Empower Pre-trained Large Language Models for Building-level Load Forecasting

Yating Zhou, *Student Member*, *IEEE*, and Meng Wang, *Senior Member*, *IEEE*

Abstract—**Short-term building-level load forecasting is significant for enhancing the stability and efficiency of power grids. Despite the superior forecasting performance, machine learning methods heavily rely on sufficient historical load data for training. This paper addresses the challenge of limited or unavailable historical data, which often occurs in new communities or due to data storage issues. This paper proposes BlackInter, a novel black-box tuning inductive adapter based on pre-trained large language models (LLMs), specifically tailored for building-level load forecasting. Our method leverages the inherent generalization capabilities of LLMs with no need for pre-training on similar domains and fine-tuning by target data. Furthermore, BlackInter exploits spatial correlations among nearby buildings to improve forecast accuracy. It directly adapts to different building sizes, a feature lacking in existing approaches. Our approach does not require prior knowledge of the LLM's structure or parameters. We validate the effectiveness of the LLM-based BlackInter using real-world datasets and compare it with four existing methods. Under the settings of limited and no historical data, the prediction error by our method can be only 42.88% and 69.44% of the error by the best alternative methods, respectively.**

Index Terms—**Load forecasting, pre-trained large language model, spatial correlations, few-shot forecasting, zero-shot forecasting.**

I. Introduction

With the implementation of the advanced metering infrastructure (AMI), short-term building-level load forecasting has diverse applications in power systems and is crucial for improving the stability and efficiency of power systems. System operators need accurate building-level load forecasts to optimize energy management, ensuring the economical and reliable operation of power systems [1], [2]. Accurate forecasting is also important for demand response (DR) programs to ensure the system stability [3]. With building-level load forecasts, system operators can allocate pre-DR resources and evaluate post-DR performance effectively [1], [4]. Additionally, building-level load forecasts can support power market participants in making better decisions for price schemes and energy transactions [5], [6], thereby making energy systems more economical and efficient. Furthermore, accurate load forecasts help building owners better understand their energy usage patterns, optimize the use of smart technologies, and shift energy consumption to off-peak periods [4], [5]. This reduces electricity bills for customers and alleviates supply stress on power systems during peak demand times. There has been extensive research on short-term load forecasting, which can generally be classified into two types: statistical methods [7], [8], [9], [10] and machine learning methods [11], [12], [3], [13], [14], [15], [16], [17], [18], [19], [20]. Compared with statistical methods, machine learning methods have the advantage of extracting complicated features and spatial-temporal correlations among time series load data, resulting in superior forecasting performance. However, training an accurate machine learning model requires a large amount of historical training data, but sufficient historical building-level load data might not exist in practice. For example, limited building-level historical load data is available in a newly built community [21]. Moreover, the failure of measurement equipment and data storage systems may also result in limited or even no historical data at all [22].

The cases where only limited or no target data is available for training is known as *few-shot learning* [23] and *zero-shot learning* [24], respectively. The existing techniques for the few-shot building-level load forecasting problem often use source load data from other domains, which are different from but related to the target data domain we aim to solve, to pre-train a model and then use the limited few-shot target load data to fine-tune the pre-trained model [21], [25], [26], [27], [22]. This technology is generally referred to as transfer learning. For example, reference [21] combines multiple kernel learning and transfer learning to resolve residential load forecasting with few historical data. Reference [26] integrates graph neural networks and transfer learning technology for load forecasting. The key factor to the success of most transfer learning methods is to select the source load data that is highly similar to the target load data. Otherwise, a negative transfer might occur, meaning that the target forecasting performance will be impaired by the transferred knowledge [26]. In practice, it might be difficult to obtain satisfying source domain data because of the limited availability of public data and the privacy concerns to obtain data from other utilities [28]. Moreover, all these existing methods require a certain amount of target historical data to enhance the target building-level load prediction performance and do not extend to the zero-shot learning case.

Spatial correlations exist in building-level load patterns due to similar weather conditions and human behaviors [18], [29]. Some load forecasting studies [18], [19], [20], [26] model spatial correlations by a fixed graph, where each building is a node of a graph and the corresponding load consumption is the node features. Both the model training and testing need to be in the same fixed graph. Thus, these methods lack the *inductive* capability, which is the ability to adapt to different graph sizes in the real-time testing stage.

To address the above issues in this paper, we pose the following question: *Can we propose an inductive method that does not require the source load data that is similar to the target load data but still achieves satisfying few-shot and zero-*

shot load forecasting accuracy?

This paper answers this question by exploring the pre-trained large language models (LLMs) with powerful generalization capabilities. The popular pre-trained LLMs, such as GPT [30] and Llama [31], have recently been applied in time-series data analysis and achieved satisfying results [32], [33], [34], [35]. Trained by large-scale unstructured text data, pre-trained LLMs contain rich knowledge and can generalize well for various downstream tasks [36]. Moreover, pre-trained LLMs have impressive few-shot and zero-shot learning performance [37]. Despite the great promise of adopting pre-trained LLMs for building-level load forecasting, some challenges still remain to be solved. First, large-scale pre-trained LLMs inevitably require substantial memory overhead for storage and the computation of first-order gradients by back-propagation [38]. Moreover, the detailed architecture and parameter information of pre-trained LLMs are often not open-source due to the potential risk of misuse and commercial considerations [39], [40]. The huge memory cost and the possibility of inaccessible pre-trained LLMs prevent the efficient applications of pre-trained LLMs for building-level load forecasting. In addition, most existing pre-trained LLM-based methods do not consider the spatial correlations among multivariate time-series data [32], [33], [34], [35].

In this paper, we propose a black-box tuning inductive adapter (BlackInter) to empower pre-trained LLMs for building-level load forecasting. Specifically, the proposed BlackInter contains three parallel components at the input side, including the input embedding layer, the spatial embedding layer, and the feature extraction layer, to generate input for pre-trained LLMs. It also includes an output projection layer that transforms the output embedding by pre-trained LLMs to predicted load values. The input embedding layer extracts the input load features directly. Inspired by the idea in [41], we propose an inductive spatial embedding layer to aggregate the spatial features of neighbors. The feature extraction layer extracts the dominant dynamics in load data by dynamic mode decomposition (DMD) [42] to enhance the forecasting model performance. In order to reduce the memory cost and handle the issue of limited access to pre-trained LLMs' structures and parameters, we propose a so-called *Mix-Adam* method to optimize the parameters only using the inference results of pre-trained LLMs. That is possible because pre-trained LLMs' owners have started releasing pre-trained LLMs as a service, which allows users to access the model inference results or intermediate embedding [39], [43]. To the best of our knowledge, this paper is the ***first*** study to explore pre-trained LLMs for building-level load forecasting so that the resulting method is inductive to different graph sizes and does not require information about the model architecture and parameters of pre-trained LLMs. The proposed BlackInter is a *memory-efficient*, *plug-in*, and *inductive* LLM-based approach. It can efficiently empower pre-trained LLMs to predict accurate future load values for buildings with limited or even no historical load data, which benefits the operation and optimization of energy systems.

