target output. The remaining variables serve as input covariates to participate in the modeling process collectively.

To verify the cross-domain generalization ability of the model, we conducted experiments on the real-world Electricity dataset. The Electricity dataset is a publicly available dataset widely used in time series missing value imputation tasks. Characterized by a 15-minute time granularity, it includes relevant statistical values or feature values such as 321, 96, and tuples (18317, 2633, 5261). The target variable of this dataset is electricity consumption load, which covers a variety of time-related features.The Electricity dataset provides a verification platform for the model's generalization ability. Through its diverse missing patterns and multi-dimensional temporal features, it enables effective evaluation of the model's performance in complex scenarios.

4.2 Experimental design

We followed the experimental setup of TimeNet. To simulate the missing data problem caused by sensor packet loss or communication interruption in real-world scenarios, we constructed six different missing rate settings, which are 12.5%, 25%, 37.5%, 50%, 62.5%, and 75% respectively. We randomly masked the corresponding proportion of data in the sequence and generated missing masks for training and evaluation.

ImpGPT is based on the pre-trained GPT-2, fine-tuning the positional encoding and LayerNorm. Its core innovation is the Adaptive Patch Partitioning Mechanism by setting the Patch size to 8 and step size to 4, overlapping sampling is achieved, and offset prediction is introduced to focus on information-dense regions.To ensure the fairness and comparability of the experiments, we adopted the experimental configuration of TimeNet in all baseline tests and standardized the evaluation process. We verified the effectiveness of this method by comparing it with a series of models, including GPT4TS (a study that fine-tunes language models for time series analysis), Transformer-based models (Autoformer, Informer, Reformer, FEDformer, ETSformer, and PatchTST), and the linear model DLinear.

4.3 Evaluation metrics

To comprehensively evaluate the performance of the ImpGPT model in multivariate time series missing value imputation tasks, this study adopts four commonly used evaluation metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Symmetric Mean Absolute Percentage Error (SMAPE), and Normalized Root Mean Squared Error (NRMSE).

MSE measures the squared error between the predicted values and the true values when the model imputes missing data.

$$MSE = \frac{1}{|\Omega|} \sum_{(i,j,k)\in\Omega} \left(\hat{x}_{ijk} - x_{ijk}\right)^2 \qquad (20)$$

MAE enables an intuitive assessment of the imputation performance across all missing data points.

$$MAE = \frac{1}{|\Omega|} \sum_{(i,j,k)\in\Omega} \left|\hat{x}_{ijk} - x_{ijk}\right| \qquad (21)$$

SMAPE eliminates the impact of magnitude differences on error measurement and serves as a key indicator for evaluating the accuracy of trend recovery and relative imputation accuracy in imputation tasks. To address the instability of SMAPE when values are close to zero, we adopt a smoothing technique in SMAPE calculation: by adding a very small constant ε (1e−8) to the denominator, we avoid computational anomalies caused by zero or near-zero denominators, thereby ensuring the stability of SMAPE under extreme cases.

$$SMAPE = \frac{100\%}{|\Omega|} \sum_{(i,j,k)\in\Omega} \frac{\left|\hat{x}_{ijk} - x_{ijk}\right|}{\frac{\left|\hat{x}_{ijk}\right| + \left|x_{ijk}\right|}{2} + \varepsilon} \qquad (22)$$

NRMSE normalizes errors.Considering the division-by-zero issue that arises when the maximum and minimum values of the data are equal, a very small constantε (1e−8) is also added to the denominator during calculation to ensure the stability of the computation.

$$NRMSE = \frac{\sqrt{\frac{1}{\Omega} \sum_{(i,j,k)\in\Omega} \left(\hat{x}_{ijk} - x_{ijk}\right)^2}}{max(x) - min(x) + \varepsilon} \qquad (23)$$

All evaluation metrics employ a masking mechanism, only the errors at missing positions are calculated, which ensures that the model evaluation focuses on the accuracy of the imputation task.

4.4 Experimental results and analysis

4.4.1 Overall imputation performance comparison

This study evaluates the performance of the ImpGPT model in the missing value imputation task on the aviation sensor dataset and Electricity dataset through experiments. The experimental setup covers evaluations under different missing rates (12.5%–75%), and the adopted evaluation metrics include Mean Squared Error (MSE), Mean Absolute Error (MAE), Normalized Root Mean Square Error (NRMSE), and Symmetric Mean Absolute Percentage Error (SMAPE). Table 2 presents the average evaluation results of all models under the six different missing rates.

On the aviation sensor dataset, ImpGPT outperforms PatchTST and GPT4TS in terms of MSE, MAE, SMAPE, and NRMSE across all missing rates. Particularly under a 75% missing rate, ImpGPT achieves an MSE of 0.0921, an MAE of 0.2073, an SMAPE of 38.76%, and an NRMSE of 0.0339 performing significantly better than PatchTST (MSE: 0.0948, SMAPE: 40.64%) and GPT4TS (MSE: 0.1021, SMAPE: 42.86%). ImpGPT can effectively tackle sensor data recovery tasks under high missing rate scenarios, and it is superior to other models especially in terms of recovery accuracy and stability.

On the Electricity dataset, ImpGPT also demonstrates excellent imputation capability. Under the condition of a 75% missing rate, ImpGPT achieves an MSE of 0.0921, an MAE of 0.2073, an SMAPE of 38.76%, and an NRMSE of 0.0339, continuing to outperform other models. It shows strong recovery ability particularly in sensitive metrics such as SMAPE and NRMSE.

Compared with other baseline models, ImpGPT maintains the lowest error values across all missing rates, especially in high missing rate scenarios. Although Autoformer and Informer perform well at low missing rates, their errors (especially in SMAPE and NRMSE) rise sharply under high missing rate conditions, leading to a significant decline in recovery performance. In contrast, through its Adaptive Patch Partitioning Mechanism and Box Encoding Module, ImpGPT can maintain stable imputation accuracy under complex missing patterns.

ImpGPT exhibits significant advantages in handling multivariate time series missing value imputation tasks, particularly under high missing rate conditions. It not only effectively adapts to the missing patterns of different datasets but also outperforms existing mainstream models in terms of imputation accuracy and stability proving its broad adaptability and reliability in practical applications.

