EAI Endorsed Transactions
on Energy Web
Research Article EAI.EU

Prompt-TSF: A Power Load Forecasting Model Based on Prompt Learning and Two-Stage Fine-Tuning

Hongxiang Cai[1], Zhiyong Wang[2,*], Yangping Tang[1], Jin Ao[2], Qian Li[2]

[1]State Grid Zhejiang Electric Power Co., Ltd. Ninghai County Power Supply Company, Ninghai, 315600, China
[2]State Grid Blockchain Technology (Beijing) Co., Ltd., Xiongan, 070001, China

Abstract

Accurate power load forecasting is vital for grid stability. Traditional models struggle with complex nonlinearities, while adapting Large Language Models (LLMs) to time-series data faces modal alignment challenges. This paper proposes Prompt-TSF, a framework employing prompt learning and two-stage fine-tuning for multivariate forecasting. We utilize a high-frequency dataset of load and weather variables from Quanzhou, China. A custom tool transforms numerical data into natural language prompts, bridging the modal gap and enriching context. Furthermore, a unique two-stage fine-tuning strategy trains the LLM first on domain knowledge and formatting, followed by specific predictive regression. Experiments demonstrate that Prompt-TSF significantly outperforms traditional forecasting methods.

Keywords: Power Load Forecasting, Large Language Models, Prompt Learning, Two-Stage Fine-Tuning, Time Series Analysis.

Received on 19 September 2025, accepted on 20 December 2025, published on 15 April 2026

doi: 10.4108/ew.12109

1. Introduction

In the context of the "peak carbon and carbon neutrality" strategy, it has become a core national policy to build a new power system mainly based on new energy. The importance of accurate load forecasting as a key technology to support the stable operation of the system and the efficient operation of the market is self-evident. However, the intermittency of large-scale renewable energy sources and the complexity of customer-side loads have posed unprecedented challenges to the accuracy and generalization ability of existing forecasting models [1].

To address these challenges, researchers have proposed a variety of advanced models based on deep learning. For example, Informer [2] and Autoformer [3] dealt with the long sequence dependency problem by designing innovative attention mechanisms; DLinear [4] went the other way around and proved that simple linear models can outperform complex Transformer structures in some scenarios as well; and PatchTST [5] introduced the idea of patching the time series (Patching) idea and achieved excellent performance in several benchmark tests. Despite the excellent performance of these models on benchmark datasets, they still share common limitations: firstly, the knowledge is isolated and the models are completely dependent on the input numerical data, and are unable to utilize the outside world knowledge or the domain textual information for inference; secondly, the models are poorly generalizable, and it is difficult to flexibly migrate or adapt these complex structures, which are designed for a specific task, to the requirements of a new task with a textual description.

Large language models provide a new paradigm to overcome the above limitations with their powerful contextual understanding and reasoning capabilities.

*Corresponding author. Email: 15771341236@163.com

However, applying LLMs to time series forecasting still faces the core obstacles of multimodal disparity and task adaptation. To address this issue, this paper proposes an innovative Prompt-TSF model for power market time-series forecasting based on prompt learning and two-stage fine-tuning. The framework constructs a new and accurate multidimensional power load dataset, which integrates two years of high-frequency power load data and related meteorological data from a city in southern China, and proposes a set of customized cue-word conversion script tools to "textualize" the original multidimensional numerical time-series data into LLMs, and then transforms the original multidimensional numerical time-series data into LLMs. At the same time, we propose a set of customized prompt conversion script tools to "textualize" the raw multidimensional numerical time-series data into natural language prompts that are easy to understand and process by the LLM and rich in contextual information, which provides a high-quality real-world corpus for the study. Based on this, our approach conducts model training through a systematic step-by-step process. In the first stage, we focus on time series data alignment, which familiarizes the LLM with the intrinsic patterns of time through patch-based processing of continuous load series and performing autoregressive supervised fine-tuning. In the second phase, we perform "downstream prediction task fine-tuning", which utilizes the efficient Low Rank Adaptation (LoRA) technique [6] and focuses on improving the model's accurate regression capability on specific prediction tasks by incorporating meteorological data and market-specific application scenarios through Prompt Engineering. The proposed power market time-series forecasting model, based on prompt learning and two-stage fine-tuning, effectively bridges the modal gap between time series data and large language models and realizes the deep integration of domain knowledge and data-driven through efficient fine-tuning and Prompt Engineering, which provides a new and efficient paradigm for the application of LLM in the field of power system forecasting.

2. Related Work

This section aims to provide a comprehensive review of the theoretical foundations and cutting-edge technologies relevant to the research presented herein. To clearly demonstrate the innovation and necessity of our proposed large language model-based power market forecasting framework, this literature review will focus on four core areas: First, we explore the cross-modal knowledge alignment capabilities of large models, which form the theoretical cornerstone for applying LLMs to purely numerical time series tasks. Second, we systematically review common techniques in the field of power load forecasting to clarify the strengths of existing methods and the challenges that remain to be addressed. Next, we delve into efficient fine-tuning and alignment techniques for large language models, which are crucial for adapting pre-trained models to specific domain tasks. Finally, we focus on the emerging research direction of directly applying large language models to time series forecasting, analysing its latest advancements and existing limitations. Through this systematic review of these domains, this chapter will establish a solid theoretical and practical foundation for the subsequent proposed methodology.

2.1. Model Cross-modal Knowledge Alignment Capabilities

In recent years, large language models have made significant breakthroughs in the field of natural language processing (NLP), e.g., models such as GPT-4, PaLM, and ChatGLM have demonstrated near-human level performances in text generation, question answering, and reasoning tasks [7]. However, real-world information covers a wide range of forms such as text, images, audio, video, etc., and is inherently multimodal. How to equip large language models with cross-modal knowledge alignment capability to understand and correlate the semantic information between different modal data has become an important research direction in the field of artificial intelligence. Meanwhile, the importance of cross-modal knowledge alignment is not only reflected in its ability to extend the application scenarios of models (e.g., visual Q&A, audio subtitle generation), but also in its ability to enhance the generalization of the models in complex tasks, such as combining medical images and clinical reports in medical diagnosis, or realizing interactive learning with illustrations and text in the field of education [8].

