International Journal of Pattern Recognition and Artificial Intelligence
© World Scientific Publishing Company

Generative Artificial Intelligence Model for Multi-granularity Power Data Simulation

Yiwen Jiang
Big Data Center, State Grid Co. of China,
Shanghai, China
1064131401@qq.com

Sheng Xiang
Australian Artificial Intelligence Institute, University of Technology Sydney,
Sydney, Australia
sheng.xiang@uts.edu.au

Yihan Dai
Department of Computer Science and Technology, Tongji University,
Shanghai, China
2053496@tongji.edu.cn

Dawei Cheng
Department of Computer Science and Technology, Tongji University,
Shanghai, China
dcheng@tongji.edu.cn

Electric power resources are essential for the efficient and orderly development of society. Accurate power load forecasting is a key driver for the low-carbon upgrade of power systems. Traditional forecasting methods often struggle to capture long-term dependencies. Additionally, extracting complex nonlinear features from data remains a significant challenge, making it challenging to meet the accuracy demands of modern power systems. Besides, current deep learning-based forecasting methods cannot simulate multi-granularity power load data. To address these challenges, this paper presents a Generative Pre-trained Transformer model, GPT4PLTS, designed for power data simulation and fine-grained power load forecasting. The model leverages the Transformer architecture, incorporating the first six layers of the GPT decoder structure. It utilizes a multi-head attention mechanism to extract temporal features and includes a time alignment layer to maintain the sequence of time-series data, addressing both short-term and long-term dependencies. Extensive experiments are conducted on load observations from 2000 enterprises. The results demonstrate that GPT4PLTS achieves high accuracy in data simulation and forecasts across different time granularities, particularly excelling in short and medium-term predictions. Future research could focus on optimizing the model structure to enhance the model's generalization ability.

Keywords: Generative Artificial Intelligence, Power Load Prediction, Transformer, Time Series Data.

1. Introduction

As economies rapidly develop and urbanization accelerates, the dependence on electricity for residential life, commercial activities, and industrial production continues to grow. Electricity is one of the cleanest, most efficient, safest, and highest-quality renewable energy sources available [9]. Therefore, to build a resource-efficient and environmentally friendly society and to promote sustainable economic and social development, it is essential to optimize electricity supply costs and accurately forecast power load.

Accurate power load forecasting is crucial for effective electricity planning, resource allocation, and efficient management of power systems. Electricity suppliers rely on load forecasts to develop contracting strategies, pricing strategies, trading strategies, and economic assessments for individual users. The precision of these forecasts directly affects pricing; inaccurate predictions can result in significant deviation costs. Precise short- and medium-term load forecasts ensure the safe and stable operation of power systems and serve as the foundation for creating cost-effective generation plans [46]. For electricity consumers, reliable forecasts enable the alignment of production schedules with electricity prices, thereby reducing operational costs [42].

In recent years, extensive research has been conducted to enhance the accuracy of power load forecasting. Traditional methods, such as the Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) models, have been the foundation of early load forecasting techniques. These methods, while useful, often fall short in capturing the nonlinear and non-stationary nature of power load data. For instance, Huang et al.[19] improved the ARMA model by incorporating non-Gaussian processes, and Lee et al.[22] enhanced the ARIMA model using biorthogonal wavelets for spatial decomposition. However, these improvements still struggle with the rapid external disturbances characteristic of power load data. To address these limitations, modern machine learning methods have been employed. Support Vector Machines (SVM)[16] and Artificial Neural Networks (ANN)[37] have shown significant promise. Niu et al.[26] utilized SVM combined with an ant colony algorithm for load forecasting, while Selakov et al.[28] enhanced SVM with particle swarm optimization. In the realm of neural networks, Long Short-Term Memory (LSTM) networks [17] have been particularly effective in handling time series data, as demonstrated by Zheng et al. [50] in their short-term load forecasting model. Cheng et al. [6,23] leverage multi-modality graph neural networks to predict future time series.

Building on the advancements of deep learning, generative models show various applications such as image synthesis [24], text generation [38], and graph simulation [44]. In particular, Transformer architecture, with its attention mechanisms, has revolutionized generative models and time series forecasting. Transformers excel at capturing complex dependencies and have been successfully applied in various forecasting tasks. For example, Du et al.[10] combined LSTM and attention [33,31]

mechanisms for power load forecasting, and Geng et al.[14] used K-Means clustering with LSTM for load prediction. More recently, advanced models like Transformer-GAN [11] have been explored to enhance the detection and prediction of anomalies in power load data.

Despite these advancements, most existing works struggle to simultaneously address the stochastic nature and event-driven impacts of power load time series data, as well as its strong periodic characteristics. Specifically, existing power load forecasting work typically suffers from at least one of the following limitations: 1) Lack of end-to-end power load time series data simulation; 2) Inability to capture the full distribution of power load time series affected by events and user characteristics; 3) Inability to provide accurate power load fluctuation predictions at multiple granularities.

1.1. *Motivations*

To address these limitations on modeling power load time-series data, our research is motivated by the following:

- Generative Pre-trained Transformers (GPTs) have demonstrated remarkable performance in end-to-end modeling of sequential data, particularly in natural language processing tasks. Despite their success, there has been no application of GPTs in the end-to-end simulation of power load time series data. This gap presents an opportunity to leverage the strengths of GPTs in accurately modeling the complexities of power load data, which exhibit both stochastic and cyclical patterns.
- Event features play a crucial role in influencing power load time-series data. Events such as holidays, weather changes, and significant socio-economic activities can cause substantial fluctuations in power demand. Therefore, it is essential to model the complete power load time series pattern by incorporating not only the inherent user characteristics but also the diverse and impactful event features.
- Power load time-series data exhibit different trends and attributes at various granularities, such as hourly, daily, weekly, and monthly levels. These multi-granularity perspectives provide valuable insights into the power load patterns over different time scales. However, existing models often fail to provide simulated power load time series data at multiple granularities. Addressing this issue is crucial for developing a versatile model. Such a model must generate accurate predictions and simulations across diverse time scales.
- The existing methodologies for power load forecasting often require extensive parameter fine-tuning, which can be resource-intensive and time-consuming. There is a need for an efficient and simplified generative approach that can produce accurate power load time series data without extensive fine-tuning processes. Such an approach would significantly re-

duce the computational overhead and improve the usability of the model in dynamic and rapidly changing environments.
- The effectiveness of any power load forecasting model is ultimately judged by its performance in real-world scenarios. Therefore, it is imperative to develop a model that not only excels in controlled experimental settings but also demonstrates robust performance across diverse and unforeseen conditions. This includes the ability to generate accurate and reliable predictions in the presence of anomalies and irregular patterns, which are common in power load data due to various external and internal factors.

1.2. *Contributions*

The main contributions of this paper are summarized as follows:

- We propose an end-to-end power load time series data simulation model based on GPTs. To effectively represent power load data, we model the complete power load time series pattern by incorporating learnable user characteristics and event features into the decoder of the GPTs.
- In both the training and generating stages, we introduce multi-granularity timestamp embeddings in both the encoding and decoding stages to encode and to provide simulated power load time series data at multiple granularities.
- We present an alternative, simplified, fine-tuning-free generative paradigm that enables rapid generation of power load time series data through multi-agent prompt engineering.
- Experimental results on thousands of entities and three popular evaluating metrics demonstrate that our power load time series simulator outperforms the previous best models on multi-granularity generation tasks.
- The results of the case study on power load events injection show that, our model effectively captures the stochastic nature and event-driven impacts of power load time series data while maintaining high prediction accuracy across different time scales.

The remainder of this paper is organized as follows: Section 2 discusses the related works on power load forecasting and generative artificial intelligence. Section 3 presents our proposed method. Section 4 describes our experimental settings. Section 5 details our experimental results. Section 6 provides our conclusion and future directions.

2. Related Works

In this section, the works related with this paper will be discussed in three fields: 1) traditional power load forecasting methods; 2) machine learning-based forecasting methods; and 3) generative artificial intelligence.