The rest of this paper is structured as follows. Section II states the problem formulation, the methodology's motivation, and the technical challenges. Section III represents the overall architecture and the details of the designed BlackInter method. Section IV presents the experimental results that verify the effectiveness of the proposed technique. Section V concludes the paper.

II. Problem Formulation and Challenge

A. Problem formulation

The building-level load forecasting problem aims to learn a prediction function $f(\cdot;\Theta)$, parameterized by trainable weights Θ, that maps the past K steps of loads to the next F steps of load values. After trained on historical data, $f(\cdot;\Theta)$ can be leveraged to forecast future loads given real-time measurements. In the existing studies [18], [19], [20], [26], the number of buildings in the training stage is the same as that in the real-time testing stage. However, the expanding and decreasing of buildings often happens in one community, leading to different numbers of buildings in the training and testing stages. Moreover, when the model is transferred to other test data, the number of buildings is likely to be different. Thus, $f(\cdot;\Theta)$ needs to be inductive to forecast the load values of an arbitrary number of buildings in the real-time operation stage.

Mathematically, the problem formulation in this paper is as follows. Let B_T denote the building set in the offline training stage, and the total number of the buildings in B_T is N, i.e., $|B_T| = N$. Let $\mathbf{x}_{it} \in \mathbb{R}^K$ represent the historical loads of building i ($i \in B_T$) from time $t-K+1$ to time t. Let $\mathbf{x}_{it}(a)$ denote the ath element in $\mathbf{x}_{it}$. Let $\mathcal{N}^i$ denote the set of neighboring buildings of building i, and $\mathcal{N}_t^i = \{\mathbf{x}_{jt}, j \in \mathcal{N}^i\}$ denote the load values of buildings in the neighbor set from time $t-K+1$ to time t. Let $\mathbf{y}_{it} \in \mathbb{R}^F$ and $\hat{\mathbf{y}}_{it} \in \mathbb{R}^F$ denote the ground-truth and predicted values of building i from time $t+1$ to $t+F$, respectively. Then, the prediction in the training stage is

$$\hat{\mathbf{y}}_{it} = f(\mathbf{x}_{it}, \mathcal{N}_t^i; \Theta). \tag{1}$$

The objective is to optimize the learnable parameters Θ so that the predicted load values are close to the ground truth load values, i.e.,

$$\min_{\Theta} L(\Theta) := \frac{1}{N \times (T-F-K+1) \times F} \sum_{i \in B_T} \sum_{t=K}^{T-F} \sum_{j=1}^{F} |\hat{\mathbf{y}}_{it}(j) - \mathbf{y}_{it}(j)|^2, \tag{2}$$

where T is the total historical time steps. $\hat{\mathbf{y}}_{it}(j)$ and $\mathbf{y}_{it}(j)$ denote the jth element of $\hat{\mathbf{y}}_{it}$ and $\mathbf{y}_{it}$, respectively. Note that the input and target output of the function $f(\cdot;\Theta)$, representing load values for different time periods with varying sizes, are denoted by $\mathbf{x}_{it}$ and $\mathbf{y}_{it}$ for simplicity. Let Θ^* denote the optimal parameters obtained in the offline training stage by solving (2).

Let B_E denote the building set in the real-time operation stage, where the total number of buildings is M, i.e., $|B_E| = M$. Let $\mathbf{f}_{us} \in \mathbb{R}^K$ denote the observed real-time measurements of building u ($u \in B_E$) from time $s-K+1$ to s. Let $\mathcal{P}^u$ denote the neighboring buildings of building u, and $\mathcal{P}_s^u = \{\mathbf{f}_{js}, j \in$

$\mathcal{P}^u\}$ denote the load values of buildings in the neighbor set B_E from time $s-K+1$ to time s. Then, the corresponding load values in the future F steps can be predicted by the learned model as

$$\hat{\mathbf{p}}_{us} = f(\mathbf{f}_{us}, \mathcal{P}^u_s; \Theta^*), \tag{3}$$

where $\hat{\mathbf{p}}_{us}$ is the predicted load values of building u from time $s+1$ to time $s+F$.

The challenges of the problem result from two aspects. One is the lack of enough training data to fully optimize Θ when there is limited or even no historical training samples, i.e., T is close to $K+F$. The other challenge is ensuring the model inductive capability for different numbers of buildings in training and real-time operation stages when the spatial correlations are considered.

B. Motivation and challenge for pre-trained LLM-based forecasting models

Pre-trained LLMs possess strong representation and generalization capabilities on new modalities [44], [45]. The capability of LLMs to adapt to new modalities may result from the learned common modality-agnostic representation features when LLMs are trained by large text datasets over many natural language tasks [46]. The forecasting model $f(\cdot;\Theta)$ can contain a pre-trained LLM as a subnetwork. Then, when optimizing Θ in the offline stage, the parameters in the LLM are pre-trained and provide a good initialization point. That can ensure superior performance with limited or even no training samples in the downstream tasks. Despite the great potential of pre-trained LLMs, there exist technical challenges when we adopt pre-trained LLMs in the building-level load forecasting problem. First, storing the large-scale pre-trained LLMs takes up huge memory space. Second, computing the first-order gradients by the back-propagation process to optimize Θ needs to compute $\frac{\partial L(\Theta)}{\partial \hat{\mathbf{y}}_{it}} \frac{\partial \hat{\mathbf{y}}_{it}}{\partial \Theta}$, which not only takes memory overhead [38] but also requires knowledge of the architecture and parameters of pre-trained LLMs. However, this information may not be open-source for commercial considerations [39], [40]. In addition, most existing pre-trained LLM-based methods do not consider the spatial correlations among time-series data [32], [33], [34], [35].

III. Methodology

In order to adopt pre-trained LLMs to enhance building-level load forecasting, this paper proposes BlackInter, which contains a few trainable layers that can be combined with any pre-trained LLMs to achieve superior load forecasting performance. Section III-A summarizes the overall architecture of the proposed load forecasting method in this paper. Section III-B introduces the structure details of the proposed BlackInter. Section III-C presents the black-box optimization method for tuning the parameters of BlackInter.