The core challenge of cross-modal knowledge alignment lies in inter-modal heterogeneity. For example, in a power market environment, loads in a power grid are represented as continuous signals, while linguistic textual data has discrete and symbolic features. Traditional approaches typically rely on hand-designed feature extractors (e.g., SIFT for images, MFCC for audio) and shallow alignment models (e.g., Typical Correlation Analysis CCA), but these methods struggle to capture high-level semantic relationships [9]. In terms of cross-modal representation learning, early work such as DeViSE [10] proposed methods for mapping images and text into a shared semantic space to achieve cross-modal retrieval via linear transformations. With the development of deep learning, neural network-based representation learning techniques (e.g., Transformer architecture) provide new solutions for cross-modal alignment. For example, CLIP (Contrastive Language-Image Pretraining) achieves zero-sample image classification by mapping images and text to a shared embedding space through contrast learning [11]. Subsequently, models such as UNITER [12] and ViLBERT [13] introduced a multimodal Transformer architecture to achieve deep inter-modal fusion through joint training. Similarly, models such as Florence and Kosmos further push the boundaries of cross-modal alignment through generative pre-training (e.g., diffusion model-based image generation) [14]. The success of these

models suggests that the introduction of pre-training techniques significantly improves the performance of cross-modal alignment. A well-chosen pre-trained model, combined with fine-tuning, can be effectively adapted to specific tasks.

2.2. Commonly Used Technologies in the Field of Power Load Forecasting

With the transformation of the global energy structure and the rapid development of the smart grid, the importance of research and application of power system planning and operation technology has gradually come to the fore. Power load forecasting is the core link of power system planning and operation, and its accuracy directly affects the economic scheduling, equipment maintenance and energy resource allocation of the power grid. The traditional power system relies on fossil energy sources, and the load curve is relatively stable and less difficult to forecast; however, with the large-scale grid integration of renewable energy sources (e.g., wind power, photovoltaic), the volatility and uncertainty of the load curve increase significantly. Studies have shown that every 1% reduction in forecasting error can save millions of dollars for the power system [15]. Therefore, the development of highly accurate load forecasting models has become a common focus in both academia and industry.

The techniques used for power load forecasting have evolved from early statistical methods to today's deep learning models, and methodologies have been constantly innovated, but still face challenges such as data heterogeneity and model generalization [16]. Their development can be divided into three stages: traditional statistical methods, machine learning methods, and deep learning methods. Traditional statistical methods such as linear regression and time series models—for instance, the Autoregressive Integrated Moving Average (ARIMA) model—rely on linear assumptions of historical data and are difficult to capture nonlinear features [17]. A simple linear regression model can be expressed as:

$$yt = \beta 0 + \beta 1x1, t+\beta 2x2, t+\cdots+\beta nxn, t+\varepsilon t \quad (1)$$

where yt is the load at time t, β_0 are the coefficients, $x_{i,t}$ are the feature values, and ε_t is the error term. The ARIMA model, incorporating autoregressive and moving average components, is formulated as:

$$\left(1-\sum_{i=1}^{p}\emptyset_i B^i\right)(1-B)^d y_t = c+\left(1+\sum_{j=1}^{q}\theta_j B^j\right)\varepsilon_t \quad (2)$$

where B is the backshift operator, $\emptyset_i$ and θ_j are parameters, p and q are the orders of autoregression and moving average, d is the degree of differencing, and c is a constant.

Machine learning methods (e.g., Support Vector Machine (SVM), Random Forest) improve prediction accuracy through feature engineering, but have limited ability to handle high-dimensional time series data [18]. In recent years, deep learning techniques (e.g., LSTM, Transformer) have become mainstream methods for load forecasting due to their powerful nonlinear modelling capabilities. The core mechanism of an LSTM cell includes:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (5)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (6)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$
$$h_t = o_t * \tanh(C_t) \quad (8)$$

where σ denotes the sigmoid function, and $*$ represents element-wise multiplication.

For example, Google DeepMind reduced the wind power prediction error by 20% using LSTM models. Inspired by the success of large language modeling techniques applied in multiple domains, the Transformer model has also shown significant potential. It relies on the self-attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (9)$$

where Q, K, and V are the query, key, and value matrices, respectively. The Transformer architecture can capture long-range dependencies similarly to LSTMs, while also handling long texts, understanding complex contexts, and supporting multi-task learning, which has a promising future in dealing with data prediction in power system scenarios.

2.3. Fine-tuning for Large Language Models

Model alignment through fine-tuning is the process of adapting a pre-trained model's internal parameters to meet specific task objectives [19]. For example, adapting to specific application scenarios, reducing model errors, mitigating biased outputs, etc [20].

The techniques involved in fine-tuned alignment can usually be categorized into two types: supervised learning and reinforcement learning. Supervised learning uses labelled datasets to retrain pre-trained models. Labelled datasets are usually in text format and contain content that needs to be learned and fine-tuned by the model [21]. In supervised fine-tuning, the objective is typically to minimize a loss function such as cross-entropy over the labelled dataset:

$$L_{SFT}(\theta) = -E_{(x,y)\sim D}\left[\sum_{t=1}^{T}\log p_\theta(y_t|x, y_{<t})\right] \quad (10)$$

where x denotes the input, $y = (y_1, y_2, \cdots, y_T)$ is the target sequence, θ represents the model parameters, and D is the labeled dataset.