2.1. *Traditional Power Load Forecasting*

Traditional load forecasting methods are primarily based on statistical and time series analysis techniques, as power load inherently takes the form of time-series data. These methods rely heavily on historical load data to predict future demand, often overlooking external factors like weather, holidays, and economic changes. Common traditional time series forecasting methods include the Autoregressive Moving Average (ARMA) model and the Autoregressive Integrated Moving Average (ARIMA) model. [19] introduced non-Gaussian processes into short-term time series forecasting by embedding cumulants and bispectra into the ARMA model, thereby enhancing its performance and significantly improving load forecasting accuracy. [22] incorporated lifting schemes into the ARIMA model, using biorthogonal wavelets for spatial decomposition. They split the load series into sub-series based on frequency, predicted these sub-series using the ARIMA model, and then combined them using an inverse lifting scheme, outperforming traditional ARIMA models. [18] also proposed a new Particle Swarm Optimization (PSO) method for identifying the Autoregressive Moving Average with Exogenous Variables (ARMAX) model for hourly load forecasting from one day to one week in advance. The PSO algorithm's ability to converge to the global minimum of complex error surfaces greatly improved the precision of fine-grained time series forecasts. However, these traditional time series forecasting methods are best suited for stationary series. Since power load data is non-stationary and nonlinear, with rapid external disturbances affecting it, time series methods can produce significant errors, highlighting their limitations.

2.2. *Machine Learning-based Forecasting*

Modern machine learning methods have gradually replaced traditional load forecasting techniques [4], becoming mainstream in power load prediction. These methods include Support Vector Machines (SVM) [16] and Artificial Neural Networks (ANN) [37]. SVM is a supervised learning algorithm that uses kernel functions, such as polynomial and radial basis function (RBF) kernels, to map data into a high-dimensional feature space, enabling it to find effective linear decision boundaries in the original space. SVM is widely used in text classification, image recognition, and bioinformatics. [26] used SVM combined with an ant colony algorithm to develop a power load forecasting system, which processed large datasets by removing redundant information, thus reducing SVM training data and overcoming the limitations of large data volumes and slow processing speeds. [28] improved the SVM model with a particle swarm optimization algorithm, adding a module to capture temperature variations, which significantly enhanced the model's learning ability and prediction accuracy. However, the SVM's prediction accuracy heavily relies on the design of its kernel function, making it less adaptable to changes in load time series characteristics and limiting its generalizability in real-world forecasting tasks. Additionally, SVM can be computationally intensive and inefficient with large datasets. Artificial Neural Networks (ANN) operate by adjusting the relationships between numerous

interconnected nodes to process information. Some studies have optimized time series methods using Kalman filtering combined with ANN, leveraging ANN's nonlinear fitting capabilities and Kalman filtering's ability to fit stable stage loads to improve forecasting accuracy [39]. Researchers have also introduced feedforward neural networks for load forecasting in non-linear scenarios [5]. Recurrent Neural Networks (RNN), particularly Long Short-Term Memory (LSTM) [17] networks, are commonly used in time series prediction. LSTM networks effectively address long-term dependency issues, retaining past information to influence future outputs. In recent years, LSTM has been widely applied in short-term load forecasting. [50] utilized LSTM to capture long-term dependencies in time series data for next-day load consumption prediction, achieving better results than traditional time series methods. RNNs can also be combined with other deep learning models to improve prediction accuracy under complex conditions. For instance, hybrid models combining LSTM or Gated Recurrent Unit (GRU) [7] networks with Convolutional Neural Networks (CNN) [21] handle two-dimensional spatiotemporal features, leveraging historical load and meteorological data to capture the nonlinearity, non-stationarity, and temporal characteristics of load data.

Recently, advanced neural network architectures have been introduced to further improve the forecasting accuracy and efficiency of time series models. Temporal Convolutional Networks (TCN) [1] extend traditional convolutional neural networks (CNN) by leveraging dilated convolutions to capture long-range dependencies in time series data. This approach enhances the generalization ability of the model while preserving the sequential order of the data. Similarly, DLinear [48] combines linear models with deep neural networks, using a linear layer to preprocess input data before applying a multi-layer neural network to learn complex nonlinear transformations, enabling better interpretability and performance for sequence modeling. Transformer-based models have also gained traction in load forecasting tasks. Informer [51] incorporates a sparsity constraint in the self-attention [32,30,34] mechanism, allowing it to efficiently handle long sequences with varying lengths. By modeling long-term dependencies with a generative process, Informer demonstrates robust performance in long-term prediction tasks. Crossformer [43], on the other hand, introduces cross-channel attention mechanisms that enhance feature extraction and facilitate learning complex relationships in sequence data while retaining the Transformer's original self-attention mechanism. These innovations allow Crossformer to effectively model both local and global dependencies, further improving forecasting accuracy.

The adoption of these advanced models underscores the shift toward leveraging sophisticated machine learning architectures to address the challenges posed by power load forecasting tasks. By capturing temporal, spatial, and feature-wise dependencies, these methods provide a foundation for developing robust and adaptive forecasting systems.

2.3. *Generative Artificial Intelligence*

On the foundation of Recurrent Neural Networks, the Transformer architecture, centered on the attention mechanism, has revolutionized generative models, significantly enhancing feature processing capabilities and laying the groundwork for the era of large models. GPU-accelerated software and applications [29,41] also boosts the advance of generative artificial intelligence. Generative models with large parameters and robust learning capabilities can handle and understand complex feature relationships. In natural language processing, generative models have demonstrated impressive performance. Due to their flexibility and scalability, these models are being applied to time series forecasting tasks.

The core of the Transformer [38] is the attention mechanism. [10] combined LSTM and attention mechanisms to improve the screening of anomalous data points in power load datasets, enabling automatic power load forecasting. [14] used K-Means clustering and LSTM models to predict power load. [25] developed an LSTM and seq2seq model to extract historical power load data features, deriving load data trends and considering various factors' impacts on the grid for short-term load forecasting. Generative Adversarial Networks (GAN) have also been explored for power load forecasting. [11] proposed an improved Transformer-GAN model to better capture trend features for anomaly detection and identification in load data.

With the ongoing development of large models, researchers are leveraging their powerful representation and generalization capabilities to extract key features from vast datasets and capture complex temporal patterns. By learning and modeling historical load data and meteorological data end-to-end, large models address traditional methods' challenges of high data dimensionality and complex temporal correlations, improving prediction accuracy and stability. This advancement provides more reliable support for power system scheduling and operation. [52] introduced the concept of Freezing Pre-trained Transformers (FPT), retaining the residual blocks' self-attention and feedforward layers in pre-trained language or image models, and fine-tuning them for time series tasks to evaluate predictive performance. [3] improved prediction accuracy by preprocessing data with time alignment and using a two-stage fine-tuning method based on the pre-trained GPT model. [45] explored a novel time series prediction approach by converting numerical sequences into language prompts and leveraging pre-trained language models, showcasing better generalization capabilities. To fully capture time series data characteristics, [36] proposed constructing an embedding layer tailored for time series data before model input, improving representation and prediction accuracy. [47] used instruction fine-tuning based on LLMs for financial time series prediction, maintaining high interpretability and increasing the model's relevance. [35] highlighted the challenges large language models face with numerical or temporal data, suggesting methods like proper time data labeling and multi-modal adapters to improve task performance. [15] proposed encoding time series as numerical strings to enhance LLMs' performance in digit tokenization and uncertainty calibration. [2] introduced

the TEMPO framework, leveraging two inductive biases in time series tasks for effective representation and learning. [20] proposed the Time-LLM framework, combining prompt-as-prefix techniques for robust time series prediction. [49] presented a Transformer-based multi-variate time series representation learning framework, outperforming current best methods across various datasets.

In addition to these pre-trained model-based methods, researchers are developing foundation models trained on extensive time series datasets. These models learn time series representations from large datasets and transfer these representations to downstream tasks. [12] introduced the multi-scale Temporal Transformer (TTM), the first micro pre-trained model (with one million parameters) for effective transfer learning on public TS data. [8] proposed a model pre-trained on a large time series corpus using a patch decoder-style attention model, performing well across different prediction time granularities. [13] presented the TimeGPT model, the first foundation model for time series, capable of generating accurate predictions for unseen datasets, demonstrating excellent zero-shot inference performance. [27] introduced the Lag-Llama model, a general-purpose univariate probabilistic time series forecasting model trained on extensive time series datasets, showing strong zero-shot prediction capabilities for unseen "out-of-distribution" time series data.

Our Approach. Unlike the aforementioned advanced deep learning models such as LSTM, CNN, and Transformer variants, our method employs a GPT-2 decoder architecture enriched with spatiotemporal embeddings to model time series data. This design enables our model to effectively capture intricate temporal dependencies while integrating spatial features, ensuring enhanced forecasting performance in complex real-world scenarios.

3. Proposed Methods

In this section, we will introduce the preliminaries of power load forecasting and our proposed Generative Pre-Trained Transformer for Power Load Time Series (GPT4PLTS).