A. Overall architecture

As Fig.1 shows, the pre-trained large language model (LLM) is regarded as a frozen and unknown backbone, which can provide output embedding given an input. The designed BlackInter contains three parallel layers at the input side and an output projection layer at the output side of the pre-trained LLM. Specifically, the three parallel layers include the input embedding layer, the spatial embedding layer, and the feature extraction layer. The training sample $\mathbf{x}_{it}$ is sent to the input embedding layer and the feature extraction layer to generate the input embedding E^1_{it} and feature embedding E^3_{it}, respectively. The training sample $\mathbf{x}_{it}$ and its corresponding neighbor samples $\mathcal{N}^i_t = \{\mathbf{x}_{jt}, j \in \mathcal{N}^i\}$ are sent to the spatial embedding layer to generate the spatial embedding E^2_{it}. The concatenated embedding of E^1_{it}, E^2_{it}, and E^3_{it}, is regarded as the input to the pre-trained LLM, and the pre-trained LLM generates $\mathbf{h}_{it}$, the output embedding accordingly. The output projection layer converts the output embedding $\mathbf{h}_{it}$ to the predicted load values $\hat{\mathbf{y}}_{it}$ of building i from time $t+1$ to time $t+F$. Note that only the parameters in the BlackInter are trainable, and the parameters in the pre-trained LLM are frozen.

B. The components of BlackInter

Next, we present the structure details of each component in BlackInter.

1) Input embedding layer: The input embedding layer extracts input load features directly. The input embedding layer mainly consists of three steps: normalization, patching, and linear transformation.

Given a training sample $\mathbf{x}_{it} \in \mathbb{R}^K$, we first normalize the training data by its mean and variance. Second, the normalized training sample $\mathbf{x}^{\mathrm{n}}_{it}$ is divided into several patches by applying the patching method in [47]. The benefit of patching data is to preserve the local semantic information by grouping local time series load data information into one patch. Let L_p denote the patch length and S denote the stride, i.e., the non-overlapping region between two consecutive patches. Then, the number of patches P is calculated by

$$P = \lfloor \frac{K - L_p}{S} \rfloor + 2, \tag{4}$$

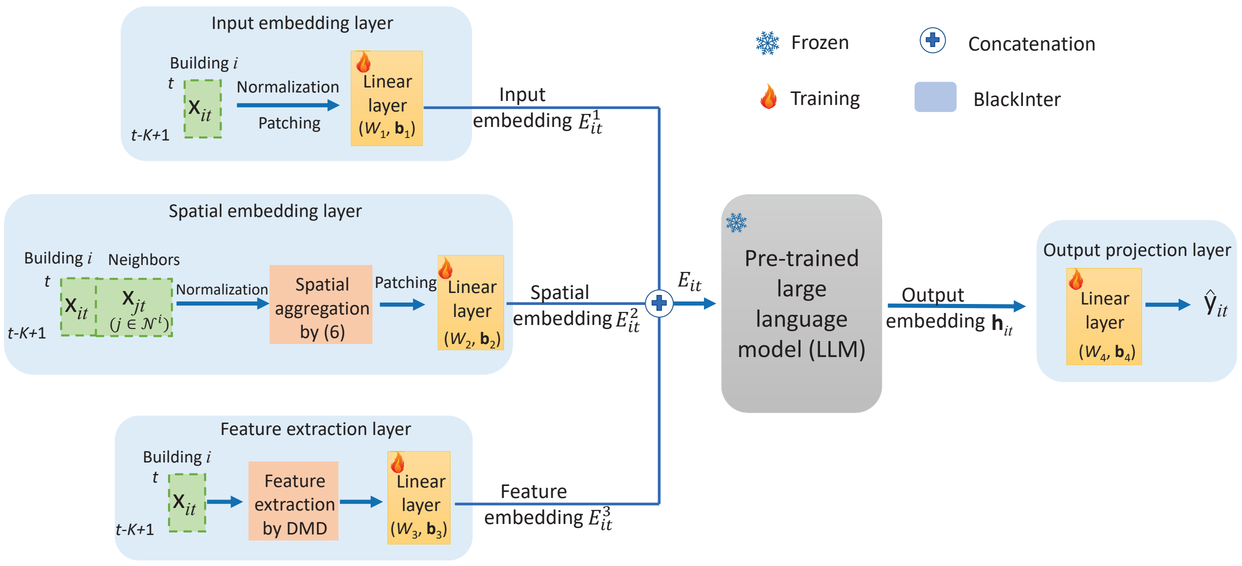
where $\lfloor\cdot\rfloor$ means the floor function. Let $\mathbf{x}^{\mathrm{n\_p}}_{it} \in \mathbb{R}^{P\times L_p}$ denote the training sample after patching. Finally, the sample $\mathbf{x}^{\mathrm{n\_p}}_{it}$ after normalization and patching is sent to a linear layer, which can be represented by

$$E^1_{it} = \mathbf{x}^{\mathrm{n\_p}}_{it} W_1^\top + \mathbf{b}_1, \tag{5}$$

where $W_1 \in \mathbb{R}^{d_m \times L_p}$ and $\mathbf{b}_1 \in \mathbb{R}^{d_m}$ are learnable parameters. $E^1_{it} \in \mathbb{R}^{P\times d_m}$ is the generated input embedding, and d_m is the column size of the input for the pre-trained LLM.

2) Spatial embedding layer: The spatial embedding layer aims to aggregate the information from the neighbors to enhance the forecasting performance. The spatial embedding layer encompasses four steps: (i) neighbor sampling, (ii) normalization, (iii) neighbor information aggregation, and (iv) patching and linear transformation.

The first step is to determine the neighbor sets of each building. The load consumption patterns have strong spatial correlations among the same type of buildings, which is verified in section IV-C. Thus, we randomly select B buildings



Input embedding layer
Building i
t
t-K+1
x_{it}
Normalization
Patching
Linear layer (W_1, $\mathbf{b}_1$)
Input embedding E^1_{it}
Spatial embedding layer
Building i
Neighbors
x_{jt} ($j \in \mathcal{N}^i$)
Spatial aggregation by (6)
Linear layer (W_2, $\mathbf{b}_2$)
Spatial embedding E^2_{it}
Feature extraction layer
Feature extraction by DMD
Linear layer (W_3, $\mathbf{b}_3$)
Feature embedding E^3_{it}
E_{it}
Pre-trained large language model (LLM)
Output embedding $\mathbf{h}_{it}$
Output projection layer
Linear layer (W_4, $\mathbf{b}_4$)
$\hat{y}_{it}$
Frozen
Training
Concatenation
BlackInter

Fig. 1. Overall architecture of the pre-trained LLM-based BlackInter forecasting method.