Reinforcement learning (RLHF, or "Reinforcement Learning Based on Human Feedback") has a relatively fixed process [22]. First, the model is asked to answer a specific question. The model gives multiple alternative answers as required. Then, human workers are asked to rate and rank the model's output. Finally, the model makes its

output better match expectations by learning human preferences. This process can be formalized as optimizing the following reward function via reinforcement learning (e.g., using Proximal Policy Optimization):

$$\max_{\theta}\left\{\mathbb{E}_{x\sim D,y\sim p_{\theta}(\cdot|x)}\left[r_{\phi}(x,y)\right]-\beta \cdot D_{KL}\left(P_{\theta}(y|x)\,\|\,P_{\theta_{ref}}(y|x)\right)\right\} \quad (11)$$

here, $r_{\emptyset}(x,y)$ denotes the reward model (trained from human feedback), D_{KL} measures the Kullback-Leibler divergence between the fine-tuned policy P_{θ} and a reference pre-trained model $P_{\theta_{ref}}$, and β is a hyperparameter controlling the deviation from the base model.

Comparing these two approaches, it is easy to see that in the prediction scenario of time-series data, supervised learning is more suitable for dealing with textualized labelled data (input-output pairs) and learning the features of the data from this mapping to achieve prediction. The supervised paradigm directly leverages sequential supervisory signals, such as predicting the next time step given previous observations:

$$P(y_t|x_{<t}) = f_{\theta}(x_1, x_2, \cdots, x_{t-1}) \quad (12)$$

where f_{θ} is the predictive model, and the goal is to accurately estimate future values y_t based on historical data $x_{<t}$.

2.4. Time Series Prediction for Large Languages

Currently, time series prediction for large language models is still in its infancy, and by analysing the papers in the field, technical approaches can be broadly categorized into two types based on whether the large model's parameters are frozen. One is not to freeze the internal parameters of the large model, and fine-tune the model parameters for prediction based on the prompted time series [23]; one is to freeze the large model, modify only the input and output layers of the model, and use the cross-attention mechanism to train the input layer parameters to align the time series data with the text data [24]. By comparing these two methods, we find that the method of freezing the big model does not shift the model's attention to the time series data, but transforms the time series data into textual representations with information about the data changes, and the big model does not really understand the changes in the data itself. This approach essentially uses the model as a mere output generator, leveraging only its external interface. In order to shift the model's attention to the time-series data, it is still necessary to fine-tune the parameters inside the model.

3. Methodology

To address the multimodal disparity and task adaptation challenges of applying LLMs to power load forecasting, we propose a systematic two-stage adaptive fine-tuning framework. As shown in Figure 1, the core idea of the framework is to first force the LLM to learn the intrinsic representations of time series through an autoregressive generative task, and then focus its capability on accurate forecasting through a regression task incorporating domain knowledge. The framework consists of a first phase: supervised generative fine-tuning for Representation Alignment and a second phase: downstream task fine-tuning for Regression Forecasting.

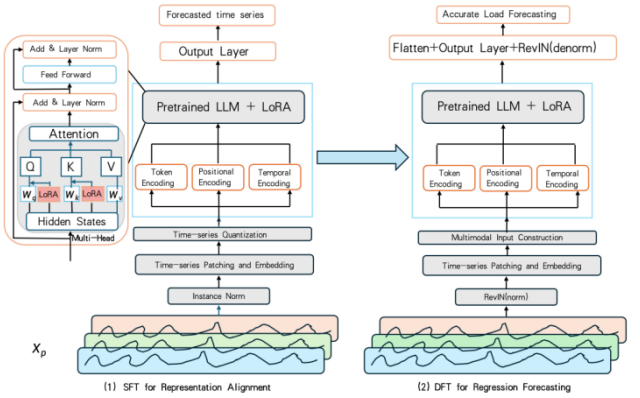

Figure 1. Systematic two-stage adaptive fine-tuning framework

3.1. Phase 1: Supervised Generative Fine-Tuning for Representation Alignment

The goal of this phase is to reshape LLM from a natural language processor to a time series pattern comprehender. We design a "time-series patch to discrete codebook token" generation task and utilize the SFT paradigm to train the LLM to learn the basic distributional and dynamic properties of time-series data.

3.1.1. Time-series Patching and Embedding

The power load sequence is not irregular random noise, but consists of a large number of pattern units with local semantics. Dividing the sequence into patches is equivalent to discretizing these continuous pattern units, so that their structure is similar to the "vocabulary" or "phrases" in natural language. This allows LLM to utilize its powerful ability to process discrete token sequences to capture these local dynamic patterns with real meaning, rather than processing seemingly meaningless floating-point numbers individually. Consider a univariate power load time series processed by reversible instance normalization (RevIN) [25]. $x = \{x_1, x_2, \ldots, x_T\} \in \mathbb{R}^T$, where T is the total length of the sequence. We first partition it into N patches (patches) [5] of length P and step size S, constituting the patch sequence. $P = (p_1, \ldots, p_N)$, which $p_i \in R^P$. To input these numerical patches into the embedding space of the LLM, we use a

learnable linear projection layer $W_{proj} \in R^{P \times d_{model}}$. Map each patch to a hidden dimension of the model:

$$e_i = \text{Linear}(p_i; W_{proj}) = p_i W_{proj} \quad (13)$$

In order to preserve the crucial timing information, we introduce a learnable positional encoding $PE \in R^{N_{max} \times d_{model}}$. The final patch input h_i to the LLM is represented as:

$$h_i = e_i + PE_i \quad (14)$$

As a result, the raw timing data is converted into an embedded sequence suitable for LLM processing.