4.4.2 Comparative experiments under different missing rates

To further verify the robustness of the model under different missing rates, we analyzed the changing trends of MSE and MAE of each model under six missing rate settings. Figure 4 and Figure 5 respectively show the performance of ImpGPT and other models under different missing rates on the aviation sensor dataset and the Electricity dataset.

From the experimental results on the aviation sensor dataset and the Electricity dataset, the differences in robustness of various models under different missing rates profoundly reflect the degree of adaptation between their design logic and missing characteristics, further highlighting the advantages of ImpGPT. Particularly in high missing rate scenarios, ImpGPT exhibits the gentlest error growth and demonstrates more stable performance compared with GPT4TS and other baseline models.

In the aviation sensor dataset, ImpGPT exhibits the gentlest error growth: its MSE increases from 0.0175 to 0.0346, and its MAE rises from 0.0642 to 0.0961. This is attributed to ImpGPT's Adaptive Patch Mechanism, which can dynamically adjust the temporal capture range, and the Box Encoding, which accurately models the geometric positions and proportions of missing data thus enabling effective data recovery. In contrast, GPT4TS shows a much steeper error increase: its MSE goes up from 0.017 to 0.045, and its MAE climbs from

Table 2: Model evaluation

Methods	aviation sensor dataset（MSE/MAE/NRMSE/SMAPE）	Electricity dataset（MSE/MAE/NRMSE/SMAPE）
Autoformer	0.0513/0.1644/0.0621/65.5181%	0.1170/0.2425/0.0545/61.3926%
Informer	0.0808/0.1656/0.0703/64.7003%	0.2310/0.3414/0.0617/60.6241%
Reformer	0.0428/0.1191/0.0506/52.5766%	0.2110/0.3217/0.0475/49.2738%
DLinear	0.0398/0.1259/0.0493/54.5472%	0.1571/0.2862/0.0433/51.1285%
FEDformer	0.0627/0.1609/0.0605/65.7942%	0.1528/0.2778/0.0531/61.6657%
ETSformer	0.0670/0.1680/0.0636/71.2949%	0.2316/0.3532/0.0558/66.8154%
PatchTST	0.0245/0.0820/0.0390/43.3606%	0.0948/0.2094/0.0342/40.6423%
GPT4TS	0.0296/0.0935/0.0426/45.7348%	0.1021/0.2200/0.0347/42.8625%
ImpGPT	0.0235/0.0767/0.0386/41.3836%	0.0921/0.2073/0.0339/38.7612%

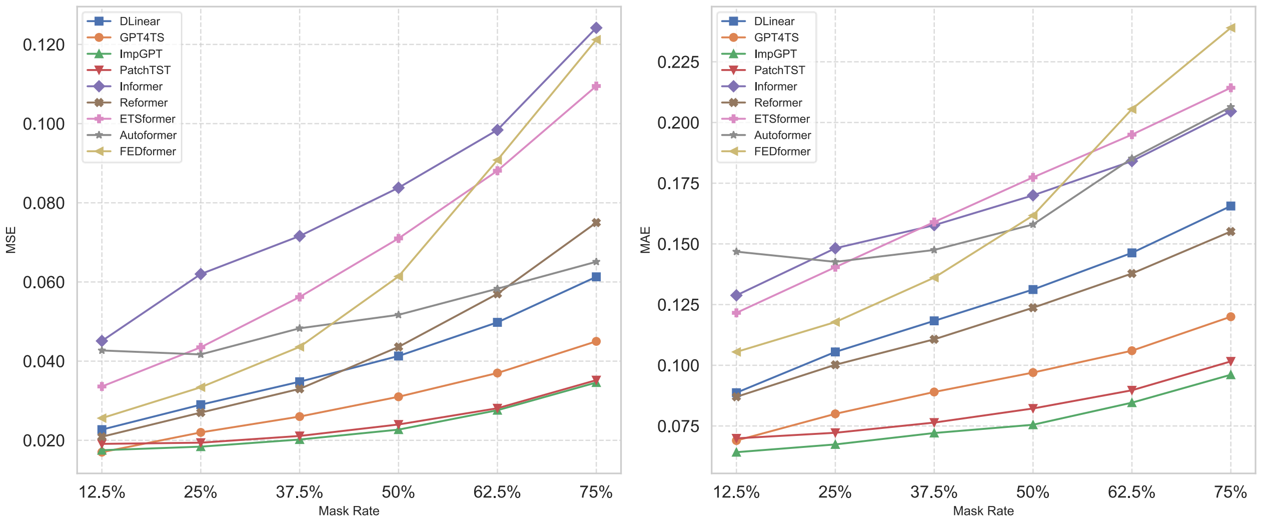

Figure 4: Evaluation results under different missing rates on the aviation sensor dataset.

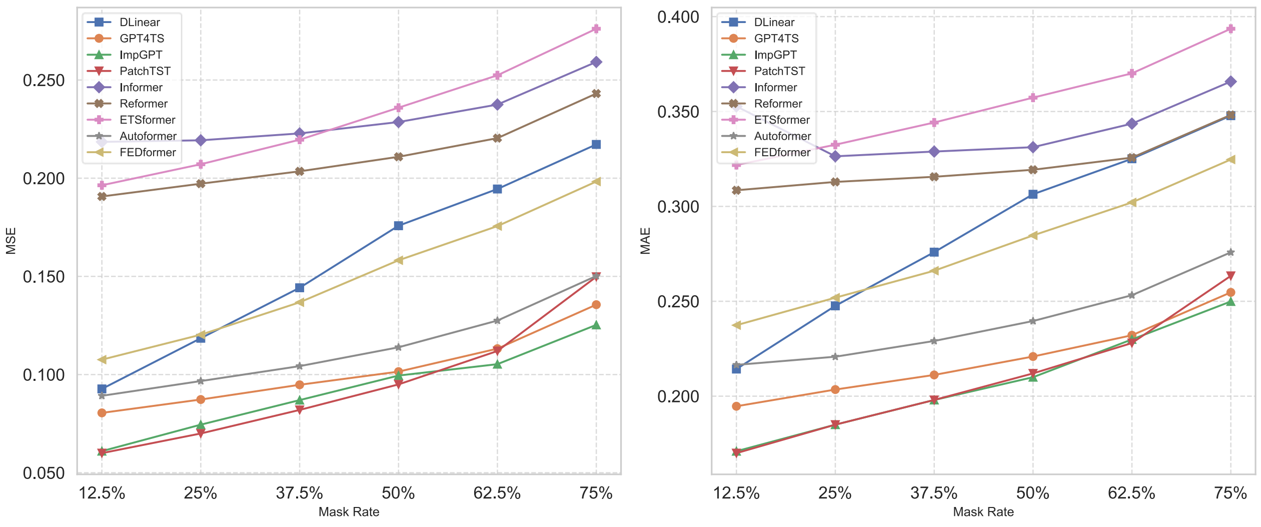

Figure 5: Evaluation results under different missing rates on the electricity dataset.