3.1. *Preliminaries*

Electric power load forecasting is a type of time series prediction that involves analyzing historical time series data to identify development trends and subsequently predict possible load values at future time points or over a certain period. Given a single variable time series $X_t = \{x[0], x[1], ..., x[t]\}$, the input data window size is W, defined as a vector $\mathbf{X}_t = \{x[t-W+1], x[t-W+2], ..., x[t]\}$, where $x[t]$ is the observed value at time t, and W is the window size.

The goal of the forecaster C is to predict the load value within the next T time steps, with the prediction value defined as $\hat{x}_{t+1}, \hat{x}_{t+2}, ..., \hat{x}_{t+T}$. The prediction function can be formulated as:

$$\hat{\mathbf{X}}_{t+1:t+T} = C(\mathbf{X}_t; \theta) \tag{1}$$

Table 1. Notation used in the electric load forecasting model

Notation	Description
X_t	Single variable time series data
$x[t]$	Observed value at time t
W	Input data window size
$\mathbf{X}_t$	Input data vector: $\{x[t-W+1], x[t-W+2], ..., x[t]\}$
T	Number of future time steps for prediction
$\hat{x}_{t+1}, \hat{x}_{t+2}, ..., \hat{x}_{t+T}$	Predicted load values for future time steps
C	Forecaster function
θ	Optimized parameter vector

where θ is the optimized parameter vector. This model needs to effectively capture the long-term dependencies in the sequence data through the use of Transformer's self-attention mechanism and multi-head attention mechanism, allowing for simultaneous focus on different positions within the sequence.

Based on the problem definition, the main challenges and key points in this paper for achieving accurate electric load prediction at different granularities include:

- **Historical Information Mining:** The model needs to fully receive and mine the historical information presented by the sequence data to capture development trends and dependencies.
- **Feature Relationship Extraction:** The model must effectively extract and utilize feature relationships within the data, such as periodicity and other contextual features.
- **Granularity Flexibility:** The model should be capable of providing predictive results at different feature granularities, adapting to various levels of detail in the data.
- **Long-term Dependency Capture:** Utilizing the Transformer's self-attention and multi-head attention mechanisms to dynamically adjust attention weights and capture long-term dependencies in the sequence data.
- **Sequence Understanding:** Unlike traditional recurrent neural networks, the model should process sequences of variable lengths and understand the order within the sequences through positional encoding.

3.2. *Model Architecture*

The proposed model architecture is designed to effectively handle the complexities of power load forecasting by leveraging the strengths of the Transformer framework. The model consists of several key components: the Embedding Layer, Time Alignment Processing, a Pre-Trained Transformer, and the Time Series Data Generator.

Specifically, the embedding layer is responsible for converting the raw input data into a dense vector representation that can be effectively processed by the

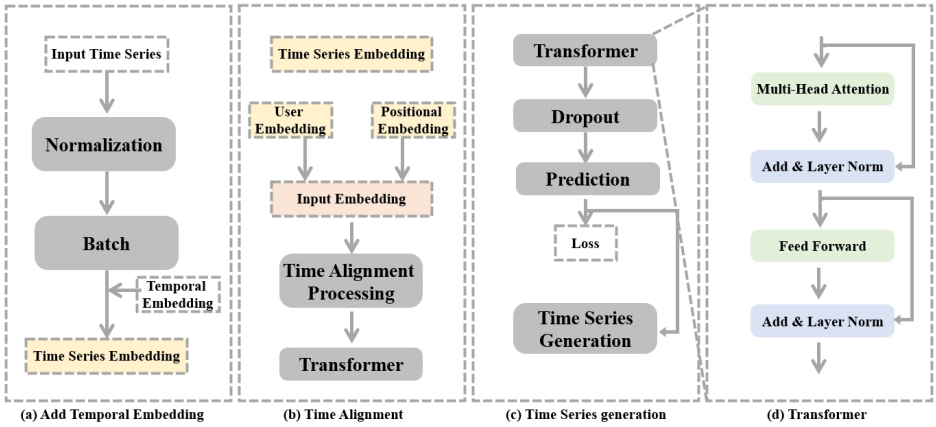

Fig. 1. Model Architecture of the proposed Generative Pre-Trained Transformer for Power Load Time Series (GPT4PLTS), this architecture effectively captures and processes the various aspects of power load data, from initial embedding and alignment to prediction and generation of time series data, resulting in a robust and accurate forecasting model.

model. It incorporates several components to capture different aspects of the input data, including value embedding, temporal embedding, position embedding, and user embedding. Then, the time alignment processing ensures that multiple time series data are aligned to the same time scale for unified data processing. This step involves converting all data timestamps to a consistent format and to estimate the next time series data points start from current timestamps.

The core of the model is a pre-trained Transformer, specifically the decoder of GPT, which is fine-tuned for the task of power load forecasting. Another architecture option is BERT, however, which is not as good as GPT for time series prediction due to its weakness of long tokens [40]. The Transformer architecture, with its self-attention mechanism and multi-head attention, is adept at capturing long-term dependencies and complex relationships within the data. Finally, the Time Series Data Generator transforms the model output back into time series data. This component applies a linear transformation to map the processed model output to the original time series format, ensuring that the predictions are accurately scaled and adjusted.

3.3. *Embedding Layer*

The embedding layer is designed to incorporate several components to capture different aspects of the input data. The overall embedding $\mathbf{e}$ is a combination of four embeddings: value embedding, temporal embedding, position embedding, and user embedding.

$$\mathbf{e} = \mathbf{e}_{\text{value}} + \mathbf{e}_{\text{temporal}} + \mathbf{e}_{\text{position}} + \mathbf{e}_{\text{user}} \tag{2}$$

where $\mathbf{e}_{\text{value}}$ denotes the mbedding representing the observed values of the time series data, $\mathbf{e}_{\text{temporal}}$ denotes the embedding representing the temporal aspects such as month, day, hour, and minute, $\mathbf{e}_{\text{position}}$ is the positional embedding to capture the sequence order, and $\mathbf{e}_{\text{user}}$ is the embedding representing user-specific information. The temporal embedding $\mathbf{e}_{\text{temporal}}$ is a sum of embeddings for different temporal granularities:

$$\mathbf{e}_{\text{temporal}} = \text{embed}_{\text{month}} + \text{embed}_{\text{day}} + \text{embed}_{\text{hour}} + \text{embed}_{\text{minute}} \tag{3}$$

where each embed_* represents the embedding vector for the corresponding time granularity. This allows the model to capture fine-grained temporal features.

The value embedding $\mathbf{e}_{\text{value}}$ is generated using a convolutional token layer to encode the input data vector $\mathbf{X}_t$:

$$\mathbf{e}_{\text{value}} = \text{Conv}_{\text{token}}(\mathbf{X}_t) \tag{4}$$

where $\text{Conv}_{\text{token}}$ is a convolutional layer applied to the input data vector to capture local dependencies and patterns within the input window.

The positional embedding $\mathbf{e}_{\text{pos}}$ is derived using a learnable lookup table E_{pos} based on the position index i:

$$\mathbf{e}_{\text{pos}} = E_{\text{pos}}(i) \tag{5}$$

where E_{pos} maps the position index to a high-dimensional embedding space, allowing the model to understand the order of the sequence.

3.4. *Time Alignment Processing*

To better accomplish the task of time series prediction and capture the temporal information in large amounts of data, this paper proposes a time series alignment phase to match LLMs with time series data. Time alignment processing aligns multiple time series data to the same time scale to facilitate unified data processing operations. First, all data timestamps are converted to the same time format, and interpolation methods are used to estimate data points that do not fall on the unified time scale, ultimately ensuring accurate alignment of data from different sources on the same time dimension.

Given that we chose GPT as the backbone model, which is a causal language model, we ensure that the same autoregressive training method used in its pre-training phase is applied at this stage. Figure 2 illustrates the autoregressive goal of the time series alignment phase: given a series of time series data as the input sequence, the backbone model generates an output sequence that is shifted one step to the right.

3.5. *Pre-Trained Transformer*

The Transformer model, pivotal for capturing the complex dependencies and relationships within time series data, is employed for fine-grained power load forecasting. The Transformer architecture is leveraged due to its capability of dynamically

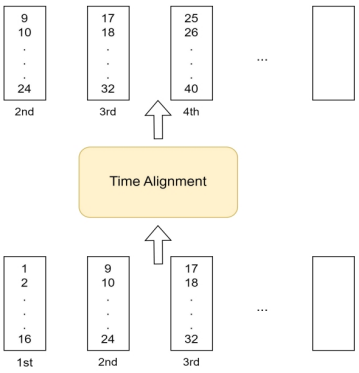

Fig. 2. Time alignment phase for autoregressive training: given an input sequence of time series data, the backbone model generates an output sequence shifted one step to the right.

adjusting attention weights based on the temporal sequence, thereby capturing the temporal dependencies more effectively. This comprehensive approach ensures that the model is well-equipped to handle the intricacies of electric load forecasting, thereby improving prediction accuracy and enhancing the interpretability of the model outcomes. This section details the core components of the Transformer architecture: self-attention mechanism, decoder, and positional encoding.