3.1.2. Time-series Quantization and Discretization

In order to unify the fine-tuning tasks in this phase under the generative paradigm of LLM (i.e., predicting discrete tokens), we introduced the Vector Quantization (VQ) technique [26]. We constructed a learnable codebook $C = \{c_1, \ldots, c_V\} \subset R^P$, which contains V code vectors (codewords), each having the same dimension as the patches. For any time series patch, the quantization process can be defined as:

$$Q(p_i) = \arg \min_{k \in \{1,\ldots,V\}} || p_i - c_k ||_2^2 \quad (15)$$

This operation returns the index in the codebook (i.e., token ID) of the code vector with the closest Euclidean distance to the patch p_i. With this operation, a sequence of consecutive patches is converted into a sequence of discrete token IDs $Z = (z_1, \ldots, z_N)$, which $z_i = Q(p_i)$.

3.1.3. Supervised Fine-Tuning with Causal Language Modeling Objective

We formulate the first stage of SFT as a Causal Language Modeling (CLM) task [27]. Given a sequence of historical patches $(p_1, \ldots, p_L)$with a context window of length L, the goal of the model is to predict the discrete token ID z_{L+1} corresponding to the next patch p_{L+1}.

θ is set as a trainable parameter in the LLM. The model M_θ receives as input the embedded sequence $H_{1:L} = (h_1, \ldots, h_L)$ of historical patches and outputs a probability distribution over the entire discrete vocabulary (V codebook tokens). For the j th possible token, its predicted probability is:

$$P(z_t = j | H_{1:L}; \theta) = \text{Softmax}(\text{Linear}(M_\theta(H_{1:L}))_j) \quad (16)$$

where $M_\theta(H_{1:L})$ is the output logits of the model at time step t.

The training objective in this phase is to minimize the negative log-likelihood, i.e., the standard Cross-Entropy Loss (CEL). For a dataset D_{SFT1} containing M training samples, the total loss function is:

$$L_{SFT1}(\theta) = -\sum_{i=1}^{M} \sum_{t=L+1}^{N_i} \log P(z_t^{(i)} | H_{t-L:t-1}^{(i)}; \theta) \quad (17)$$

The model learns the conditional probability distribution $P(p_t | p_{t-L}, \ldots, p_{t-1})$ of the time series t by minimizing this loss, thus implicitly capturing its complex dynamics.

3.2. Phase 2: Downstream Task Fine-tuning Towards Regression Prediction

After the model has acquired a basic understanding of time series, the goal of this phase is to focus its capabilities on high-precision, multivariate regression prediction tasks and to integrate knowledge related to the power domain to improve model precision and generalization.

3.2.1 Multimodal Input Construction via Prompt Engineering

The ability of LLMs to understand and generate language is highly dependent on input design, specifically prompt engineering. Prompt serves as a bridge between users and models and directly determines the quality and relevance of the model output. For example, OpenAI's GPT series of models can achieve zero or few-sample learning without additional fine-tuning through well-designed Prompts [28]. Prompt design often relies on human expertise, such as using templates like "Question: Q Answer:" to guide the model in Q&A tasks. During the initial pre-training phase, LLMs are trained on large amounts of textual data to learn how to generate coherent sentences, understand the context, recognize linguistic patterns, and so on. The training phase usually uses massive unlabelled data, which can be books, articles, web content, etc. After the base model has been trained, fine-tuning is the process of further training the model on a specific task or domain. Fine-tuning usually uses smaller, labelled datasets to enable the model to perform better on specific tasks (e.g., sentiment analysis, translation, Q&A, etc.). Thus, clear instructions and task-specific prompt design ensure that the model understands the requirements and generates high-quality output that meets the requirements based on the given task needs and context, reducing unnecessary training complexity.

Converting power load time-series prediction into a language generation task is crucial for applying LLMs to data prediction. This addresses the modal gap between continuous time-series data in power load forecasting and the discrete token space inherent to LLMs. Discrete Prompts take the form of readable text (e.g., "Please summarize the following article: text") and leverage natural language interpretability [29]. This discrepancy makes LLM unable to accurately perform disambiguation operations on temporal data. For this reason, this experiment transforms the raw time-series dataset of power loads into textual descriptions.

We construct the forecasting task as a context-rich question-and-answer format. In addition to the historical load series x_{hist}, we also introduce a contemporaneous multidimensional sequence $W_{hist} = \{w1, \ldots, wL\}$ of meteorological features, where w_t is a vector of meteorological features at time step t. The query component provides essential information for the forecasting task. Meanwhile, we design a structured

Prompt Template to textualize and encapsulate all the information:

$$\text{Input Text} = T_{\text{prompt}}(\text{Context}, \text{Task}, W_{\text{hist}}) \quad (18)$$

This format seamlessly integrates numerical data, task instructions, and domain context, allowing LLM to reason in richer contexts. Table 1 illustrates the template used to transform the time-series data prediction task into a text generation task. The template can be roughly structured into two parts: input prompts and output prompts. The input prompt contains both instructions and input parts. The instruction part is used to create the linguistic environment of the power market for the large model, to distract the attention of the large model, and to prompt the large model that the current time series data prediction task needs to refer to the relevant information of the power market. In this experiment, the instruction part is the same text in all the data. The input section details the external environment and power load conditions during the time period t_0 and specifies the task objective: predicting the power load for the subsequent 10 time periods. It contains information such as historical power load, minimum temperature, maximum temperature, average temperature, humidity, rainfall, etc. The output section provides the predicted grid load for the next ten time periods according to the task requirements.

3.2.2. Parameter-Efficient Fine-Tuning via LoRA

In real power system applications, specialized prediction models need to be trained for different regions, seasons, or user types. The cost would be huge if full-parameter fine-tuning is performed every time. The incorporation of LoRA enhances the framework's flexibility and scalability. Therefore, we employ the LoRA technique [6] for computationally efficient fine-tuning. By introducing low-rank matrices at each Transformer layer to adapt to downstream tasks, rather than updating all weights, the number of trainable parameters is significantly reduced, greatly improving computational efficiency and reducing overhead.