0.069 to 0.120. The reason for this is that GPT4TS does not explicitly utilize missing structure information; it only treats time series data as text tokens, making it difficult to handle non-stationary missing patterns. PatchTST performs close to ImpGPT when the missing rate is low, but its error surges sharply once the missing rate exceeds 50%. Reformer and FEDformer perform even worse under high missing rates, particularly showing limitations in capturing the long-term dependencies and short-term fluctuations of flight data.

In the Electricity dataset, ImpGPT still achieves the optimal performance: its MSE increases from 0.061 to 0.1253, and its MAE rises from 0.171 to 0.2499. Its Adaptive Patch Mechanism effectively adapts to the "stable-peak" structural changes, while the Box Encoding assists in trend recovery. In contrast, GPT4TS performs poorly in the Electricity dataset with its MSE increasing from 0.0805 to 0.1356 and MAE from 0.1947 to 0.2547. PatchTST's error also rises sharply under high missing rates, as its fixed Patches fail to match the dynamic structure of electricity load data. Models such as Reformer and Informer also see a significant increase in error under high missing rates, performing poorly in terms of MSE and MAE, and are unable to effectively handle the complex missing patterns of electricity load data.

Relying on its Adaptive Patch Mechanism and Box Encoding, ImpGPT maintains low errors and a gentle error growth trend across all missing rate ranges, demonstrating strong robustness in high missing rate scenarios. In contrast, models like GPT4TS, PatchTST, and Reformer show a significant increase in error under high missing rates and struggle to adapt to non-stationary missing patterns validating the practicality of ImpGPT as an industrial-grade time series missing value imputation solution.

4.4.3 Loss convergence analysis

To verify the generalization ability of ImpGPT on domain-specific small datasets such as the aviation sensor dataset, and to avoid the overfitting problem that large pre-trained models may encounter in tasks with limited data volume and specific scenarios, we analyzed the changing trends of training loss, test loss, and validation loss. The results are shown in Figure 6.

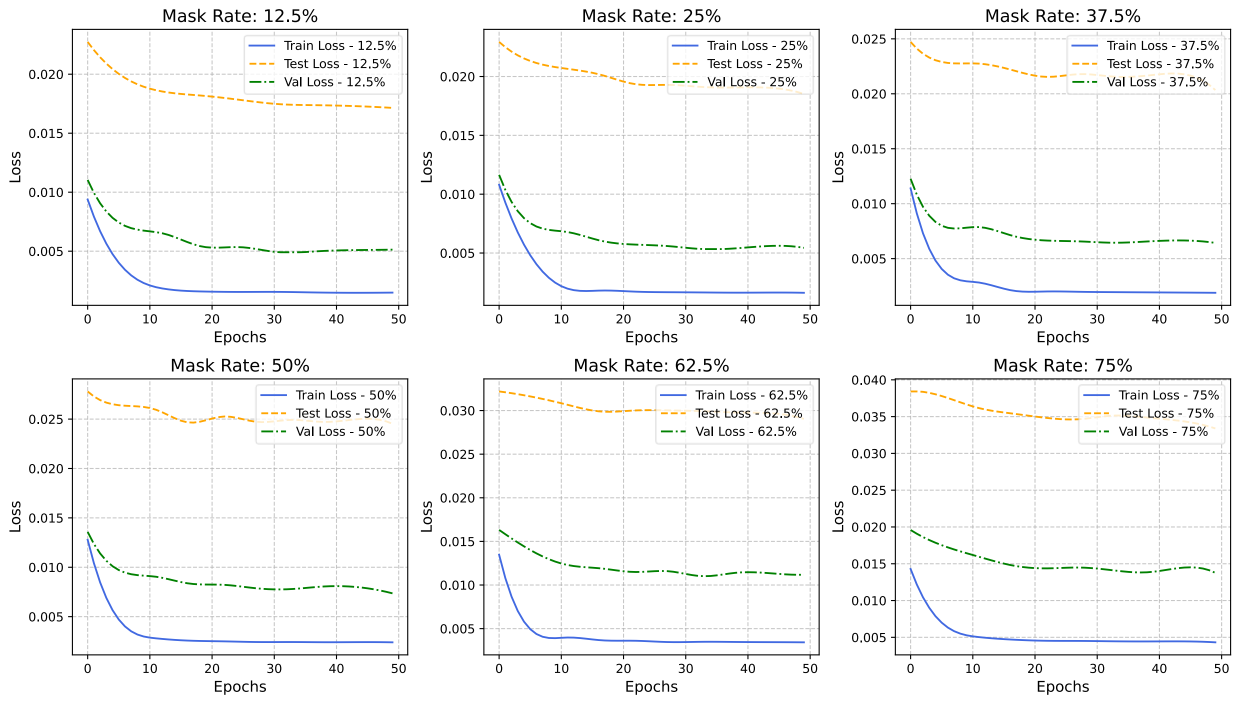

Figure 6: Curves under different missing rate.

Table 3: Results of different expansion lengths

sequence length		**2**	**4**	**8**	**16**	**32**	**64**
64	**MSE**	0.0256	0.0223	0.0209	0.0211	0.0221	0.0231
	MAE	0.0814	0.0736	0.0685	0.0699	0.0724	0.0769
96	**MSE**	0.0272	0.0226	0.0207	0.0214	0.021	0.0223
	MAE	0.0877	0.076	0.0699	0.0701	0.0707	0.0757
128	**MSE**	0.028	0.0228	0.0204	0.0209	0.0211	0.0215
	MAE	0.0935	0.0791	0.0715	0.0706	0.0662	0.0717

Under all missing rate settings (12.5% to 75%), the training (Train), test (Test), and validation (Val) losses of ImpGPT converge stably, with consistent trends across the three types of losses—no overfitting occurs. By freezing the core layers of GPT-2 and fine-tuning the positional encoding and LayerNorm layers, the model effectively avoids overfitting on small datasets. Therefore, ImpGPT demonstrates good generalization ability on the small aviation sensor dataset, with a low risk of overfitting.