($B < N$) from the N buildings of the same type to construct the neighbor set $\mathcal{N}^i$ for each building i. We do not choose all other buildings as neighbors because it might lead to data redundancy and degrade the model performance. Note that the neighbor set in the real-time testing stage is also constructed by the random selection. In the second step, we apply normalization to the training sample $\mathbf{x}_{it}$ and its corresponding neighbor sample $\mathbf{x}_{jt} \in \mathbb{R}^K$ ($j \in \mathcal{N}^i$). Let $\mathbf{x}^{\mathrm{n}}_{jt}$ denote the normalized neighbor sample $\mathbf{x}_{jt}$. Then, we aggregate the information from the training sample and its neighbor samples by

$$\mathbf{x}^{\mathrm{agg}}_{it} = \frac{\mathbf{x}^{\mathrm{n}}_{it} + \sum_{j\in\mathcal{N}^i} \mathbf{x}^{\mathrm{n}}_{jt}}{B+1}. \tag{6}$$

Since the neighbor information aggregation is performed node (building) by node, the numbers of buildings in the offline training and real-time testing stages can be different. Note that the number of neighbor buildings B can be selected differently in the offline and online testing stages, too.

After we obtain the aggregated features $\mathbf{x}^{\mathrm{agg}}_{it} \in \mathbb{R}^K$, we divide it into patches using the patching method mentioned in the previous section to obtain $\mathbf{x}^{\mathrm{agg\_p}}_{it} \in \mathbb{R}^{P\times L_p}$. We then send it to a linear layer as

$$E^2_{it} = \mathbf{x}^{\mathrm{agg\_p}}_{it} W_2^\top + \mathbf{b}_2, \tag{7}$$

where $W_2 \in \mathbb{R}^{d_m\times L_p}$ and $\mathbf{b}_2 \in \mathbb{R}^{d_m}$ are learnable parameters. $E^2_{it} \in \mathbb{R}^{P\times d_m}$ is the generated spatial embedding.

3) Feature extraction layer: The feature extraction layer extracts the dominated features for time series data dynamics to assist the prediction model learning. The feature extraction layer mainly contains two steps: dominated feature extraction by the dynamic mode decomposition (DMD) [42] and linear transformation.

First, we present how to apply DMD to extract the dominant features behind a training sample. For an univariate training sample $\mathbf{x}_{it} \in \mathbb{R}^K$, we first convert it to a multi-dimensional augmented data matrix using the Hankel operator to fully capture the dynamics of load series [42]. The constructed multi-dimensional augmented data matrix using the Hankel operator is represented as

$$\begin{aligned} O_{it} &= \begin{bmatrix} \mathbf{x}_{it}(1) & \mathbf{x}_{it}(2) & \cdots & \mathbf{x}_{it}(K+1-c) \\ \mathbf{x}_{it}(2) & \mathbf{x}_{it}(3) & \cdots & \mathbf{x}_{it}(K+2-c) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{it}(c) & \mathbf{x}_{it}(c+1) & \cdots & \mathbf{x}_{it}(K) \end{bmatrix} \\ &= [\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_{K+1-c}], \end{aligned} \tag{8}$$

where $\mathbf{x}_{it}(a)$ denote the ath element for $\mathbf{x}_{it}$.

As shown in references [48] and [49], there exist time dependencies among load values during a local time neighborhood, and the observation $\mathbf{x}_{it}(a)$ can be expressed as the linear combination of previous q observations with negligible residual error e_a, which is described by the q-order autoregressive model

$$\mathbf{x}_{it}(a) = \sum_{w=1}^{q} \alpha_w \mathbf{x}_{it}(a-w) + e_a \approx \sum_{w=1}^{q} \alpha_w \mathbf{x}_{it}(a-w), \tag{9}$$

where $\alpha_w \in \mathbb{R}$ is a weight. Then, there exists a linear operator $\mathcal{A} \in \mathbb{R}^{c\times c}$ satisfying that

$$\mathbf{o}_{m+1} \approx \mathcal{A}\mathbf{o}_m, \tag{10}$$

where $m = 1, \cdots, K-c$. Let two consecutive entry matrices O_1 and O_2 be constructed by

$$\begin{cases} O_1 = [\mathbf{o}_1, \cdots, \mathbf{o}_{K-c}] \in \mathbb{R}^{c\times(K-c)} \\ O_2 = [\mathbf{o}_2, \cdots, \mathbf{o}_{K-c+1}] \in \mathbb{R}^{c\times(K-c)}. \end{cases} \tag{11}$$

Following equation (10), we have

$$O_2 \approx \mathcal{A} O_1. \tag{12}$$

The eigenvalues of the matrix $\mathcal{A}$ are informative about the dynamic features of load series, such as the decay and growth [42], [50]. Thus, we extract the eigenvalues of $\mathcal{A}$ to construct the dominant features. The feature extraction process is as follows.

(1) Conduct the singular value decomposition (SVD) with rank r to approximate the matrix O_1 as

$$O_1 \approx U_r \Sigma_r V_r^\dagger, \tag{13}$$

where $\Sigma_r = \mathrm{diag}[\sigma_1, \cdots, \sigma_r] \in \mathbb{R}^{r\times r}$ contains the largest r singular values in diagonal entries. $U_r \in \mathbb{R}^{c\times r}$ and $V_r \in \mathbb{R}^{(K-c)\times r}$ include the corresponding left and right singular vectors. $\dagger$ represents the conjugate transpose operation.

(2) Project $\mathcal{A}$ onto the r-dimensional subspace U_r as

$$\tilde{\mathcal{A}} = U_r^\dagger \mathcal{A} U_r \approx U_r^\dagger O_2 V_r \Sigma_r^{-1}, \tag{14}$$

where the approximation equality is from (12) and (13). The r eigenvalues of the projected matrix $\tilde{\mathcal{A}}$ coincide with the r dominant eigenvalues of $\mathcal{A}$ [51].

(3) Conduct eigendecomposition on $\tilde{\mathcal{A}}$ by

$$\tilde{\mathcal{A}} = Q\Lambda Q^\dagger, \tag{15}$$

where $Q \in \mathbb{R}^{r\times r}$ and $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_r) \in \mathbb{R}^{r\times r}$ contain the eigenvectors and eigenvalues, respectively.