For any of Transformer's pre-trained weight matrices $W_0 \in R^{d\times k}$ in LLM, we introduce two low-rank trainable matrices $A \in R^{d\times r}$ and $B \in R^{r\times k}$, where rank $r \ll \min(d, k)$. In forward propagation, the operations on the inputs x_{in} are modified as follows:

$$h_{out} = W_0 x_{in} + \alpha \frac{1}{r} BAx_{in} \quad (19)$$

where α is a scaling constant. During training, the huge matrix W_0 is frozen and only the parameters of the low-rank matrices A and B are updated. This allows us to achieve comparable or even better performance than full parameter fine-tuning with only a very small number of parameters, which is well-suited for power load forecasting task scenarios.

3.2.3. Regression Prediction and Loss Function

In practical power system operations, large prediction deviations can cause grid frequency instability or even large-scale blackouts, which are far more critical than minor prediction errors. Therefore, we select MSE as the regression loss function. We take the final hidden state vector $h_{final} \in R^{d_{model}}$ obtained after processing the cue text by LLM and directly map it to the predicted load values $\hat{y} \in R^H$ for the next H time steps by a simple linear regression header $W_{reg} \in R^{d_{model}\times H}$:

$$\hat{y} = h_{final} W_{reg} \quad (20)$$

The objective of fine-tuning in this phase is to minimize the Mean Squared Error (MSE) between the forecasted value $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_H)$ and the true future load value $y = (y_1, \ldots, y_H)$. The loss function is defined as:

$$L_{DFT}(\theta_{LoRA}, \theta_{reg}) = \frac{1}{H}\sum_{i=1}^{H}(\hat{y}_i - y_i)^2 = \frac{1}{H}\|\hat{y} - y\|_2^2 \quad (21)$$

where θ_{LoRA} and θ_{reg} represent the trainable parameters of the LoRA adapter and regression header, respectively. In the inference phase, the predicted values $\hat{y}$ are restored to the original data scale through the inverse process of RevIN for better fine-tuning.

Table 1. Prompt templates for time series prediction

	Category	Template	Example
Input Prompt	Instruction	You are an expert in power data prediction. Here is a description of the power load. Please predict the power load for the next [period] based on the comprehensive data of the previous [period].	You are an expert in power data prediction. Here is a description of the power load. Please predict the power load for the next 10 time points based on the comprehensive data of the previous 10 time points.
	Input	The current time period is from [start time] to [end time]. The minimum temperature is [min temp] degrees, the maximum temperature is [max temp] degrees, the average temperature is [avg temp] degrees, the relative	The current time period is from 19:00 on January 2, 2016 to 4:00 on January 3, 2016. The minimum temperature is 20 degrees, the maximum temperature is 13 degrees, the average temperature is 16 degrees, the relative humidity is 59%, the rainfall is 0 mm, and the

		humidity is [humidity]%, the rainfall is [rainfall] mm, and the power load is [load values]. Please predict the power loads for the next 10 time periods.	power load is 6180.0, 6100.0, 5916.0, 5418.0, 4622.0, 3998.0, 3738.0, 3583.0, 3484.0, 3426.0. Please predict the power loads for the next 10 time periods.
Output Prompt	Output	The predicted power load for the next 10 periods is [predicted values].	The predicted power load for the next 10 periods is 3442.0, 3593.0, 4188.0, 6255.0, 7027.0, 7299.0, 7178.0, 5557.0, 5986.0, and 6994.0.

4. Experiment

This section is divided into two main parts. First, we detail the experimental settings and implementation, including the pre-processing of data, construction of prompts, evaluation metrics, baselines, the fine-tuning environment construction of LLM based on LLaMA-Factory, and the setting of fine-tuning and training parameters. Secondly, we demonstrate the superiority of our method through comparisons between LLMs and traditional models.

4.1 The Settings and Implementation of the Experiment

Data pre-processing. Data were collected from a city power grid in southern China (Quanzhou), which includes two years of power load, temperature, humidity, rainfall, and other information. Part of the data is shown in Table 2. 10,5000 data points were collected at 15-minute intervals. As shown in Table 3, in order to overcome over-fitting and to improve the generalization ability and stability of the model, we take the average of 4 data within one hour and make the data frequencies change to 1 piece per hour. This processing not only reduces training time but also enables the model to learn from a more appropriate temporal scale, thereby improving prediction performance.

Table 2. Power load datasets for a certain place in the south China from 2016 to 2018, with one data point every 15 minutes, including temperature, humidity, rainfall, and other information

Times	Power Grid Load	Maximum Temperature (℃)	Minimum Temperature (℃)	Average Temperature (℃)	Relative Humidity	Rainfall (mm)
2016/1/1 0:00	3967.259968	19.5	12.1	15.8	63	0
2016/1/1 0:15	3859.196416	19.5	12.1	15.8	63	0
2016/1/1 0:30	3759.877696	19.5	12.1	15.8	63	0
2016/1/1 0:45	3669.973024	19.5	12.1	15.8	63	0
2016/1/1 1:00	3563.567968	19.5	12.1	15.8	63	0
2016/1/1 1:15	3497.436736	19.5	12.1	15.8	63	0
2016/1/1 1:30	3409.991392	19.5	12.1	15.8	63	0

Table 3. The comparison between data before and after process.