3.5.1. *Multi-Head Self-Attention*

The self-attention mechanism is at the heart of the Transformer architecture. Traditional neural networks struggle with capturing long-term dependencies and varying input sizes, particularly for sequential data such as power load data, which exhibit significant temporal variations and periodic patterns. The self-attention mechanism addresses these challenges by dynamically focusing on different parts of the input sequence, enabling the model to capture dependencies irrespective of their distance in the sequence.

Given an input sequence, the self-attention mechanism calculates three vectors for each word: Query (Q), Key (K), and Value (V). These vectors are derived from the input embeddings and are used to compute the attention scores. The formula for self-attention is as follows:

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

where Q is the query matrix, K is the key matrix, V is the value matrix, and d_k is the dimension of the key vectors.

To enhance model performance, the Transformer employs multi-head attention, which allows the model to attend to information from different representation sub-

spaces at different positions. The multi-head attention mechanism is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{7}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{8}$$

where head_i is the i-th attention head, W_i^Q, W_i^K, W_i^V are projection matrices for the i-th head, and W^O is the output projection matrix.

Each head in multi-head attention can focus on different parts of the sequence, capturing complex dependencies that are crucial for accurate power load forecasting.

3.5.2. *Transformer Decoder*

The decoder layer is composed of six identical sub-decoders. Similar to the encoder, each sub-layer consists of two main components: multi-head attention mechanism and a feed-forward neural network. Additionally, residual connections are employed around each sub-layer followed by layer normalization. The decoder transforms the contextual embeddings from the encoder into target sequence predictions.

The decoder takes the context vectors generated by the encoder and processes them to produce the output sequence. This is particularly important for power load forecasting, where understanding the sequential nature and dependencies in the data can lead to more accurate predictions.

3.5.3. *Positional Encoding*

To capture the sequential nature of the input data, which is critical for time series like power load data, the Transformer uses positional encoding. Unlike traditional RNNs or CNNs, which inherently consider the order of data, the Transformer relies on positional encodings to provide this information.

The positional encoding is added to the input embeddings at the bottom of the encoder and decoder stacks. The positional encoding vector is computed using sine and cosine functions of different frequencies:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{9}$$

$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{10}$$

where pos is the position, i is the dimension, and d_{model} is the dimension of the model. This encoding allows the model to learn the positional relationships within the sequence data, essential for accurately forecasting power loads over time.

3.5.4. *Pre-Training Optimization Objectives*

The pre-training optimization objectives are designed to ensure that the model learns to capture the underlying patterns and dependencies in power load time series data. During pre-training, the model is optimized to minimize specific loss functions associated with different tasks. The main objectives include:

Next Token Prediction The model is trained to predict the next token in the sequence based on the previous tokens. This objective helps the model learn temporal dependencies within the data. The loss function for the next token prediction is given by:

$$\mathcal{L}_{\text{NTP}} = -\sum_{t=1}^{T} \log P(x_{t+1}|x_{1:t};\theta) \tag{11}$$

where x_{t+1} is the true next token, $x_{1:t}$ are the previous tokens, $P(x_{t+1}|x_{1:t};\theta)$ is the predicted probability of the next token given the previous tokens, and θ represents the model parameters.

Sequence Reconstruction The model is trained to reconstruct the original sequence from a corrupted version. This helps the model learn robust representations that can handle noise and missing data. The loss function for sequence reconstruction is given by:

$$\mathcal{L}_{\text{SR}} = -\sum_{t=1}^{T} \log P(x_t|\tilde{x}_{1:T};\theta) \tag{12}$$

where x_t is the true token at time t, $\tilde{x}_{1:T}$ is the corrupted sequence, and $P(x_t|\tilde{x}_{1:T};\theta)$ is the predicted probability of the token given the corrupted sequence.

Masked Token Prediction In this objective, certain tokens in the input sequence are masked, and the model is trained to predict these masked tokens. This encourages the model to capture contextual information from the surrounding tokens. The loss function for masked token prediction is given by:

$$\mathcal{L}_{\text{MTP}} = -\sum_{t\in\mathcal{M}} \log P(x_t|x_{1:T\setminus\mathcal{M}};\theta) \tag{13}$$

where $\mathcal{M}$ is the set of masked token positions, $x_{1:T\setminus\mathcal{M}}$ are the unmasked tokens, and $P(x_t|x_{1:T\setminus\mathcal{M}};\theta)$ is the predicted probability of the masked token given the unmasked tokens.

Total Loss The total loss function for pre-training is a weighted sum of the individual loss functions:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{NTP}}\mathcal{L}_{\text{NTP}} + \lambda_{\text{SR}}\mathcal{L}_{\text{SR}} + \lambda_{\text{MTP}}\mathcal{L}_{\text{MTP}} \tag{14}$$

where λ_{NTP}, λ_{SR}, and λ_{MTP} are the weights assigned to each loss component.

These pre-training optimization objectives ensure that the model is well-equipped to handle the complexities of power load time series data, capturing both short-term fluctuations and long-term trends.

3.6. *Time Series Data Generator*

To ensure that the model's output is effectively transformed back into time series data, we incorporate a linear output layer. This layer takes the processed model output and maps it back to the original time series format. The transformation process can be described by the following formula:

$$\mathbf{h}_t = \mathbf{W}_{out}\mathbf{o}_t + \mathbf{b}_{out} \tag{15}$$

where $\mathbf{h}_t$ represents the generated time series data at time step t, $\mathbf{W}_{out}$ is the weight matrix of the output layer, $\mathbf{o}_t$ is the output from the model at time step t, and $\mathbf{b}_{out}$ is the bias term. This linear transformation ensures that the predicted values are properly scaled and adjusted to match the expected time series format. By applying this transformation, the model can produce time series data that align with the original input format, enabling integration into the power load forecasting workflow.

3.7. *Fine-Tuning Free Time Series Generation*

Based on the LLMTIME [15], we can leverage existing GPT models and appropriate prompts to achieve event sequence generation without the need for fine-tuning. The main steps are as follows:

- **Define the Time Series Prediction Task**: The task is formulated by treating historical event sequence data as tokens. The goal is to predict the next data point in the sequence, which corresponds to predicting the next token in the GPT model. This involves structuring the time series data in a way that GPT can process, transforming the sequence of events into a format suitable for token prediction.
- **Provide Few-Shot Examples**: A small amount of data is used as examples to illustrate the proper input-output structure. These examples serve as a guide for the GPT model, demonstrating how to map input sequences of historical events to their corresponding next events. The few-shot learning approach allows the model to understand the task with minimal data, leveraging the powerful pre-trained capabilities of GPT.

- **Batch Input for Generation**: The defined input data for the time series generation task is fed into the GPT model in batches. By doing so, we can efficiently generate the required output sequences. Each batch of input data contains sequences for which the next events need to be predicted, and the model processes these sequences to generate the corresponding output results.

This method enables rapid and efficient generation of time series data using GPT models without the need for extensive fine-tuning, making it a practical and scalable approach for various time series prediction tasks.

3.8. *Scalability Analysis for Long-Term Forecasting*

Given the increasing need for handling large-scale, long-term forecasting tasks, we extend our discussion to address the scalability of the proposed approach. The model's time and space complexity, especially for large datasets and high-frequency data, are analyzed as follows:

- **Time Complexity**: The proposed model processes input data in batches, with the complexity of each batch prediction primarily determined by the self-attention mechanism in the GPT model, which scales quadratically with the input sequence length. For long-term forecasting, this can be a limiting factor. To mitigate this, sparse attention mechanisms [51] can be integrated to reduce computational overhead while preserving prediction accuracy.
- **Space Complexity**: The memory requirements grow linearly with the number of parameters in the model and the batch size. For large-scale datasets, reducing the batch size or employing model compression techniques such as quantization or pre-defined knowledge graphs [6] can alleviate the memory footprint without significantly degrading performance.
- **Adaptability for Real-Time Applications**: For real-time forecasting, latency is a critical factor. The use of lightweight model variants, along with hardware acceleration via GPUs or TPUs, can ensure that predictions are generated within acceptable time frames. Additionally, online learning techniques can be incorporated to adapt the model to dynamic changes in real-time data streams.