We select the top r eigenvalues of matrix $\mathcal{A}$ in (15) as features, and we also note that the singular values $[\sigma_1, \cdots, \sigma_r]$ of O_1 in (13) reflect the strength of signal component in load series [51]. Thus, we use both the dominant eigenvalues and singular values to construct the feature vector, denoted by $\mathbf{v}_{it} = (\lambda_1; \cdots, \lambda_r; \sigma_1; \cdots; \sigma_r) \in \mathbb{R}^{2r}$. Then, the constructed $\mathbf{v}_{it}$ is passed through a linear layer, which is described as

$$E_{it}^3 = \mathbf{v}_{it} W_3^\top + \mathbf{b}_3, \tag{16}$$

where $W_3 \in \mathbb{R}^{d_m\times 1}$ and $\mathbf{b}_3 \in \mathbb{R}^{d_m}$ are learnable parameters. $E_{it}^3 \in \mathbb{R}^{2r\times d_m}$ is the generated feature embedding. The main steps of the feature extraction layer are summarized in Fig. 2.

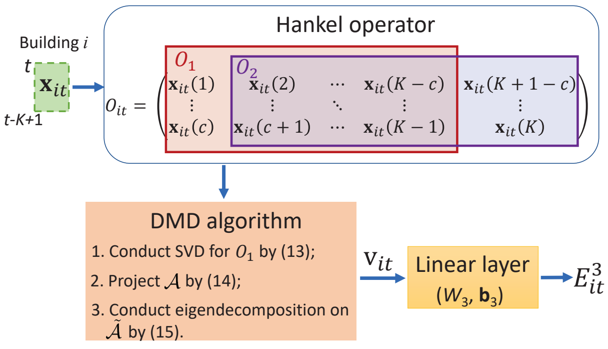

Hankel operator
Building i
t
$\mathbf{x}_{it}$
t-K+1
O_1
O_2
$O_{it} = \begin{pmatrix} \mathbf{x}_{it}(1) & \mathbf{x}_{it}(2) & \cdots & \mathbf{x}_{it}(K-c) & \mathbf{x}_{it}(K+1-c) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{x}_{it}(c) & \mathbf{x}_{it}(c+1) & \cdots & \mathbf{x}_{it}(K-1) & \mathbf{x}_{it}(K) \end{pmatrix}$
DMD algorithm
1. Conduct SVD for O_1 by (13);
2. Project $\mathcal{A}$ by (14);
3. Conduct eigendecomposition on $\tilde{\mathcal{A}}$ by (15).
$\mathbf{v}_{it}$
Linear layer
(W_3, $\mathbf{b}_3$)
E_{it}^3

Fig. 2. Illustration of the main steps in the feature extraction layer.

The input to the pre-trained LLM is the concatenation of the embeddings of all the three layers,

$$E_{it} = \begin{bmatrix} E_{it}^1 \\ E_{it}^2 \\ E_{it}^3 \end{bmatrix} \in \mathbb{R}^{2(P+r)\times d_m}. \tag{17}$$

4) Output projection layer: Let $\mathbf{h}_{it} \in \mathbb{R}^d$ denote the embedding of the last hidden layer of the pre-trained LLM. In natural language tasks, the output tokens are typically generated with randomness based on $\mathbf{h}_{it}$ by a decoding strategy [52], [53]. Unlike in natural language tasks, here we use a linear layer to transform $\mathbf{h}_{it}$ to the predicted load values, which can be described by

$$\hat{\mathbf{y}}_{it} = \mathbf{h}_{it} W_4^\top + \mathbf{b}_4, \tag{18}$$

where $W_4 \in \mathbb{R}^{F\times d}$ and $\mathbf{b}_4 \in \mathbb{R}^F$ are learnable parameters. $\hat{\mathbf{y}}_{it} \in \mathbb{R}^F$ is the predicted load values corresponding to the training sample $\mathbf{x}_{it}$.

C. Mix-Adam training method

The typical way to optimize the trainable parameters uses back-propagation to compute the first-order gradients and apply gradient descent based optimization methods. Because the memory cost for the back-propagation throughout the pre-trained LLM is high and the model parameters of the pre-trained LLM may be inaccessible, we propose an optimization method, referred to as Mix-Adam, to learn the trainable parameters in BlackInter. In Mix-Adam, we compute the first-order gradients for the parameters W_4 and $\mathbf{b}_4$ in the output linear layer, and compute the zero-order gradients using model inference results for the parameters $\{W_j, \mathbf{b}_j; j = 1, 2, 3\}$ in the linear layers at the input side. In this way, we avoid back-propagation throughout the backbone pre-trained LLM and do not need to know the architecture and parameter information of the pre-trained LLM. We name the proposed method as "mix" because it involves computing both first-order gradients and zero-order gradients. After we obtain the gradients, an adaptive moment estimation (Adam) method [54] is applied to optimize the parameters using the computed gradients. Adam has the advantages of adaptive learning rate and fast convergence rate [54]. Next, we present the calculation details of the proposed Mix-Adam.

1) Compute the first-order gradients of W_4 and $\mathbf{b}_4$: We apply chain rule to calculate the first-order gradients of W_4 and $\mathbf{b}_4$ using (18) and (2), which can be described as

$$\begin{cases} \nabla W_4 := \frac{\partial L}{\partial W_4} = \frac{\partial L}{\partial \hat{\mathbf{y}}_{it}} \frac{\partial \hat{\mathbf{y}}_{it}}{\partial W_4} \\ \nabla \mathbf{b}_4 := \frac{\partial L}{\partial \mathbf{b}_4} = \frac{\partial L}{\partial \hat{\mathbf{y}}_{it}} \frac{\partial \hat{\mathbf{y}}_{it}}{\partial \mathbf{b}_4}, \end{cases} \tag{19}$$

where ∇W_4 and $\nabla \mathbf{b}_4$ represent the first-order gradients of W_4 and $\mathbf{b}_4$, respectively.