	Collection Period	Collection Frequency	Training Set	Validation Set	Test Set	Total
Original Data	From 2016/01/01 To 2018/12/31	1 piece/15 mins	From 2016/01/01 To 2018/02/06 73500 Pieces	From 2018/02/06 To 2018/09/10 21000 Pieces	From 2018/09/10 To 2018/12/31 10500 Pieces	105000 Pieces
Data after Process	From 2016/01/01 To 2018/12/31	1 piece/hour	From 2016/01/01 To 2018/02/06 18375 Pieces	From 2018/02/06 To 2018/09/10 5250 Pieces	From 2018/09/10 To 2018/12/31 2625 Pieces	26250 Pieces

Construction of prompts. Fine-tuning an LLM requires constructing textual prompts. The basic prompt structure includes task description, context, query, and response. Based on the power load forecasting requirements and the prompt structure described in the Methodology section, we constructed prompts for redirecting LLM attention and downstream fine-tuning tasks, as shown in Tables 4 and 5.

Table 4. The prompt used to redirect the LLM's attention

Input	Output
3814.0 3454.0 3208.0 3050.0 2932.0 2853.0 2792.0 2718.0 2786.0 3040.0	3242.0 3335.0 3063.0 2856.0 2813.0 2860.0 3036.0 3310.0 3771.0 3739.0

Table 5. The prompt used for downstream fine-tuning tasks.

Instruction	You are an expert in power data prediction. Here is a description of the power load. Please predict the power load for the next 10 time points based on the comprehensive data of the previous 10 time points.
Input	The current time period is from 19:00 on January 2, 2016 to 4:00 on January 3, 2016. The minimum temperature is 20 degrees, the maximum temperature is 13 degrees, the average temperature is 16 degrees, the relative humidity is 59%, the rainfall is 0 mm, and the power load is 6180.0, 6100.0, 5916.0, 5418.0, 4622.0, 3998.0, 3738.0, 3583.0, 3484.0, 3426.0. Please predict the power loads for the next 10 time periods.
Output	The predicted power load for the next 10 periods is 3442.0, 3593.0, 4188.0, 6255.0, 7027.0, 7299.0, 7178.0, 5557.0, 5986.0, and 6994.0.

In our experiment, we set the sliding window size to 10. Given the power load from t_0 to t_9, we asked the large model to predict the load from t_{10} to t_{19}. The data used to redirect the LLM's attention consisted of 10,000 data points, divided into training, validation, and test sets in a 7:2:1 ratio. The data used to fine-tune the pre-trained LLM for downstream tasks consisted of 30,000 data points, also split into a training, validation, and test set ratio of 7:2:1.

Evaluation metrics. Three evaluation criteria are used in this experiment to evaluate the prediction results of the power load forecasting model: mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE), as shown in Equations (22)-(24).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}| y_i - \hat{y}_i | \tag{22}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}| \frac{y_i - \hat{y}_i}{y_i} |\times 100\% \tag{23}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{24}$$

Baselines. To establish an accurate, reliable, and convincing benchmark for LLMs, we select traditional forecasting models commonly used in the field of power market forecasting. These models include the AutoARIMA model and the Long Short-Term Memory (LSTM).

AutoARIMA is an automated time series forecasting method that automatically selects the optimal parameter configuration to fit given time series data [30]. The ARIMA model consists of three main components: AutoRegressive (AR), Integrated (I), and Moving Average (MA) [31]. AR predicts current values based on past values. I makes non-stationary time series stationary. MA predicts current values based on past forecast errors [32]. When fitting training data, AutoARIMA explores different combinations of AR, I, and MA parameters, including seasonal components. In our experiment, the period was set to 24 due to the daily periodicity of power load. The optimal parameters are selected based on the training results of historical data [33]. AutoARIMA then evaluates the model's performance using error metrics (such as AIC, BIC, and RMSE) and performs cross-validation. Finally, AutoARIMA uses the trained parameters to forecast future time series data [34].

An LSTM network is a special type of Recurrent Neural Network (RNN) proposed by Hochreiter and Schmidhuber in 1997 [35]. It aims to address the vanishing and exploding gradient problems of traditional RNNs during long-sequence training [36]. By introducing gating mechanisms (input gate, forget gate, output gate) and memory cells [37], LSTM can effectively learn long-term dependencies and has achieved excellent performance in fields such as time series prediction [38], NLP and speech recognition [39].

Meanwhile, to visually demonstrate the numerical distribution of the power load dataset and compare the performance differences between different models, the experiment also considered some simple prediction methods, such as using the values from the Previous Hour (P1H) as the prediction value, using the Average load value for the Previous 24 Hours (AP24H) as the prediction value, and using the load data from the Same Time Yesterday (STY) as the prediction value. By comparing with these baselines, we can more clearly demonstrate the advantages and improvements of LLMs in power load forecasting.

Implementation of the LLM Fine-Tuning Environment Using LLaMA-Factory

LLaMA-Factory is an efficient LLM fine-tuning framework that simplifies the fine-tuning and deployment of LLM (such as LLaMA and GPT). The following Table 6 lists the packages used to build the LLM fine-tuning environment in LLAMA-Factory.

Table 6. The packages used to build the LLM fine-tuning environment in LLAMA-Factory

Environment Name	Required Version	Selected Version
transformers	>=4.41.2,<=4.48.3	4.41.2
datasets	>=2.16.0,<=3.2.0	3.2.0
accelerate	>=0.34.0,<=1.2.1	1.2.1
peft	>=0.11.1,<=0.12.0	0.12.0
trl	>=0.8.6,<=0.9.6	0.9.6
tokenizers	>=0.19.0,<=0.21.0	0.19.1
gradio	>=4.38.0,<=5.12.0	5.12.0
pandas	>=2.0.0	2.0.0
matplotlib	>=3.7.0	3.10.0
numpy	<2.0.0	1.26.4
tyro	<0.9.0	0.8.14
Scipy, einops, sentencepiece, tiktoken, protobuf, uvicorn,		
pydantic, fastapi, sse-starlette, fire, packaging, Pyyaml, av, librosa	Not limited	Latest commanded version

Fine-tuning parameter settings and training. We conducted training using the web interface provided by LLaMA-Factory. Considering the complexity of the power load prediction, we employed the ChatGLM3-6b model locally. Fine-tuning was performed using the LoRA method with supervised learning. Key parameters included: AdamW optimizer with initial learning rate 5e-5, 2.0 training epochs, gradient clipping norm 1.0, maximum samples per dataset 100,000, and default values for other parameters. The training results are shown in the Figure 2 below.