4.4.4 Parameter sensitivity study experiment

The adaptive patch expansion mechanism aims to dynamically adjust the size and position of each Patch, so as to adapt to the local changes and global trends of the data. In different data regions, the model may need Patches of different sizes to effectively capture local details and global patterns. Table 3 presents the results of our study on the impact of different expansion lengths. In this paper, different expansion length configurations {2, 4, 8, 16, 32, 64} are set to evaluate the effect of different patch expansion length settings on the model's imputation accuracy. The experiment was conducted on the aviation sensor dataset under a 35% missing rate.to assess the impact of different patch expansion lengths on the model's imputation accuracy.

Reasonably setting the expansion length is crucial for fully leveraging the advantages of the adaptive patch mechanism. When the expansion length is 8, the model achieves relatively optimal imputation effects across all sequence lengths, and the MSE and MAE show the most stability. This indicates that, at this time, the model can capture a sufficient amount of contextual information while preserving local

structural details. However, an overly small expansion length shows relatively high errors in all sequence settings, which suggests that its capture range is limited, making it difficult to learn effective temporal dependencies. On the other hand, although a too - large expansion length covers a wider range, it averages out important local details, making it difficult for the model to distinguish key features.

4.4.5 Interpretability analysis

To intuitively demonstrate the model's learning process for key temporal regions, we compared the performance of four models—ImpGPT, PatchTST, GPT4TS, and Autoformer—on the aviation sensor dataset. To analyze the learning dynamics and interpretability of different models during training, we presented the saliency maps of each model at Batch 0, 100, and 200 in Figure 7. Additionally, we used these saliency maps to compare and show the changes in each model's attention to key time steps throughout the training process.

Through the analysis of saliency maps, ImpGPT exhibits a significant ability to focus on key time steps. Particularly in the late stage of training, its focus gradually converges on critical moments, which significantly improves imputation accuracy. PatchTST's saliency maps show relatively stable changes, indicating that it has limited ability to capture temporal dynamic changes. GPT4TS exhibits small gradient fluctuations and demonstrates a certain degree of focusing ability, but this ability is less pronounced than that of ImpGPT. Autoformer shows relatively uniform attention to key time steps and lacks focus on specific moments, which affects imputation accuracy. ImpGPT's Adaptive Patch Mechanism helps it more flexibly capture critical moments, thereby achieving better imputation results.

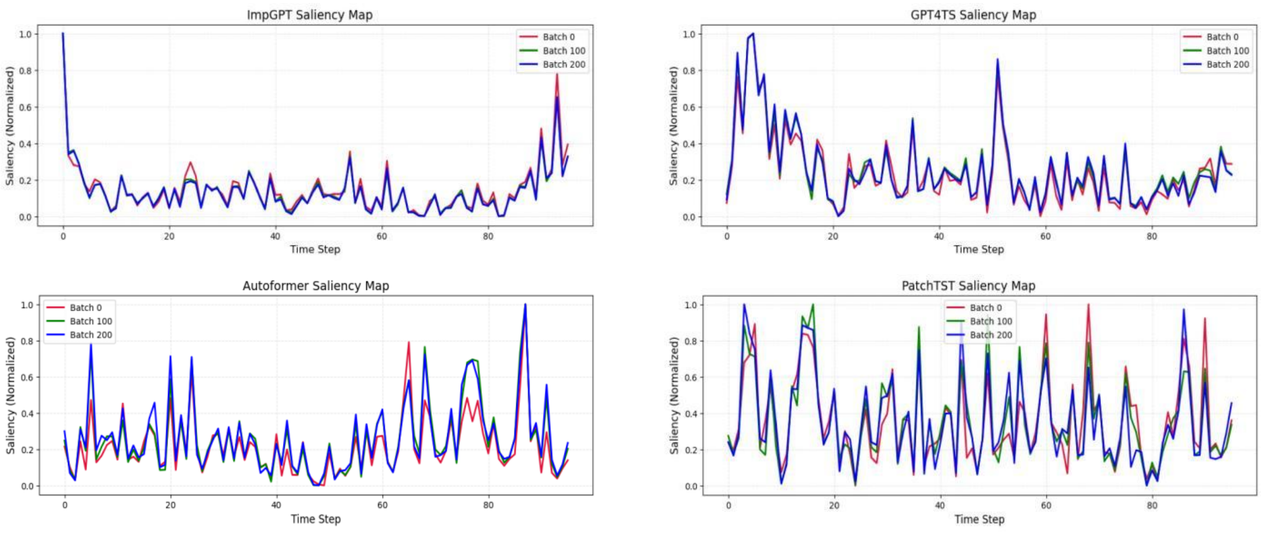

Figure 7: Comparison of saliency maps across different models.

4.5 Ablation experiments

In multivariate time series research, Ablation Experiments are used to verify the role of core components. After removing the key fusion module, the model's imputation error increases significantly this is to prove the importance of this module in temporal feature extraction [29]. We conducted Ablation Experiments under various missing rates on the aviation sensor dataset, as shown in Figure 8. The experiments include the Full Model and its five variants: removing the Box Encoder (w/o Box), removing the Patch Module (w/o Patch), removing both the Box Encoder and the Patch Module (w/o Box+Patch), removing the pretraining strategy (w/o Pretrain), and removing the freezing mechanism (w/o Frozen).

The Full ImpGPT model achieves excellent imputation performance across different missing rates. By contrast, when the Box Encoder or Patch Module is removed, the model accuracy generally decreases—this trend is more pronounced under high missing rate conditions, indicating that these two modules play a key role in modeling local structures and contextual boundaries. Furthermore, when both modules are removed simultaneously, the performance degrades even more significantly, which suggests a certain synergistic effect between them.