While the proposed approach provides a strong foundation for scalable forecasting, certain limitations persist. For instance, the quadratic scaling of attention mechanisms restricts its application to extremely long sequences. Future work will explore integrating hierarchical attention models [43] or hybrid frameworks that combine traditional statistical methods with neural architectures to improve scalability further.

4. Experimental Settings

4.1. *Dataset*

4.1.1. *Basic Power Load Dataset*

In this study, the target dataset used comprises load monitoring data from various domains and users between 2021 and 2023. The data is collected from 2000 electricity-using enterprises, with each enterprise being recorded 96 times per day at 15-minute intervals, resulting in a total of 96 load measurements per day. The format of the load monitoring data is shown in Table 2. Additionally, enterprise information data, such as industry type, electricity usage category, etc., is included, with the format shown in Table 3.

Table 2. Part of Enterprise Load Monitoring Data

Enterprise ID	Date	D1	D2	...	D96
UID00001	2021/8/30	175.8	165.45	...	169.8
UID00001	2021/8/31	164.8	154.21	...	145.9
UID00001	2021/9/1	175.5	166.00	...	167.3
UID00001	2021/9/2	184.8	175.67	...	163.2
UID00001	2021/9/3	188.9	182.56	...	168.5

4.1.2. *Power Load Temporal Features*

Electricity load typically exhibits certain periodic regularities and is influenced by multiple factors. Seasonal variations, holidays, special events, industrial and commercial activities, regional and lifestyle habits all impact load patterns. For example, UID00002's daily load data from August 30, 2021, is shown in Figure 3.

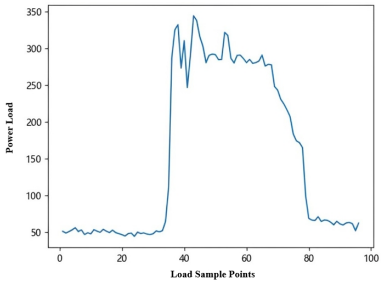

Fig. 3. Daily Load Profile of Enterprise UID00002 on August 30, 2021

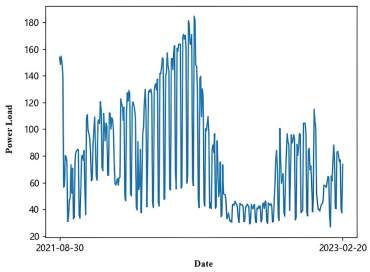

Fig. 4. Load Profile of Enterprise UID00002 from August 30, 2021, to February 20, 2023

Table 3. Part of Enterprise Information Data

Enterprise ID	Contract Capacity	Measurement Capacity	Label	Industry	Region	Usage Category
UID00001	1250	1250	Official	Tertiary Industry	Business	Single Power Source
UID00002	1600	1600	Official	Tertiary Industry	Transport	Dual Power Source
UID00003	3500	1000	Official	Secondary Industry	Equipment Manufacturing	Dual Power Source
UID00004	800	800	Official	Tertiary Industry	Other Services	Single Power Source
UID00005	8000	1260	Official	Secondary Industry	Integrated Circuits	Dual Power Source
UID00006	1260	630	Official	Tertiary Industry	Business	Dual Power Source

The load data of UID00002 from August 30, 2021, to February 20, 2023, reflecting the average daily load, is shown in Figure 4. Due to the impact of holidays and weekends, the curve shows periodic fluctuations. The overall trend indicates an increase starting in June and peaking in mid-August, then gradually decreasing. This trend corresponds to the typical annual electricity consumption cycle affected by heating and cooling demands.

Table 4. Weekly Load Data of Enterprise UID00002 (August 30, 2021 - September 5, 2021)

Enterprise ID	Weekly Avg Load	Weekly Peak Load	Peak Load Date	Min Load Date
UID00001	123.02	154.61	2021/9/1	2021/9/4

4.1.3. *Industry Load Features*

Different enterprises exhibit varying electricity usage types, capacities, and industry characteristics, as shown in Table 3. Figure 5 and Figure 6 illustrate the average daily load profiles of six enterprises from different industries and six enterprises from the transport industry on August 30, 2021, respectively.

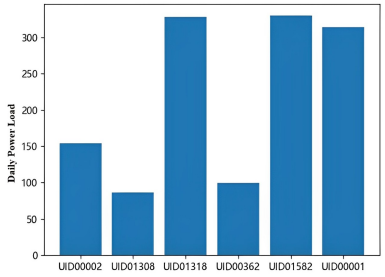

Fig. 5. Average Daily Load Profiles of Six Enterprises from Different Industries (August 30, 2021)

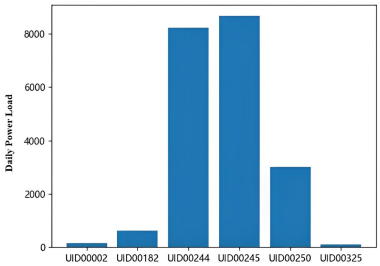

Fig. 6. Average Daily Load Profiles of Six Enterprises from the Transport Industry (August 30, 2021)

4.1.4. *Holiday Information*

Due to people's lifestyles and economic activities, holidays usually significantly impact electricity load. To improve model prediction accuracy, holiday auxiliary information is added to the load data. Table 5 provides holiday information from September 1, 2021, to September 11, 2021.

Table 5. An example of holiday information (September 1, 2021 - September 11, 2021)

Date	Weekday	Holiday (1: Yes, 0: No)
2021/9/27	Monday	0
2021/9/28	Tuesday	0
2021/9/29	Wednesday	0
2021/9/30	Thursday	0
2021/10/1	Friday	1
2021/10/2	Saturday	1
2021/10/3	Sunday	1

4.2. *Data Pre-Process*

4.2.1. *Pre-Process and Normalization*

After analyzing the dataset, we found that the observation dates for enterprises in the load monitoring files from August 30, 2021, to February 20, 2023, are not continuous, with missing dates and NaN values in the load observation data. To address these issues, we used linear interpolation to fill in the missing dates and replaced NaN values with the previous observation point's data. This approach ensures data continuity and accuracy for subsequent analysis and modeling. We also detected outliers in the dataset through statistical analysis. Typically, electricity load is a positive value, as electricity consumption occurs during specific periods.

Therefore, negative load values or unrealistically high values were identified as outliers and removed. Additionally, we found duplicate records in the dataset. These duplicates were merged by taking the average of the duplicate entries for the same enterprise on the same day, ensuring each enterprise-day pair is represented by a single entry.

Due to the diverse electricity demands and usage habits across different industry domains and user types, we processed the electricity load monitoring data separately for each user, normalizing it to a unified scale. This normalization minimizes the impact of data magnitude differences between user groups, enabling more accurate modeling.

The Min-Max normalization method, preserving the original data distribution and structure, was applied to generate train and test data of neural networks. This method maintains the relative relationships and distribution characteristics of the data, as shown in Equation 16:

$$x' = \frac{x - min}{max - min} \tag{16}$$

where, x represents the original data, x' is the normalized data, min is the minimum value, and max is the maximum value of the data column.

4.2.2. *Mini-Batch Data Training*

Mini-Batch training involves dividing the entire training dataset into multiple batches, using each batch to calculate gradients and update model parameters. This method leverages hardware efficiently, accelerates computation, and enhances model training stability by averaging the gradients over batches, reducing variance and smoothing the training process. For this study, we pre-processed and featured the dataset according to batch training requirements. The dataset was divided into batches based on different electricity users, including historical load data and additional user attributes. These attributes include time-related features (e.g., hour, day of the week, holiday) and user basic information (e.g., industry domain). Such a division ensures that each batch accurately reflects real-world electricity usage conditions, providing robust support for model training and generalization. The batch size impacts computation time and result accuracy. In Section 4.4, we discuss the optimal batch size based on our experiments, providing guidance for future model training and setup.