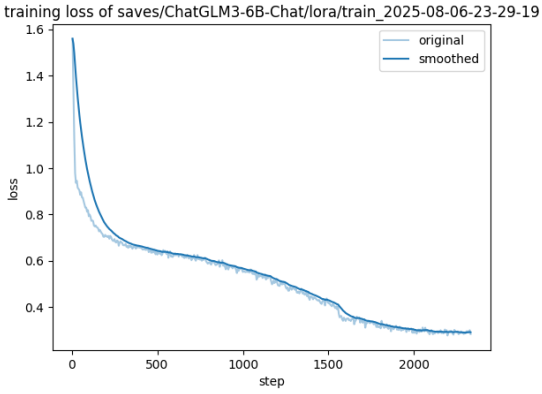

Figure 2. Systematic two-stage adaptive fine-tuning framework

The training loss curve exhibits a generally positive trend. The smoothed loss decreases with increasing training steps, indicating continuous model learning and optimization, along with gradually improving task-fitting capability. Subsequently, the loss decreases more slowly, gradually approaching a stable value and indicating model convergence. Further training would yield diminishing returns, so we terminated the process.

4.2. Comparison with Traditional Prediction Models

After multiple fine-tuning rounds, the model effectively learned from the training data and shifted its focus from pure text to power load prediction. After training, the model's performance was evaluated on the test set.

Table 7 compares the performance of the fine-tuned LLM with multiple baselines on the power load prediction task, including the pre-trained LLM before fine-tuning, three simple prediction methods, AutoARIMA, and LSTM. We can clearly see that fine-tuning significantly improves the performance of LLM on this task. The pre-trained LLM before fine-tuning does not perform well in predicting power load. Experimental data shows that the pre-trained LLM achieves a root mean square error of 1890.116, a mean absolute error of 1436.583, and a mean absolute percentage error of 19.837% on the test data. Its performance is even significantly worse than the P1H and STY methods used as the baselines. This occurs because the non-fine-tuned LLM primarily focuses on text rather than time series data, failing to adequately model power load scenarios or comprehend temporal patterns. The fine-tuned LLM outperformed the LSTM model and AutoARIMA on the power load forecasting task, significantly surpassing three other simpler forecasting methods. Experimental data showed that the fine-tuned LLM achieved a root mean square error of 247.401, a mean absolute error of 184.083, and a mean absolute percentage error of 3.345% on the test set. It also achieved a 16.492% higher accuracy than the pre-trained LLM, a 2.206% higher accuracy than the LSTM model, and a 5.715% higher accuracy than AutoARIMA on the validation set.

Table 7. Performance of different prediction methods on power load data

Method	MAE	MAPE	RMSE
P1H	332.732	6.791%	457.983
AP24H	843.556	20.551%	1011.646
STY	353.355	8.863%	514.572
AutoARIMA	700.291	9.060%	975.941
LSTM	321.125	5.551%	420.997

ChatGLM-BF	1436.583	19.837%	1890.116
ChatGLM-AF	184.083	3.345%	247.401

Figure 3 shows the prediction differences among various methods on the test data. The experimental results demonstrate that LLMs exhibit strong generalization capabilities, capturing complex patterns and long-range dependencies in data. This enables them to not only understand the context of natural language but also handle data with temporal and trend characteristics, such as time series data. Although LSTM and AutoARIMA are designed for time series data, they have limitations in capturing complex nonlinear relationships and high-dimensional interactions. Although LSTM excels at capturing long- term dependencies, it requires manual hyperparameter tuning for different tasks. While AutoARIMA can be configured with a period parameter to help it learn periodic data, it only focuses on a single period and performs poorly on data with multiple periodic attributes. In contrast, LLMs, with their large number of parameters and training data, can discover and capture more complex temporal patterns, even automatically learning patterns in time series data without explicit guidance. Furthermore, through self-attention mechanisms, LLMs can capture long-range dependencies, enabling them to process data with longer time spans and better avoid the issues inherent in traditional RNN structures.

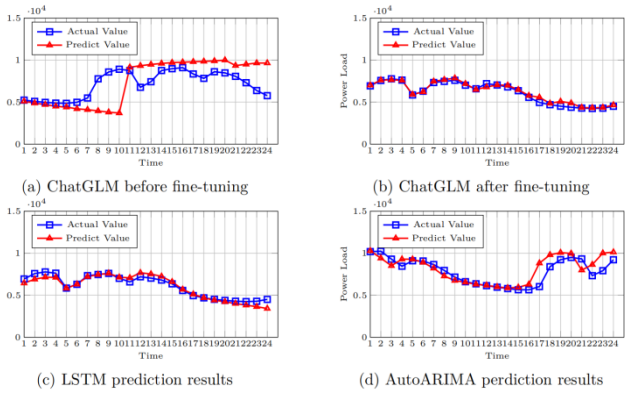

Figure 3. Comparison of different methods

5. Conclusion

To achieve power load forecasting, this paper constructs a two-layer fine-tuned ChatGLM3-6b prediction model. Experimental testing reveals that traditional methods such as AutoARIMA and LSTM perform poorly in power market forecasting. To address this issue, we draw on the successful experience of large language models in other fields and fine-tune the model to shift its focus from text generation to power load forecasting. To support fine-tuning experiments, we develop a data-to-text prompt conversion method and cre- ate datasets for model attention shifting and downstream fine-tuning tasks. Validation on a test set demonstrates that the two-layer fine-tuned ChatGLM3-6b prediction model outperforms traditional forecasting methods in terms of accuracy. The findings from this study can provide improved solutions for power grid maintenance and operation, exploring the application of LLMs in power load forecasting and demonstrating promising research prospects.