The pretraining strategy also has a positive impact on improving the model's initial performance, while the freezing mechanism provides moderate stability under high missing rates. Removing the pretraining strategy (w/o Pretrain) that is, not using the pre-trained GPT-2 model and training from scratch—leads to a decline in the model's performance across all missing rates. At a 75% missing rate, the MSE is 0.0380 and the MAE is 0.1099, which are significantly higher than those of the Full Model (MSE: 0.0364, MAE: 0.1073). This indicates that pretraining has a significant impact on the model's performance; removing pretraining reduces the model's ability to learn temporal features.

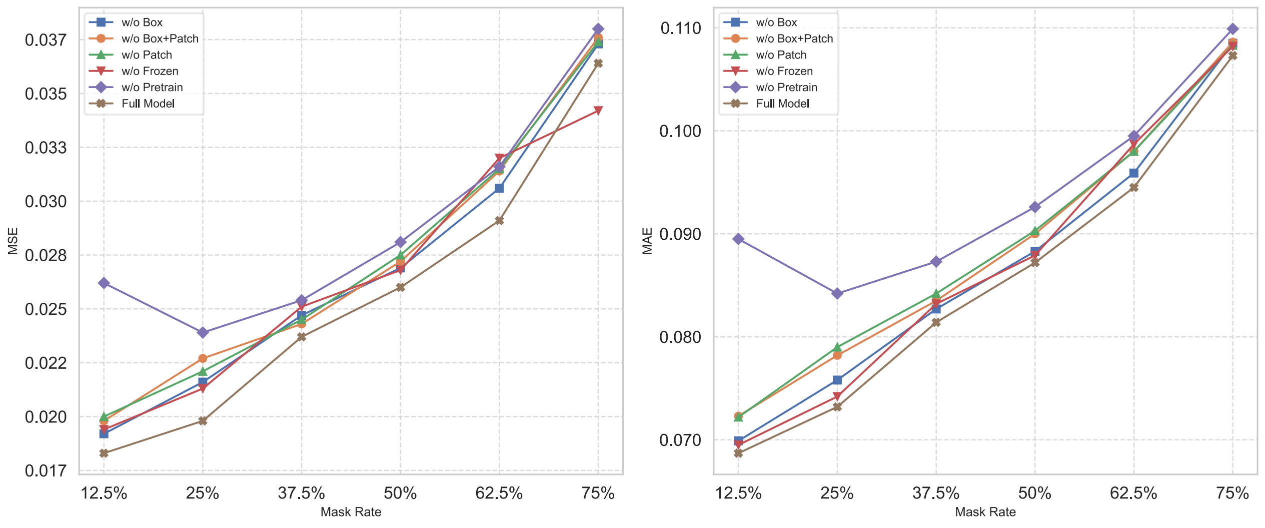

Figure 8: Ablation study

When the freezing mechanism is removed (w/o Frozen) and full fine-tuning is performed on all layers, the model achieves an MSE of 0.0342 and an MAE of 0.1082 at a 75% missing rate, which are slightly higher than those of the Full Model. Full fine-tuning enhances flexibility but also introduces additional computational overhead. The choice between freezing and fine-tuning should be balanced based on task complexity, data volume, and computational resources.

Overall, the Adaptive Patch Partitioning Mechanism plays a crucial role in improving the imputation accuracy of the ImpGPT model under high missing rate conditions. After removing each module, it is found that the freezing mechanism, pretraining strategy, and Adaptive Patch Partitioning all have a significant impact on performance. The Adaptive Patch Partitioning Mechanism provides a new solution for temporal data imputation and enhances the ability to recover missing information.

To further evaluate the impact of different training strategies of the ImpGPT model on computational efficiency, we compared two strategies: freezing part of the GPT-2 weights (Full Model) and full fine-tuning (no_frozen) in Table 4. Freezing part of the GPT-2 weights significantly improves computational efficiency. Under this configuration, the training time per epoch is approximately 23 seconds, which is about 25% faster than the 31 seconds per epoch of full fine-tuning.

Furthermore, the model with frozen weights has lower memory usage: the allocated memory per epoch is 0.35 GB, which is much lower than the 0.84 GB of full fine-tuning. This configuration is highly suitable for environments with limited computational resources, as it enables running more datasets or conducting more frequent experiments within limited memory and time.Freezing weights not only accelerates training speed but also effectively reduces computational overhead under resource-constrained conditions—this makes it more advantageous in rapid experiments and resource-limited environments.

Table 4: Comparison of model training time and computational resources

Method	Training Time (Seconds/Epoch)	Allocated VRAM	Reserved VRAM
Full Model	23s	0.35 GB	2.21 GB
w/o Frozen	31s	0.84 GB	3.13 GB

4.6 Study on model stability

To further evaluate the stability and robustness of the model in missing value imputation tasks, this study conducted independent repeated experiments on the model under 12.5%-75% missing rate settings. In each experiment, the missing positions were regenerated in a completely random manner to ensure that the missing masks were independent of each other across different trials. For the imputation results of each experiment, four evaluation metrics were adopted: Mean Squared Error (MSE), Mean Absolute Error (MAE), Symmetric Mean Absolute Percentage Error (SMAPE), and Normalized Root Mean Squared Error (NRMSE). Based on these metrics, 95% confidence intervals for the model's imputation accuracy under each missing rate were constructed, with the results presented in Table 5.

In contrast, after the missing rate increases,especially under the 62.5% and 75% missing rate scenarios,the confidence intervals of SMAPE and MSE for Informer and ETSformer expand significantly, reflecting their weak adaptability to high-missing scenarios. PatchTST and GPT4TS show moderate stability across most metrics, with relatively concentrated confidence intervals and small performance fluctuations. Although DLinear exhibits a certain degree of stability in MSE and NRMSE, it has large fluctuations in SMAPE, indicating that it still faces challenges in handling nonlinear structures in time series. FEDformer experiences a slight performance decline under medium-

Table 5: 95% Confidence intervals under different missing rates.