4.3. *Compared Methods*

In this study, we selected six models as baseline methods to compare with our sequence generation model GPT4PLTS. The selected models are LSTM, TCN, GRU, DLinear, Informer, and Crossformer. The following is a brief introduction to these baseline models:

(1) **LSTM** [17]: Long Short Term Memory is a type of recurrent neural network (RNN) used for processing and predicting time series data. It controls information flow through three gates (input gate, forget gate, and output gate) and a cell state, enabling the model to selectively remember or forget information, thus better capturing important features in the sequence data.
(2) **TCN** [1]: Temporal Convolutional Network is an extension of traditional convolutional neural networks (CNN) for sequence modeling. TCN uses one-dimensional convolutions to capture local or global features of the sequence, with dilated convolutions allowing it to learn long-range dependencies, thereby improving the model's generalization ability.
(3) **GRU** [7]: Gated Recurrent Unit is a type of recurrent neural network (RNN) similar to LSTM, but with fewer parameters and a simpler structure, leading to faster training.
(4) **DLinear** [48]: This model combines linear and deep neural networks (DNN), leveraging the strengths of both to handle the complexity of sequence data. The input data is first processed by a linear model, followed by a multi-layer neural network to learn non-linear transformations, thus enhancing the model's understanding and interpretability.
(5) **Informer** [51]: Informer introduces a self-attention mechanism with a sparsity constraint, allowing it to handle sequences of varying lengths efficiently. It also incorporates the impact of long-term dependencies by introducing a generative process, thereby improving the model's robustness and generalization ability in long-term prediction tasks.
(6) **Crossformer** [43]: Compared to traditional Transformer models, Crossformer introduces cross-channel information to enhance feature extraction while retaining the self-attention mechanism of the Transformer, allowing the model to learn long-range dependencies and relationships in the sequence data more effectively.

The training process of these models was cross-validated, with the dataset divided into training and test sets. Each method was trained using batch training as described in the previous section, ensuring consistency and reliability of the results.

4.4. *Parameter Settings*

The experimental environment configuration for this study includes both hardware and software setups. The hardware configuration consists of a HiSilicon Kunpeng-920 CPU operating at 2.6 GHz, a 910B3 GPU, 32GB of memory, and 890GB of storage. The software environment includes VSCode 1.89.1, Conda 4.10.3, Python 3.10.13, and various Python libraries such as PyTorch 2.2.1, and Transformers 4.38.1.

The model training process involved several parameters impacting training outcomes. The optimized experimental configuration parameters are as follows: historical sequence length (seq_len) is 256, prediction length ($pred\_len$) can be 1, 7, 15,

32, or 96, input channels (c_in) is 96, decoder input channels (dec_in) is 96, output channels (c_out) is 96, model dimension (d_model) is 512, learning rate is 0.0007, batch size is 32, dropout rate is 0.3, and the number of training epochs is 15. The frequency of data sequences ($freq$) can be selected from various options including secondly, minutely, hourly, daily, business days, weekly, and monthly, with the time step ($stride$) set to 1. The model uses 6 layers of GPT for training, with the Adam optimizer employed to automatically optimize the learning parameters, adjusting the learning rate dynamically during the training process to enhance model convergence and stability.

4.5. *Evaluating Metrics*

The performance evaluation metrics used in this study for regression error analysis are MSE, MAE, and RMSE. The mathematical definitions of these three metrics are as follows:

MSE (Mean Square Error) is the average of the squared differences between predicted and actual values. The formula is given by Equation 17:

$$Loss_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{17}$$

MAE (Mean Absolute Error) is the average of the absolute differences between predicted and actual values. The formula is given by Equation 18:

$$Loss_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{18}$$

RMSE (Root Mean Square Error) is the square root of the average of the squared differences between predicted and actual values, which is the square root of MSE. The formula is given by Equation 19:

$$Loss_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{19}$$

In the above formulas, y_i represents the actual load data, $\hat{y}_i$ represents the predicted load data, and n denotes the sample size.

During the model training process, to monitor the regression metrics and prevent model overfitting, we employed the Early Stopping mechanism. This technique stops the training when the validation error begins to increase, preventing overfitting and ensuring the model generalizes well to unseen data.

5. Experimental Results

5.1. *Main Results*

The experimental results of the GPT4PLTS model compared to other baseline models are shown in Table 6. The evaluation metrics include MSE, MAE, and RMSE for prediction lengths of 1, 7, 15, 32, and 96 days.

Table 6. Performance Evaluation of GPT4PLTS and Baseline Models

Metric	Length	Ours	LSTM	TCN	GRU	DLinear	Informer	C.former
MSE	1	0.078	0.105	0.078	0.286	0.374	**0.065**	0.081
	7	**0.054**	0.118	0.085	0.305	0.465	0.075	0.085
	15	**0.053**	0.148	0.102	0.436	0.548	0.085	0.093
	32	**0.092**	0.226	0.142	0.602	0.612	0.098	0.129
	96	0.124	0.279	0.245	0.648	0.701	**0.119**	0.155
MAE	1	0.222	0.255	0.225	0.310	0.432	**0.210**	0.230
	7	**0.191**	0.255	0.233	0.310	0.465	0.210	0.225
	15	**0.179**	0.273	0.225	0.320	0.510	0.223	0.237
	32	0.243	0.320	0.276	0.420	0.545	**0.239**	0.287
	96	**0.256**	0.335	0.305	0.435	0.610	0.274	0.325
RMSE	1	0.280	0.324	**0.279**	0.534	0.611	0.255	0.293
	7	**0.230**	0.342	0.295	0.553	0.682	0.269	0.305
	15	**0.231**	0.384	0.313	0.660	0.741	0.273	0.316
	32	0.343	0.444	0.400	0.738	0.812	**0.320**	0.349
	96	**0.353**	0.456	0.435	0.830	0.920	0.354	0.407

From the comparative experiments, it can be observed that our GPT4PLTS model performs well in the short-term prediction of the target dataset, with the prediction performance gradually declining as the prediction horizon increases to one month. Compared to traditional prediction methods such as LSTM, GRU, and TCN, our model consistently exhibits superior performance. The Informer model shows stable prediction results, especially in ultra-short-term and short-term predictions. However, for weekly-level granularity, our GPT4PLTS model achieves lower prediction errors. The main results show that our proposed method achieves 9 best results among 15 settings.

5.2. *Parameter Sensitivity*

During the model training process, the settings of various parameters significantly impact the training results. However, the effects of different parameters on the model vary within a certain range. In Section 4, we presented the parameter settings used to construct the model in this study. These settings were derived from the parameter sensitivity analysis experiments, which provided optimized or optimal

parameter values, enhancing the model's prediction accuracy.

This section introduces some crucial parameters and their specific results from the experiments, exploring the effects of parameter settings on the model's accuracy to further improve prediction precision.

The root mean square error (RMSE) is used as the primary evaluation metric in this study. RMSE is particularly suitable for this research because it emphasizes larger prediction errors, which are often more critical in assessing model performance for electricity load prediction. By penalizing larger deviations more heavily, RMSE provides a clear indication of the model's prediction accuracy, helping to identify configurations that minimize these errors effectively. Moreover, RMSE is widely recognized in similar predictive modeling tasks, ensuring the results of this study are comparable to existing research.

The impact of batch size on model accuracy (RMSE) is shown in Figure 7 (a). The RMSE of the electricity load prediction model fluctuates with changes in batch size. When the batch size is 8 and 16, the RMSE is relatively stable, especially at batch size 16, where the RMSE reaches its lowest value. This indicates that for these two batch sizes, the model's prediction error is relatively small, and the prediction accuracy is optimal. As the batch size increases to 32, 64, and 96, the RMSE rises slightly, but the increase is not significant, maintaining a relatively stable level. Therefore, a batch size of 32 is selected as the optimal value, achieving the best model performance at this value.

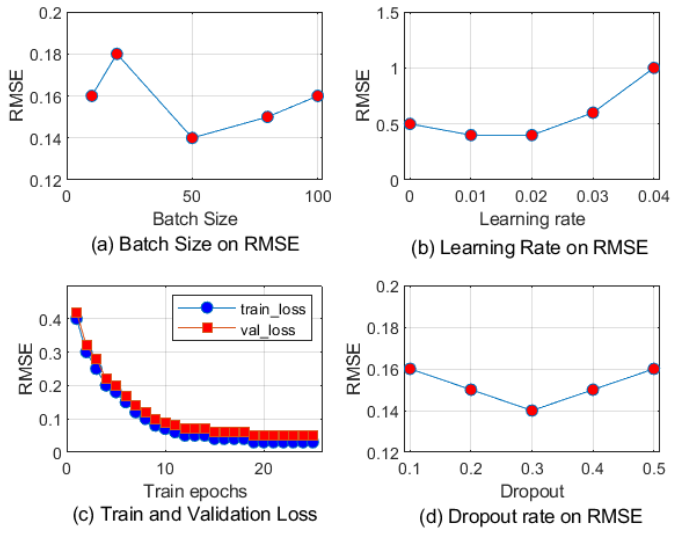

Fig. 7. Parameter Sensitivity Analysis. (a) Impact of Batch Size on RMSE, (b) Relationship between Learning Rate and RMSE, (c) Relationship between Training Epochs and Loss, (d) Impact of Dropout Rate on RMSE

Finding an appropriate learning rate can accelerate model convergence, reducing training time, and avoiding the instability caused by excessively high or low learning rates, thereby enhancing the model's overall performance and accuracy. This study experimented with different learning rates, monitoring the RMSE changes to find the optimal learning rate value, as shown in Figure 7 (b). The optimal learning rate was found to be 0.0007.