2) Compute the zero-order gradients for the parameters $\{W_j, \mathbf{b}_j; j = 1, 2, 3\}$: We adopt the classic zero-order gradient estimator, simultaneous perturbation stochastic approximation (SPSA) [55], [56], to estimate the zero-order gradients using only the model inference results. SPSA is proven to converge theoretically [55] and demonstrates superior performance in optimizing neural network problems [57], [58], [59]. The estimation process can be described as

$$\hat{\nabla}\Theta' = \frac{L(\Theta' + \epsilon\mathbf{z}) - L(\Theta' - \epsilon\mathbf{z})}{2\epsilon}\mathbf{z}, \tag{20}$$

where $\Theta' = \{W_j, \mathbf{b}_j; j = 1, 2, 3\}$ and $\hat{\nabla}\Theta'$ denotes the zero-order gradient of Θ'. $\mathbf{z}$ is a random vector drawn from the standard Gaussian distribution $\mathcal{N}(0, I)$ and the size of $\mathbf{z}$ is the same as the size of Θ'. ϵ is the perturbation scale parameter.

3) Optimize the parameters Θ by Adam: Let $\nabla\Theta = \{\hat{\nabla}\Theta', \nabla W_4, \nabla \mathbf{b}_4\}$ denote the computed gradients of all parameters. In the iteration t, the parameter optimization process can be described by

$$\begin{cases} \mathcal{M}_{(t)} = \beta_1 \mathcal{M}_{(t-1)} + (1-\beta_1)\nabla\Theta_{(t-1)} \\ \mathcal{V}_{(t)} = \beta_2 \mathcal{V}_{(t-1)} + (1-\beta_2)(\nabla\Theta_{(t-1)})^2 \\ \hat{\mathcal{M}}_{(t)} = \frac{\mathcal{M}_{(t)}}{1-\beta_1^{(t)}} \\ \hat{\mathcal{V}}_{(t)} = \frac{\mathcal{V}_{(t)}}{1-\beta_2^{(t)}} \\ \Theta_{(t)} = \Theta_{(t-1)} - \alpha \hat{\mathcal{M}}_{(t)} / (\sqrt{\hat{\mathcal{V}}_{(t)}} + \gamma), \end{cases} \tag{21}$$

where $\mathcal{M}_{(t)}$ and $\mathcal{V}_{(t)}$ denote the biased first moment estimate and second raw moment estimate at iteration t, respectively. $\hat{\mathcal{M}}_{(t)}$ and $\hat{\mathcal{V}}_{(t)}$ are the biased-corrected first moment estimate and second raw moment estimate at iteration t, respectively. $\beta_1, \beta_2 \in [0, 1)$ are the hyperparameters that control the exponential decay rates. α is the learning rate and γ is a small constant for numerical stability. Θ_t is the updated trainable parameters at iteration t. The iterative process for updating Θ stops when the loss function values from (2) converges.

Fig. 3 shows the main process of the designed Mix-Adam method. In summary, we first calculate the first-order gradients for the parameters W_4 and $\mathbf{b}_4$ by (19), and then estimate the zero-order gradients for the parameters $\{W_j, \mathbf{b}_j; j = 1, 2, 3\}$ by (20) using model inference results. After calculating the gradients, we adopt Adam to optimize all the trainable parameters, which is described as (21).

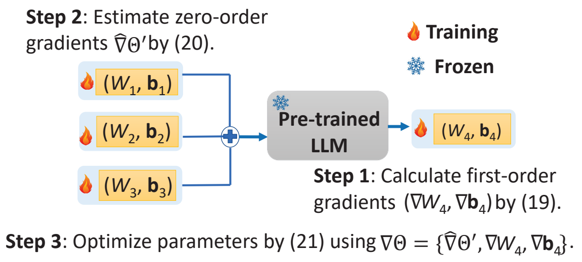

Step 2: Estimate zero-order gradients $\hat{\nabla}\Theta'$ by (20).
Training
Frozen
$(W_1, \mathbf{b}_1)$
$(W_2, \mathbf{b}_2)$
$(W_3, \mathbf{b}_3)$
Pre-trained LLM
$(W_4, \mathbf{b}_4)$
Step 1: Calculate first-order gradients $(\nabla W_4, \nabla \mathbf{b}_4)$ by (19).
Step 3: Optimize parameters by (21) using $\nabla\Theta = \{\hat{\nabla}\Theta', \nabla W_4, \nabla \mathbf{b}_4\}$.

Fig. 3. Illustration for the Mix-Adam method.

IV. Numerical Experiments

A. Experimental setup

Dataset description: All the load data used in this paper are selected from the one-year hourly load in the Typical Meteorological Year 3 (TMY3) [20]. Specifically, we select the residential and hotel load data from Colorado (CO), Hawaii (HI), and Washington (WA). The characteristics of these load data will be analyzed in Section IV-C.

Our method applies to different load types. Here we use residential and hotel load as examples. Note that the residential and hotel load data are used for training and testing separately. Some descriptions used in the experiments are explained as follows.

- **Target load data:** The load data in CO are used as the target data for the forecasting methods. The selected time periods and the number of buildings in the CO dataset vary in different experiments and will be described later.
- **Similar source load data:** The one-year load data of ten residential buildings (or ten hotels) in WA are set as the similar source load data, which have similar data characteristics as the target load data in CO.
- **Dissimilar source load data:** We choose the one-year hourly data of ten residential buildings (or ten hotels) in HI as the dissimilar source data, which have dissimilar data characteristics from the target load data.

Both the similar and dissimilar source load data are used to pre-train prediction models in some experimental settings. During the pre-training stage, 70% of the data starting from January are the training data, and the following 10% of the data are used for the validation.

Experiment setting: We mainly evaluate BlackInter on few-shot and zero-shot load forecasting cases, for which the method is designed. The main experimental settings in Section IV-C are described as follows and summarized in Table I. Our model training in all the following settings refers to training the three input layers and one output layer, while the pre-trained LLM remains fixed.

1) Few-shot evaluation: This experiment aims to show the model's capability to learn from few-shot target samples. Ten days of load data are selected from ten residential buildings (or ten hotels) in CO as the target data. The default selected time period is from December 22nd to December 31st for few-shot evaluation, except in the seasonal impact study in Section IV-C(5), where we use the summer data from July 22nd to July 31st. The ten-day load data are divided into training, validation, and testing datasets by the ratio of 0.3: 0.1: 0.6.

According to the availability of source load data, we design three few-shot experimental cases, and also construct one baseline setting to verify the effectiveness of few-shot learning.