Mask Rate	Models	MAE_CI	MSE_CI	NRMSE_CI	SMAPE_CI
12.5%	**Autoformer**	(0.131, 0.162)	(0.039, 0.046)	(0.047, 0.056)	(56.56, 62.83)
	Informer	(0.112, 0.146)	(0.041, 0.049)	(0.049, 0.058)	(53.162, 58.091)
	Reformer	(0.078, 0.096)	(0.019, 0.022)	(0.033, 0.039)	(41.799, 46.427)
	DLinear	(0.079, 0.098)	(0.021, 0.025)	(0.034, 0.041)	(40.893, 45.421)
	FEDformer	(0.094, 0.117)	(0.023, 0.028)	(0.037, 0.043)	(45.785, 50.857)
	ETSformer	(0.109, 0.134)	(0.031, 0.036)	(0.042, 0.050)	(54.319, 60.334)
	GPT4TS	(0.066, 0.072)	(0.016, 0.019)	(0.032, 0.035)	(35.523, 39.437)
	PatchTST	(0.067, 0.073)	(0.018, 0.020)	(0.033, 0.036)	(36.818, 39.984)
	ImpGPT	(0.063, 0.068)	(0.016, 0.017)	(0.032, 0.034)	(35.528, 38.048)
25%	**Autoformer**	(0.127, 0.157)	(0.038, 0.045)	(0.046, 0.055)	(54.313, 60.327)
	Informer	(0.129, 0.168)	(0.056, 0.068)	(0.057, 0.068)	(58.229, 63.516)
	Reformer	(0.088, 0.112)	(0.025, 0.029)	(0.038, 0.044)	(45.353, 50.128)
	DLinear	(0.093, 0.118)	(0.026, 0.032)	(0.039, 0.046)	(45.740, 50.556)
	FEDformer	(0.104, 0.132)	(0.030, 0.037)	(0.042, 0.049)	(51.025, 56.291)
	ETSformer	(0.124, 0.157)	(0.039, 0.048)	(0.048, 0.057)	(60.259, 66.580)
	GPT4TS	(0.076, 0.084)	(0.020, 0.024)	(0.035, 0.039)	(39.238, 44.122)
	PatchTST	(0.069, 0.076)	(0.019, 0.021)	(0.033, 0.037)	(37.915, 41.422)
	ImpGPT	(0.066, 0.070)	(0.018, 0.019)	(0.034, 0.036)	(36.652, 39.017)
37.5%	**Autoformer**	(0.132, 0.163)	(0.044, 0.052)	(0.049, 0.058)	(56.137, 62.3571)
	Informer	(0.136, 0.179)	(0.065, 0.078)	(0.061, 0.073)	(60.329, 65.839)
	Reformer	(0.097, 0.124)	(0.030, 0.036)	(0.042, 0.049)	(47.751, 52.472)
	DLinear	(0.104, 0.132)	(0.031, 0.038)	(0.043, 0.051)	(49.789, 54.799)
	FEDformer	(0.120, 0.152)	(0.039, 0.048)	(0.048, 0.056)	(56.080, 61.983)
	ETSformer	(0.140, 0.178)	(0.051, 0.061)	(0.054, 0.065)	(65.840, 72.664)
	GPT4TS	(0.085, 0.093)	(0.024, 0.028)	(0.038, 0.042)	(42.055, 47.497)
	PatchTST	(0.074, 0.079)	(0.020, 0.024)	(0.035, 0.038)	(39.453, 43.244)
	ImpGPT	(0.071, 0.075)	(0.020, 0.022)	(0.036, 0.038)	(38.054, 40.909)
50%	**Autoformer**	(0.141, 0.174)	(0.047, 0.051)	(0.057, 0.067)	(61.557, 68.379)
	Informer	(0.148, 0.192)	(0.076, 0.092)	(0.066, 0.079)	(63.012, 67.948)
	Reformer	(0.109, 0.138)	(0.039, 0.048)	(0.048, 0.056)	(51.285, 56.420)
	DLinear	(0.116, 0.146)	(0.037, 0.045)	(0.047, 0.055)	(53.591, 58.944)
	FEDformer	(0.143, 0.180)	(0.055, 0.068)	(0.057, 0.067)	(62.339, 68.899)
	ETSformer	(0.157, 0.198)	(0.064, 0.078)	(0.061, 0.073)	(70.597, 78.028)
	GPT4TS	(0.093, 0.101)	(0.029, 0.033)	(0.042, 0.046)	(44.703, 49.868)
	PatchTST	(0.080, 0.085)	(0.024, 0.027)	(0.037, 0.041)	(41.320, 45.265)
	ImpGPT	(0.074, 0.078)	(0.022, 0.024)	(0.037, 0.039)	(39.989, 43.224)
62.5%	**Autoformer**	(0.166, 0.204)	(0.053, 0.063)	(0.065, 0.077)	(70.043, 77.799)
	Informer	(0.161, 0.207)	(0.089, 0.108)	(0.072, 0.085)	(66.425, 72.364)
	Reformer	(0.122, 0.154)	(0.051, 0.063)	(0.055, 0.065)	(55.093, 60.607)
	DLinear	(0.129, 0.164)	(0.045, 0.055)	(0.051, 0.061)	(57.866, 63.664)
	FEDformer	(0.181, 0.230)	(0.082, 0.099)	(0.069, 0.082)	(75.323, 83.049)
	ETSformer	(0.172, 0.218)	(0.079, 0.097)	(0.068, 0.081)	(75.656, 83.619)
	GPT4TS	(0.102, 0.110)	(0.035, 0.039)	(0.046, 0.050)	(47.357, 52.854)
	PatchTST	(0.087, 0.093)	(0.027, 0.032)	(0.040, 0.044)	(43.807, 48.008)
	ImpGPT	(0.083, 0.087)	(0.026, 0.030)	(0.041, 0.043)	(42.336, 46.266)
75%	**Autoformer**	(0.185, 0.228)	(0.059, 0.0705)	(0.074, 0.088)	(73.855, 82.057)
	Informer	(0.178, 0.231)	(0.112, 0.136)	(0.081, 0.096)	(70.408, 75.905)
	Reformer	(0.136, 0.174)	(0.067, 0.083)	(0.063, 0.074)	(58.725, 64.860)
	DLinear	(0.146, 0.185)	(0.055, 0.067)	(0.057, 0.067)	(63.383, 69.922)
	FEDformer	(0.210, 0.268)	(0.109, 0.133)	(0.080, 0.094)	(84.401, 89.496)
	ETSformer	(0.189, 0.240)	(0.098, 0.121)	(0.076, 0.090)	(79.630, 88.012)
	GPT4TS	(0.115, 0.125)	(0.042, 0.048)	(0.050, 0.056)	(51.456, 57.098)
	PatchTST	(0.098, 0.106)	(0.033, 0.038)	(0.045, 0.049)	(47.608, 52.714)
	ImpGPT	(0.094, 0.099)	(0.033, 0.036)	(0.045, 0.048)	(45.953, 50.497)

to-high missing rates, but its overall stability remains acceptable.