The number of training epochs affects the model's ability to learn the information in the data. Too few epochs may result in underfitting, while too many can cause overfitting, reducing the model's generalization ability. This section explores the appropriate number of training epochs, ensuring the training results are optimal. Figure 7 (c) shows the relationship between the number of training epochs and model loss on the training and validation sets. The training loss and validation loss both decrease rapidly during the first few epochs and tend to stabilize around the 10th epoch. The validation loss fluctuates slightly, maintaining a relatively stable level. Therefore, the optimal number of training epochs is set to 15, ensuring model stability and performance.

Dropout is a regularization technique that randomly disables parts of the neurons during training to prevent overfitting, enhancing the model's ability to generalize to unseen data. However, excessive dropout can lead to underfitting, while insufficient dropout may not effectively prevent overfitting. Common dropout rates range between 0.1 and 0.5. This study explores the optimal dropout rate within this range, as shown in Figure 7 (d). When the dropout rate is 0.1, the RMSE is relatively high. As the dropout rate increases to 0.3, the RMSE reaches its lowest value, indicating the model's optimal performance at this dropout rate. With further increases to 0.4 and 0.5, the RMSE rises slightly but remains within a relatively stable range. Therefore, the optimal dropout rate is set to 0.3.

5.3. *Ablation Study*

This section presents an ablation study to verify the rationality and effectiveness of different model components and configurations. We examined the following model variants:

- **GPT(0)**: GPT model with 0 decoder layers.
- **GPT(1)**: GPT model with 1 decoder layer.
- **GPT(6)**: GPT model with 6 decoder layers.
- **GPT(12)**: GPT model with 12 decoder layers.
- **Without Embedding**: Model without any embedding layer.
- **Without Temporal Embedding**: Model with embedding layer but without temporal embedding.
- **Without Position Embedding**: Model with embedding layer but without position embedding.
- **With Temporal Processing**: Model with temporal processing layer.
- **Without Temporal Processing**: Model without temporal processing layer.

The RMSE performance of these model variants on the same test dataset is presented in Table 7.

Table 7. RMSE Performance of Different Model Variants

Model Variant	RMSE
GPT(0)	0.430
GPT(1)	0.350
GPT(6)	0.308
GPT(12)	0.308
Without Embedding	2.355
Without Temporal Embedding	1.266
Without Position Embedding	0.934
With Temporal Processing	0.232
Without Temporal Processing	0.628

The core part of the GPT model is composed of 12 stacked decoder layers. When tuning the prediction model, we explored the impact of different numbers of decoder layers on the training results. As shown in Table 7, compared to the full 12 layers or fewer layers, the 6-layer decoder structure of GPT is a reasonable choice with the best training results.

To verify the effectiveness of the embedding layer design, we trained and evaluated models with and without the embedding layer, comparing their RMSE performance on the same test dataset. Additionally, due to the embedding layer involving multiple feature combinations, we also evaluated models with or without temporal embedding and position embedding to verify the effectiveness and rationality of these embedding methods. The results are shown in Table 7.

Keeping other training parameters unchanged, we compared the training results with and without the temporal processing layer. As shown in Table 7, the RMSE increased significantly after removing the temporal processing layer, indicating a significant decline in model accuracy.

5.4. *Case Study*

During the prediction process of the GPT4PLTS model, we plotted the actual and predicted values of the monitoring data at 96 points for a certain enterprise, with each point representing a 15-minute interval, as shown in Figure 8. It can be seen that although it is difficult for the model to capture peak values, possibly due to the stochastic electricity load volatility in a short period, the generative model performs well overall on a daily basis. Therefore, the model can capture the general trend in short-term power load forecasting.

Additionally, to compare the prediction curves over different time spans, we also plotted the average daily load prediction curves for a certain enterprise over a

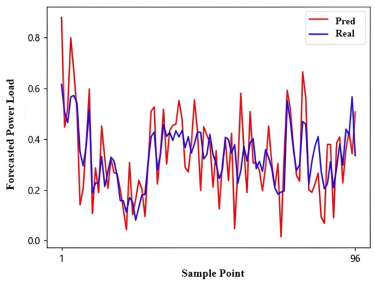

Fig. 8. Actual and Predicted Values of Monitoring Data at 96 Points for a Certain Enterprise

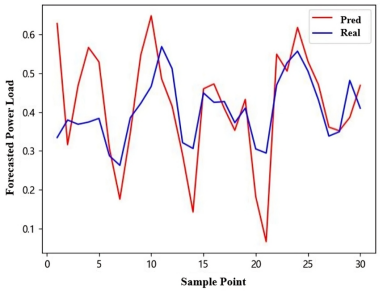

Fig. 9. Average Daily Load Prediction Curves for a Certain Enterprise over a Month (30 days)

month (30 days), as shown in Figure 9. Overall, the model's performance at a 30-day prediction length is better for predictions within a day at 15-minute intervals.

From the case study, we observe that the GPT4PLTS model can effectively capture the general trend of electricity load over short periods. Besides, The model can perform better when predicting average daily loads over a 30-day period, indicating its strength in short-term and medium-term forecasting.

6. Conclusions and Future Works

Constructing a generative power load simulation model can provide various related time data for the power system, facilitating multiple enterprise electricity demand forecasts. In this paper, we proposed a generative model for simulating and forecasting power load using the GPT architecture within the transformer framework. Our approach involved input embeddings, constructing and tuning the generative model, and power load time series generator. Experiments were conducted on multiple baselines and 3 popular evaluating metrics. The results demonstrated that our model outperformed other methods in short to medium-term predictions, especially under large-scale datasets. Future work will focus on incorporating fine-grained features and auxiliary information, such as economic activities and regional events, to improve model accuracy for ultra-short-term and long-term forecasts. Additionally, enhancing the model's scalability and adaptability for different dataset sizes and further optimizing the model's efficiency and speed are key areas for improvement. These efforts aim to refine the model's performance and ensure consistent accuracy across various forecasting scenarios.