- $\mathbf{F_1}$: Train the models from scratch only using the few-shot target training and validation data.
- $\mathbf{F_2}$: Pre-train using dissimilar source data and then fine-tune using few-shot target samples.
  - **Pre-training:** Pre-train the models using the dissimilar source load data in HI.
  - **Fine-tuning:** Fine-tune the pre-trained models using the few-shot target training and validation data.
- $\mathbf{F_3}$: Pre-train using similar source data and then fine-tune using few-shot target samples.
  - **Pre-training:** Pre-train the models using the similar source load data in WA.
  - **Fine-tuning:** Fine-tune the pre-trained models using the few-shot target training and validation data.
- **Baseline_F**: Train the models from scratch using 70% of the one-year data from the target ten residential buildings (or ten hotels) in CO as the training data and the following 10% of the data as the validation data.

For all the settings $\mathbf{F_1}$, $\mathbf{F_2}$, $\mathbf{F_3}$, and **Baseline_F**, the testing dataset is the six-day load data from ten residential buildings (or ten hotels) in CO.

2) Zero-shot evaluation: The experiment evaluates the models' zero-shot learning ability. That means the models are trained by the external load data completely, and we evaluate their generalization ability to forecast the target load values without any fine-tuning. The target task here is to predict the load values from ten residential buildings (or ten hotels) in CO in one month, while the models are trained by the historical data not in CO. We set two zero-shot experimental cases and one experimental case for the baseline as follows.

- **O$_1$**: Train the prediction models using the dissimilar source load data in HI.
- **O$_2$**: Train the prediction models using the similar source load data in WA.
- **Baseline_O**: The training process is the same as the setting **Baseline_F**.

For all settings **O$_1$**, **O$_2$**, and **Baseline_O**, the default target testing data are from the ten residential buildings (or ten hotels) in CO in one month of December, except in the seasonal impact study in Section IV-C(5), where we use the data from the ten residential buildings (or ten hotels) in CO in July.

3) Inductive ability evaluation: This experiment is designed to verify the inductive ability of the proposed pre-trained LLM-based BlackInter on the target data with different numbers of buildings in the training and testing stages. There are two experimental cases and two baselines in the inductive ability evaluation.

- **I$_1$**: Train on ten buildings and test on five buildings.
  - **Training:** The same as the setting **Baseline_F**.
  - **Testing:** Test the models on the load values in December from five residential buildings (or five hotels) in CO.
- **I$_2$**: Train on ten buildings and test on fifteen buildings.
  - **Training:** The same as the setting **Baseline_F**.
  - **Testing:** Test the models on the load values in December from fifteen residential buildings (or fifteen hotels) in CO.
- **Baseline_I$_1$**: Train and test on the same set of five buildings.
  - **Training:** Train and validate our model using the first 70% and the following 10% of the one-year data from the five residential buildings (or five hotels).
  - **Testing:** The same as the testing process in **I$_1$**.
- **Baseline_I$_2$**: Train and test on the same set of fifteen buildings.
  - **Training:** Train and validate our model using the first 70% and the following 10% of the one-year data from the fifteen residential buildings (or fifteen hotels).
  - **Testing:** The same as the testing process in **I$_2$**.

We adopt the pre-trained Llama-7B [31] as the pre-trained LLM backbone in this paper [60]. Llama-7B is a transformer-based larger language model and contains 7 billion parameters. Note that in all experimental cases, the parameters of Llama-7B are frozen, and only the parameters in BlackInter are trainable. In all the experiments in Section IV-C, we use the past 18 hours' load data to forecast the future 3-hour load values, i.e., $K = 18$ and $F = 3$. For the critical hyperparameters in BlackInter, we use the validation errors to select the optimal values. The critical hyperparameter settings in the residential load forecasting experiments are as follows: $L_p = 6, P = 5, B = 4, r = 1$. In the hotel load forecasting experiments, the parameters are set as: $L_p = 8, P = 4, B = 5, r = 2$. The required input column size d_m of Llama-7B is 4096.

TABLE I
SUMMARY OF EXPERIMENTAL SETTINGS

Setting	Abbreviation	Pre-training	Fine-tuning	Testing
Few-shot	**F$_1$**	N/A	Target data	Six days of target data
	F$_2$	Dissimilar source data		
	F$_3$	Similar source data		
	Baseline_F	Target data	N/A	
Zero-shot	**O$_1$**	Dissimilar source data	N/A	One month of target data
	O$_2$	Similar source data		
	Baseline_O	Target data		
Inductive	**I$_1$**	Ten buildings	N/A	Five buildings
	I$_2$	Ten buildings		Fifteen buildings
	Baseline_I$_1$	Five buildings		Five buildings
	Baseline_I$_2$	Fifteen buildings		Fifteen buildings

B. Method and evaluation metric

To compare to the proposed method in this paper, we select four recent forecasting methods: (i) graph wavenet (GW) [18], (ii) spatial-temporal graph neural network (STGCN) [61], (iii) patch time series transformer (PatchTST) [47], and (iv) decomposition transformers with auto-correlation (Autoformer) [62]. GW contains eight spatial-temporal blocks, and each block includes a gated temporal convolution layer and a graph convolution layer. STGCN includes two spatial-temporal blocks, and each block contains two temporal convolution layers and a graph convolution layer between the two temporal layers. PatchTST is a channel-independent patch time series transformer. Autoformer is a decomposition architecture by integrating the time-series decomposition block.

The hyperparameter settings of GW, STGCN, and Autoformer, follow the original settings in the references [18], [61], [62]. In PatchTST, the two critical parameters, the number of patches and patch length, are selected according to the validation errors of our experiments. Specifically, the number of patches and the patch length are set as 6 and 4 for the residential load experiments, and set as 6 and 3 for the hotel load experiments. The other parameters in PatchTST are the same as those in the open-source code published in [47].

We employ two evaluation metrics for comparisons, including the mean absolute percentage error (MAPE) and the normalized root mean squared error (NRMSE). Mathematically, the two evaluation metrics can be described by

$$\text{MAPE} = \frac{100\%}{M \times T' \times F} \sum_{i=1}^{M} \sum_{t=1}^{T'} \sum_{j=1}^{F} \frac{|\hat{\mathbf{y}}_{it}(j) - \mathbf{y}_{it}(j)|}{|\mathbf{y}_{it}(j)|}$$

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{M} \sum_{t=1}^{T'} \sum_{j=1}^{F} |\hat{\mathbf{y}}_{it}(j) - \mathbf{y}_{it}(j)|^2}{\sum_{i=1}^{M} \sum_{t=1}^{T'} \sum_{j=1}^{F} |\mathbf{y}_{it}(j)|^2}} \times 100\%, \tag{22}$$

where M denotes the total buildings in the testing stage, and T' represents the total time steps that are required to forecast in the testing stage.