In summary, ImpGPT maintains the optimal confidence interval convergence across different missing intensity levels and is the model with the strongest stability performance.

5 Discussion

The ImpGPT model proposed in this study delivers outstanding performance in multivariate time series missing value imputation tasks, particularly maintaining stable recovery accuracy under high missing rate conditions. Compared with the fixed partitioning strategies adopted by GPT4TS and PatchTST, ImpGPT's Adaptive Patch Partitioning Mechanism can dynamically adjust partitioning boundaries based on the local features of the data itself, effectively avoiding the semantic fragmentation issue caused by fixed partitioning. This "semantic-driven" partitioning method ensures the internal integrity of each information unit, laying a solid foundation for subsequent feature extraction.

Furthermore, the Structure-Aware Box Encoding Module introduced in ImpGPT incorporates both the geometric features of each Patch and missing statistics into consideration, making up for the deficiency of traditional methods that overlook local structural information. In high missing rate scenarios, traditional temporal Transformer variants such as FEDformer and ETSformer often perform poorly. The fundamental reason lies in the fact that the attention mechanisms of these models are easily disturbed by a large number of missing values, and their core assumptions based on periodic or trend decomposition are difficult to hold when data is severely missing. Through a two-stage strategy of "local reconstruction followed by global modeling," ImpGPT effectively mitigates the impact of missing values on performance.

Behind its performance advantages, a critical question is whether the model comes at the cost of enormous computational overhead. By comparing it with the classic sliding window method and powerful graph-based methods, it is found that ImpGPT achieves a good balance between computational complexity and scalability.The sliding window method (with a complexity of $O(L\times W\times M)$) is computationally efficient, but its fixed scale W struggles to adapt to multi-pattern changes in sequences. For methods like neural networks, their complexity is usually related to the square of the number of nodes (proportional to $L\times M$); even after sparsification, the growth of their computational overhead with sequence scale is still much higher than that of ImpGPT.By compressing the sequence into a fixed number N of Patches, ImpGPT locks the computational complexity of its core Transformer encoder at the order of $O(N^2 \times d\_model)$, thus making it insensitive to changes in the original sequence length L. Both theoretical analysis and experiments show that when the sequence length increases significantly, the computational overhead of ImpGPT can remain basically stable, exhibiting excellent scalability. Although the model incurs moderate computational costs due to adaptive partitioning and GPT-2 parameters, in industrial scenarios with strict requirements for imputation accuracy, the performance gains fully justify these overheads.

Its potential application in complex control systems is particularly noteworthy. Modern control methods, such as adaptive fuzzy control [30], output feedback projective lag synchronization [31], and robust neural adaptive control [32], rely on the integrity and accuracy of sensor data. As a data recovery module, ImpGPT can restore lost or distorted data, ensuring the robustness of control systems when sensor failures or communication anomalies occur. For instance, in flexible robot control [33], ImpGPT can recover missing joint angle data, ensuring the stability of the adaptive backstepping controller. In nonlinear optimal control [34], recovering complete state sequences is crucial for optimal control. ImpGPT can accurately reconstruct missing state information, ensuring that the optimal control algorithm operates based on real system dynamics, thereby improving control accuracy and system stability.

6 Conclusion

This study proposes ImpGPT, a missing value imputation model for multivariate time series that integrates adaptive Patch partitioning and a generative Transformer. Departing from the traditional fixed-size partitioning paradigm, the adaptive Patch partitioning mechanism dynamically adjusts the size and boundaries of Patches in real time based on the dynamic characteristics of data missing distributions. While ensuring the integrity of the temporal structure, it achieves precise preservation of locally valid information in the neighborhoods of missing points. Combined with a Box encoding module, it further embeds missing pattern features into Patch representations, significantly enhancing the model's adaptability to complex missing scenarios. In the imputation phase, by integrating the pre-trained GPT-2 model with a freezing-fine-tuning strategy, the model maximally retains the language model's advantages in modeling long-range dependencies and demonstrates superior generalization ability and imputation accuracy under high missing rate conditions.

Experimental results confirm that ImpGPT significantly outperforms existing mainstream imputation methods across various missing rate settings. Particularly in extreme scenarios with a missing rate as high as 75%, it still maintains stable and accurate imputation performance. Additionally, parameter sensitivity experiments and ablation studies further verify the critical contributions of the adaptive Patch mechanism, Box structure encoding module, and pre-training strategy to the model's performance. Model stability analysis also indicates that ImpGPT exhibits excellent robustness, capable of maintaining stable convergence of error metrics under different random missing patterns. ImpGPT demonstrates strong performance in terms of accuracy, stability, and adaptability to high missing rates, providing a feasible and efficient solution for missing value imputation in multivariate time series data. In future work, graph neural networks or attention mechanisms can be incorporated to model the correlations between variables, further improving the model's interpretability and practicality.

References

[1] Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1), 6085. https://doi.org/10.1038/s41598-018-24271-9

[2] Baraldi, A., & Enders, C. K. (2010). An introduction to modern missing data analyses. *Journal of School Psychology*, 48(1), 5–37. https://doi.org/10.1016/j.jsp.2009.10.001

[3] Zhou, T., Niu, P., Wang, X., Sun, L., and Jin, R. (2023). One fits all: Power general time series analysis by pretrained LM. *arXiv preprint* arXiv:2302.11939. https://doi.org/10.48550/arXiv.2302.11939

[4] Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B., and Tabona, O. (2021). A survey on missing data in machine learning. *In Journal of Big Data Vol*. 8, No. 1, pp. 140. https://doi.org/10.1186/s40537-021-00516-9

[5] Sharda, S., Singh, M., and Sharma, K. (2021). RSAM: Robust self-attention based multi-horizon model for solar irradiance forecasting. *IEEE Transactions on Sustainable Energy*, 12(2), 1394–1405. https://doi.org/10.1109/TSTE.2020.3014749