References

1. S. Bai, J. Z. Kolter and V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, *arXiv preprint arXiv:1803.01271* (2018).
2. D. Cao, F. Jia, S. O. Arik, T. Pfister, Y. Zheng, W. Ye and Y. Liu, Tempo: Prompt-based generative pre-trained transformer for time series forecasting, *arXiv preprint arXiv:2310.04948* (2023).
3. C. Chang, W.-C. Peng and T.-F. Chen, Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms, *arXiv preprint arXiv:2308.08469* (2023).
4. F. Chen, Y. Zhang, L. Chen, X. Meng, Y. Qi and J. Wang, Dynamic traveling time forecasting based on spatial-temporal graph convolutional networks, *Frontiers of Computer Science* **17**(6) (2023) p. 176615.
5. S.-T. Chen, D. C. Yu and A. R. Moghaddamjo, Weather sensitive short-term load forecasting using nonfully connected artificial neural network, *IEEE Transactions on Power Systems* **7**(3) (1992) 1098–1105.
6. D. Cheng, F. Yang, S. Xiang and J. Liu, Financial time series forecasting with multi-modality graph neural network, *Pattern Recognition* **121** (2022) p. 108218.
7. J. Chung, C. Gulcehre, K. Cho and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, *arXiv preprint arXiv:1412.3555* (2014).
8. A. Das, W. Kong, R. Sen and Y. Zhou, A decoder-only foundation model for time-series forecasting, *arXiv preprint arXiv:2310.10688* (2023).
9. G. Dileep, A survey on smart grid technologies and applications, *Renewable energy* **146** (2020) 2589–2625.
10. J. Du, Y. Cheng, Q. Zhou, J. Zhang, X. Zhang and G. Li, Power load forecasting using bilstm-attention, in *IOP Conference Series: Earth and Environmental Science*, Vol. 440(3) (2020) p. 032115.
11. J. Duan, Deep learning anomaly detection in ai-powered intelligent power distribution systems, *Frontiers in Energy Research* **12** (2024) p. 1364456.
12. V. Ekambaram, A. Jati, N. H. Nguyen, P. Dayama, C. Reddy, W. M. Gifford and J. Kalagnanam, Ttms: Fast multi-level tiny time mixers for improved zero-shot and few-shot forecasting of multivariate time series, *arXiv preprint arXiv:2401.03955* (2024).
13. A. Garza and M. Mergenthaler-Canseco, Timegpt-1, *arXiv preprint arXiv:2310.03589* (2023).
14. D. Geng, H. Zhang and T. Xu, Effective lstm with k-means clustering algorithm for electricity load prediction, in *Proceedings of the 2019 International Conference on Robotics, Intelligent Control and Artificial Intelligence* (2019) pp. 476–481.
15. N. Gruver, M. Finzi, S. Qiu and A. G. Wilson, Large language models are zero-shot time series forecasters, *Advances in Neural Information Processing Systems* **36** (2024).
16. M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, Support vector machines, *IEEE Intelligent Systems and their applications* **13**(4) (1998) 18–28.
17. S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural computation* **9**(8) (1997) 1735–1780.
18. C.-M. Huang, C.-J. Huang and M.-L. Wang, A particle swarm optimization to identifying the armax model for short-term load forecasting, *IEEE Transactions on Power Systems* **20**(2) (2005) 1126–1133.
19. S.-J. Huang and K.-R. Shih, Short-term load forecasting via arma model identification including non-gaussian process considerations, *IEEE Transactions on power systems* **18**(2) (2003) 673–679.
20. 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).
21. A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems* **25** (2012).
22. C.-M. Lee and C.-N. Ko, Short-term load forecasting using lifting scheme and arima models, *Expert Systems with Applications* **38**(5) (2011) 5902–5911.
23. J. Li, Y. Lei, Y. Bian, D. Cheng, Z. Ding and C. Jiang, Ra-cfgpt: Chinese financial assistant with retrieval-augmented large language model, *Frontiers of Computer Science* **18**(5) (2024) p. 185350.
24. M. Li, J. Lin, Y. Ding, Z. Liu, J.-Y. Zhu and S. Han, Gan compression: Efficient architectures for interactive conditional gans, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020) pp. 5284–5294.
25. Y. Mu, M. Wang, X. Zheng and H. Gao, An improved lstm-seq2seq-based forecasting method for electricity load, *Frontiers in Energy Research* **10** (2023) p. 1093667.
26. D. Niu, Y. Wang and D. D. Wu, Power load forecasting using support vector machine and ant colony optimization, *Expert systems with Applications* **37**(3) (2010) 2531–2539.
27. K. Rasul, A. Ashok, A. R. Williams, A. Khorasani, G. Adamopoulos, R. Bhagwatkar, M. Biloš, H. Ghonia, N. V. Hassen, A. Schneider *et al.*, Lag-llama: Towards foundation models for time series forecasting, *arXiv preprint arXiv:2310.08278* (2023).
28. A. Selakov, D. Cvijetinović, L. Milović, S. Mellon and D. Bekut, Hybrid pso–svm method for short-term load forecasting during periods with significant temperature variations in city of burbank, *Applied Soft Computing* **16** (2014) 80–88.
29. X. Shen, X. Liao, L. Zheng, Y. Huang, D. Chen and H. Jin, Archer: a reram-based accelerator for compressed recommendation systems, *Frontiers of Computer Science* **18**(5) (2024) p. 185607.
30. L. Song, S. Chen, Z. Meng, M. Sun and X. Shang, Fmsa-sc: A fine-grained multimodal sentiment analysis dataset based on stock comment videos, *IEEE Transactions on Multimedia* (2024).
31. L. Song, M. He, X. Shang, C. Yang, J. Liu, M. Yu and Y. Lu, A deep cross-modal neural cognitive diagnosis framework for modeling student performance, *Expert Systems with Applications* **230** (2023) p. 120675.
32. L. Song, H. Li, Y. Tan, Z. Li and X. Shang, Enhancing enterprise credit risk assessment with cascaded multi-level graph representation learning, *Neural Networks* **169** (2024) 475–484.
33. L. Song, J. Liu, M. Sun and X. Shang, Weakly supervised group mask network for object detection, *International Journal of Computer Vision* **129**(3) (2021) 681–702.
34. L. Song, X. Shang, R. Zhou, J. Liu, J. Ma, Z. Li and M. Sun, A multi-group multi-stream attribute attention network for fine-grained zero-shot learning, *Neural Networks* **179** (2024) p. 106558.
35. D. Spathis and F. Kawsar, The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language models, *Journal of the American Medical Informatics Association* (2024) p. ocae090.
36. C. Sun, Y. Li, H. Li and S. Hong, Test: Text prototype aligned embedding to activate llm's ability for time series, *arXiv preprint arXiv:2308.08241* (2023).
37. R. E. Uhrig, Introduction to artificial neural networks, in *Proceedings of IECON'95-21st Annual Conference on IEEE Industrial Electronics*, Vol. 1 (1995) pp. 33–37.
38. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, *Advances in neural information processing*

systems **30** (2017).
39. L. C. P. Velasco, D. L. L. Polestico, G. P. O. Macasieb, M. B. V. Reyes and F. B. V. Jr, A hybrid model of autoregressive integrated moving average and artificial neural network for load forecasting, *International Journal of Advanced Computer Science and Applications* **10**(11) (2019) 14–16.
40. J. Wang, W. Zhao, X. Tu and T. He, A novel dense retrieval framework for long document retrieval, *Frontiers of Computer Science* **17**(4) (2023).
41. J. Wang, W. Pang, C. Weng and A. Zhou, D-cubicle: boosting data transfer dynamically for large-scale analytical queries in single-gpu systems, *Frontiers of Computer Science* **17**(4) (2023) p. 174610.
42. M. Wang, Z. Yu, Y. Chen, X. Yang and J. Zhou, Short-term load forecasting considering improved cumulative effect of hourly temperature, *Electric Power Systems Research* **205** (2022) p. 107746.
43. W. Wang, W. Chen, Q. Qiu, L. Chen, B. Wu, B. Lin, X. He and W. Liu, Crossformer++: A versatile vision transformer hinging on cross-scale attention, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
44. S. Xiang, D. Cheng, J. Zhang, Z. Ma, X. Wang and Y. Zhang, Efficient learning-based community-preserving graph generation, in *2022 IEEE 38th International Conference on Data Engineering (ICDE)* (2022) pp. 1982–1994.
45. H. Xue and F. D. Salim, Prompt-based time series forecasting: A new task and dataset, *arXiv preprint arXiv:2210.08964* (2022).
46. Y. Yan-xi, L. Ding, L. Qi and Z. Gang, Short term load forecasting using a multilayer neural network with bp-ga mixed algorithms, *Information and Control* **31**(3) (2002) 284–288.
47. X. Yu, Z. Chen, Y. Ling, S. Dong, Z. Liu and Y. Lu, Temporal data meets llm–explainable financial time series forecasting, *arXiv preprint arXiv:2306.11025* (2023).
48. A. Zeng, M. Chen, L. Zhang and Q. Xu, Are transformers effective for time series forecasting?, in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37(9) (2023) pp. 11121–11128.
49. G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty and C. Eickhoff, A transformer-based framework for multivariate time series representation learning, in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (2021) pp. 2114–2124.
50. J. Zheng, C. Xu, Z. Zhang and X. Li, Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network, in *2017 51st Annual conference on information sciences and systems (CISS)* (2017) pp. 1–6.
51. H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong and W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35(12) (2021) pp. 11106–11115.
52. T. Zhou, P. Niu, L. Sun, R. Jin *et al.*, One fits all: Power general time series analysis by pretrained lm, *Advances in neural information processing systems* **36** (2023) 43322–43355.

Yiwen Jiang is a senior engineer with the Big Data Center of State Grid Corporation of China. She received her doctorate degree from the Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include information security and artificial intelligence.

Yihan Dai is an undergraduate student at Department of Computer Science and Technology, Tongji University, Shanghai, China. She received her bachelor's degree in computer science from Tongji University. Her research interests include applying large language model in data analysis.

Sheng Xiang is a PhD candidate in the Center for Artificial Intelligence, major in Computer Science, University of Technology, Sydney (UTS). He received his BSc degree in Bioinformatics Engineering from Shanghai Jiao Tong University. His research interests include graph machine learning in finance, graph generative algorithm, bipartite graph processing, and dynamic graphs.

Dawei Cheng is an associate professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. Before that, he was a postdoctoral associate at MoE Key Laboratory of Artificial Intelligence, department of computer science, Shanghai Jiao Tong University. He received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China. His research fields include graph learning, big data computing, data mining and machine learning.