Acknowledgements.

Not applicable.

References

[1] Massaoudi M, Refaat SS, Chihi I, Trabelsi M, Oueslati FS, and Abu-Rub H. A novel stacked generalization ensemble-based hybrid lgbm-xgb-mlp model for short-term load forecasting. Energy, 2021; 214:118874.

[2] Zhou H, Zhang S, Peng J, Zhang S, Li J, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence, 2021; 35(12):11106-11115.

[3] Wu H, Xu J, Wang J, Long M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Advances in neural information processing systems, 2021; 34:22419-22430.

[4] Zeng A, Chen M, Zhang L, Xu Q. Are transformers effective for time series forecasting? In Proceedings of the AAAI conference on artificial intelligence, 2022; 37(9):11121-11128.

[5] Nie Y, Nguyen NH, Sinthong P, Kalagnanam J. A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730. 2023.

[6] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, et al. Lora: Low-rank adaptation of large language models. ICLR, 2021; 1(2):3.

[7] OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774. 2024.

[8] Li C, Gan Z, Yang Z, Yang J, Li L, et al. Multi-modal foundation models: From specialists to general-purpose assistants. Foundations and Trends® in Computer Graphics and Vision, 2024; 16(1-2):1-214.

[9] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. Attention is all you need. arXiv preprint arXiv:1706.03762. 2023.

[10] Frome A, Corrado GS, Shlens J, Bengio S, Dean J, et al. Devise: A deep visual-semantic embedding model. In Neural Information Processing Systems, 2013: 26.

[11] Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning. PmLR, 2021: 8748-8763.

[12] Chen YC, Li L, Yu L, El Kholy A, Ahmed F, et al. Uniter: Universal image-text representation learning. In European conference on computer vision. Cham: Springer International Publishing, 2020: 104-120.

[13] Lu J, Batra D, Parikh D, and Lee S. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. Advances in neural information processing systems, 2019: 32.

[14] Peng Z, Wang W, Dong L, Hao Y, Huang S, et al. Kosmos-2: Grounding multimodal large language models to the world. arXiv preprint arXiv:2306.14824. 2023.

[15] Taylor JW, McSharry PE. Short-term load forecasting methods: An evaluation based on european data. IEEE Transactions on Power Systems, 2007; 22(4):2213–2219.
[16] Fekri MN, Patel H, Grolinger K, Sharma V. Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network. Applied Energy, 2021; 282:116177.
[17] Box GEP, Jenkins GM. Time Series Analysis: Forecasting and Control. Prentice Hall PTR, USA, 3rd edition, 1994.
[18] Hong T, Fan S. Probabilistic electric load forecasting: A tutorial review. International Journal of Forecasting, 2016; 32(3):914–938.
[19] Houlsby N, Giurgiu A, Jastrzebski S, Morrone B, de Laroussilhe Q, et al. Parameter-efficient transfer learning for nlp. In International conference on machine learning. PMLR, 2019: 2790-2799.
[20] Amey Hengle AK, Singh S, Bandhakavi A, Akhtar MS, Chakroborty T. Intent-conditioned and non-toxic counterspeech generation using multi-task instruction tuning with rlaif. Preprint, 2024.
[21] Wani MA, Afzal S. A new framework for fine tuning of deep networks. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017: 359–363.
[22] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 2022; 35:27730-27744.
[23] Xue H, Salim FD. Promptcast: A new prompt-based learning paradigm for time series forecasting. IEEE Transactions on Knowledge and Data Engineering, 2024; 36(11):6851-6864.
[24] Jin M, Wang S, Ma L, Chu Z, Zhang JY, et al. Time- LLM: Time series forecasting by reprogramming large language models. In The Twelfth International Conference on Learning Representations, 2024.
[25] Kim T, Kim J, Tae Y, Park C, Choi JH, Choo J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In International Conference on Learning Representations, 2022.
[26] Défossez A, Copet J, Synnaeve G, Adi Y. High fidelity neural audio compression. arXiv preprint arXiv:2210.13438. 2022.
[27] Touvron H, Martin L, Stone K, Albert P, Almahairi A, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288. 2023.
[28] Brown T, Mann B, Ryder N, Subbiah M, Kaplan J, et al. Language models are few-shot learners. Advances in neural information processing systems, 2020; 33:1877-1901.
[29] Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM computing surveys, 2021; 55(9):1-35.
[30] Box GEP, Jenkins GM, Reinsel GC, Ljung GM. Time Series Analysis: Forecasting and Control. Wiley, Hoboken, NJ, USA, 5th edition, 2015.
[31] Hyndman RJ, Khandakar Y. Automatic time series forecasting: The forecast package for r. Journal of Statistical Software, 2008; 27(3):1–22.
[32] Taieb SB, Hyndman RJ. A gradient boosting approach to the kaggle load forecasting competition. International Journal of Forecasting, 2014; 30(2):382–394.
[33] Winters PR. Forecasting sales by exponentially weighted moving averages. Management Science, 1960; 6:324–342.
[34] Montgomery DC, Johnson LA, Gardiner JS. Forecasting and Time Series Analysis. McGraw-Hill, New York, NY, USA, 2nd edition, 1990.
[35] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997; 9(8):1735–1780.
[36] Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent neural networks. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013: 6645–6649.
[37] Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J. Lstm: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 2017; 28(10):2222–2232.
[38] Gers FA, Schmidhuber J, and Cummins F. Learning to forget: continual prediction with lstm. In 1999 Ninth International Conference on Artificial Neural Networks ICANN 99, 1999; 2:850–855.
[39] LeCun Y, Bengio Y, and Hinton G. Deep learning. Nature, 2015; 521(7553):436–444.