[6] Miao, X., Wu, Y., Chen, L., Gao, Y., and Yin, J. (2023). An Experimental Survey of Missing Data Imputation Algorithms. *In IEEE Transactions on Knowledge and Data Engineering* Vol. 35, No. 7, pp. 6630-6650. https://doi.org/10.1109/TKDE.2022.3186498

[7] Tashiro, Y., Song, J., Song, Y., and Ermon, S. (2021). CSDI: Conditional Score - based Diffusion Models for Probabilistic Time Series Imputation. *arXiv preprint* arXiv:2107.03502 . https://doi.org/10.48550/arXiv.2107.03502

[8] Liu, Y., Yu, R., Zheng, S., Zhan, E., and Yue, Y. (2019). NAOMI: Non - Autoregressive Multiresolution Sequence Imputation. *arXiv preprint* arXiv:1901.10946. https://doi.org/10.48550/arXiv.1901.10946

[9] Cini, A., Marisca, I., and Alippi, C. (2022). Filling the G_ap_s: Multivariate time series imputation by graph neural networks. *arXiv preprint* arXiv:2108.00298. https://doi.org/10.48550/arXiv.2108.00298

[10] Oh, E., Kim, T., Ji, Y., and Khaylia, S. (2022). STING: Self-attention-based time-series imputation networks using GAN. *arXiv preprint* arXiv:2209.10801. https://doi.org/10.48550/arXiv.2209.10801

[11] Du, W., Cote, D., and Liu, Y. (2022). SAITS: Self-attention-based imputation for time series. *arXiv preprint* arXiv:2202.08516. https://doi.org/10.48550/arXiv.2202.08516

[12] Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. (2018). BRITS: Bidirectional recurrent imputation for time series. *arXiv preprint* arXiv:1805.10572. https://doi.org/10.48550/arXiv.1805.10572

[13] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *In Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12), 11106–11115. https://doi.org/10.1609/aaai.v35i12.17325

[14] Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint* arXiv:2001.04451. https://doi.org/10.48550/arXiv.2001.04451

[15] Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *In Advances in Neural Information Processing Systems*, 34, 22419–22430. https://doi.org/10.48550/arXiv.2106.13008

[16] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022). FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. *In Proceedings of the 39th International Conference on Machine Learning (ICML).* https://doi.org/10.48550/arXiv.2201.12740

[17] Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. (2022). ETSformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint* arXiv:2202.01381. https://doi.org/10.48550/arXiv.2202.01381

[18] Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting? *In Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9), 11121–11128. https://doi.org/10.1609/aaai.v37i9.26317

[19] Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. *In Proceedings of the International Conference on Learning Representations (ICLR 2023).* arXiv:2211.14730. https://doi.org/10.48550/arXiv.2211.14730

[20] Liu, Y., Zhang, H., Li, C., Huang, X., Wang, J., and Long, M. (2024). Timer: Generative pre-trained transformers are large time series models. *arXiv preprint* arXiv:2402.02368. https://doi.org/10.48550/arXiv.2402.02368

[21] Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. (2023). TimesNet: Temporal 2D-variation modeling for general time series analysis. *arXiv preprint* arXiv:2210.02186. https://doi.org/10.48550/arXiv.2210.02186

[22] Qian, L., Yang, Y., Du, W., Wang, J., and Ibrahim, Z. (2024). Unveiling the Secrets: How Masking Strategies Shape Time Series Imputation. *arXiv*

preprint arXiv:2405.17508. https://doi.org/10.48550/arXiv.2405.17508

[23] Liu, Y., Wu, H., Wang, J., and Long, M. (2022). Non - stationary Transformers: Exploring the Stationarity in Time Series Forecasting. *arXiv preprint* arXiv:2205.14415. https://doi.org/10.48550/arXiv.2205.14415

[24] Bian, Y., Ju, X., Li, J., Xu, Z., Cheng, D., and Xu, Q. (2024). Multi - Patch Prediction: Adapting LLMs for Time Series Representation Learning. *arXiv preprint* arXiv:2402.04852. https://doi.org/10.48550/arXiv.2402.04852

[25] Vaswani A. Attention is all you need(2023). *Advances in Neural Information Processing Systems, NIPS'17: Proceedings of the 31st nternational Conference on Neural Information Processing Systems* Pages 6000 - 6010. https://doi.org/10.48550/arXiv.1706.03762

[26] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. *In International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

[27] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. （2022）. Non-stationary transformers: exploring the stationarity in time series forecasting. *In Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22). Curran Associates Inc., Red Hook, NY, USA,* Article 718, 9881 - 9893.

[28] Ghazanfar, M.A. and Prugel-Bennett, A. 2013. The advantage of careful imputation sources in sparse data-environment of Generating recommender systems: improved SVD-based recommendations. *Informatica (Slovenia)*. 37, 1 (2013), 61–92.

[29] SHI L W. MMF-TSP: A multimodal fusion network for time series prediction using textual and numerical data. *Informatica*, 2025, 49 (24): 67-84. https://doi.org/10.31449/inf.v49i24.7939

[30] Boulkroune A, Zouari F, Boubellouta A. Adaptive fuzzy control for practical fixed-time synchronization of fractional-order chaotic systems. *Journal of Vibration and Control.* 2025;0(0). https://doi.org/10.1177/10775463251320258

[31] Boulkroune A, Hamel S, Zouari F, Boukabou A, Ibeas A. Output-feedback controller based projective lag-synchronization of uncertain chaotic systems in the presence of input nonlinearities. *Mathematical Problems in Engineering.* 2017. https://doi.org/10.1155/2017/8045803

[32] Zouari, Farouk & Ben Saad, Kamel & Benrejeb, Mohamed. (2012). Robust neural adaptive control for a class of uncertain nonlinear complex dynamical multivariable systems. *International Review on Modelling and Simulations.* 5. 2075-2103.

[33] Zouari F, Ben Saad K, Benrejeb M. Adaptive backstepping control for a single-link flexible robot manipulator driven DC motor. 2013 *International Conference on Control, Decision and Information Technologies (CoDIT), Hammamet, Tunisia, 2013,* pp. 864-871. https://doi.org/10.1109/CoDIT.2013.6689656

[34] Rigatos G, Abbaszadeh M, Sari B, Siano P, Cuccurullo G, Zouari F. Nonlinear optimal control for a gas compressor driven by an induction motor. *Results in Control and Optimization.* 2023;11:100226. https://doi.org/10.1016/j.rico.2023.100226