C. Result analysis

1) Spatial correlation verification: First, we analyze the spatial correlations among the load values of residential buildings and hotels, respectively. We calculate Pearson correlation coefficients [63] among the load values in CO, HI, and WA, respectively. The heatmaps of correlation coefficients are shown in Fig. 4-6. As Fig. 4 shows, the load values in CO have

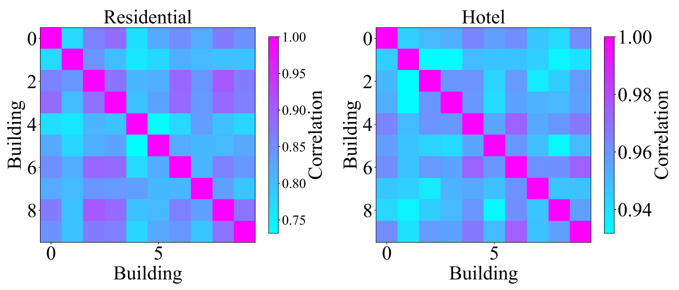


Fig. 4. Pearson correlation coefficients of load values in CO.

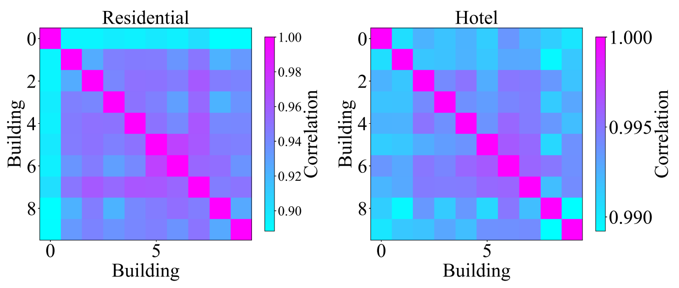


Fig. 5. Pearson correlation coefficients of load values in HI.

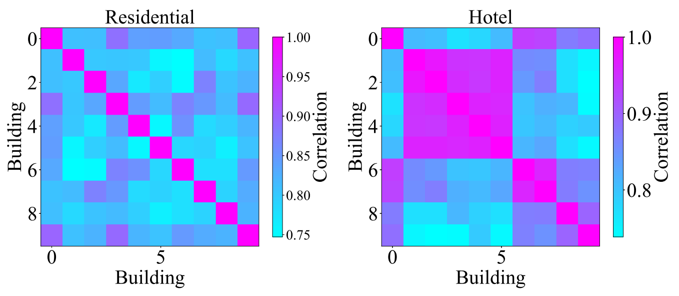


Fig. 6. Pearson correlation coefficients of load values in WA.

strong spatial correlations within the same type of buildings, especially the hotels. Fig. 5 shows that the load values in HI have stronger spatial correlations in both residential buildings and hotels. As Fig. 6 shows, the minimum Pearson correlation coefficients among the load values for residential and hotel buildings in WA are both greater than 0.7. This shows the load values in WA also exhibit strong spatial correlations.

2) Data characteristic analysis: Here, we analyze the load data characteristics in CO, WA, and HI. Fig. 7 and Fig. 8 show the probability density distributions of residential load values and hotel load values, respectively. We can see the load data characteristics in CO and WA are similar, and the data characteristics in CO and HI are dissimilar.

3) Few-shot learning evaluation: We evaluate the few-shot learning ability of the proposed pre-trained LLM-based

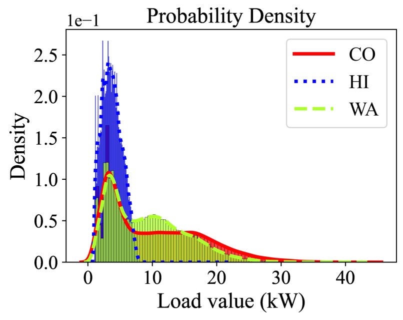


Fig. 7. Residential load data probability density.

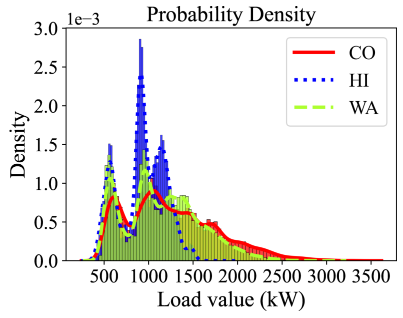


Fig. 8. Hotel's load data probability density.

BlackInter in the experimental settings described in Section IV-A. Table II shows the residential load forecasting results.

TABLE II
COMPARISONS ON FEW-SHOT RESIDENTIAL LOAD FORECASTING

Methods	From scratch ($\mathbf{F_1}$)		From dissimilar ($\mathbf{F_2}$)		From similar ($\mathbf{F_3}$)		**Baseline_F**	
	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE
STGCN	32.84%	28.69%	25.28%	26.06%	10.71%	11.25%	8.11%	8.86%
GW	24.47%	20.66%	16.67%	15.81%	13.59%	13.73%	8.89%	9.74%
PatchTST	23.02%	21.68%	12.62%	12.82%	9.72%	10.57%	8.97%	9.15%
Autoformer	26.64%	25.77%	17.81%	17.62%	11.07%	11.17%	11.55%	10.77%
BlackInter (ours)	**9.87%**	**10.75%**	**9.73%**	**10.78%**	**9.01%**	9.89%	**7.98%**	**8.65%**

Under the $\mathbf{F_1}$ setting where all these models are trained by few-shot target samples from scratch, the proposed pre-trained LLM-based BlackInter has much smaller forecasting errors compared with other methods. Specifically, for the residential load, the MAPE error by BlackInter is only 42.88% (9.87%/23.02%) of the MAPE error by PatchTST, which has the smallest error among alternative methods. Moreover, when the prediction models are pre-trained by source load data, either similar or dissimilar($\mathbf{F_2}$ or $\mathbf{F_3}$), the proposed pre-trained LLM-based BlackInter shows the best forecasting performance on the few-shot target task. Even when there exists sufficient target residential load data for training (**Baseline_F**), the proposed pre-trained LLM-based BlackInter still achieves the best accuracy. In addition, BlackInter has the least forecasting performance variation among different settings compared with other methods.

Table III represents the few-shot experimental results for forecasting hotel loads. As Table III shows, the pre-trained LLM-based BlackInter also outperforms the other four methods for hotel load forecasting in all few-shot cases. In particular, under $\mathbf{F_1}$, the MAPE error by BlackInter is only 58.74% (10.11%/17.21%) of the MAPE error by GW, which has the smallest error among alternative approaches. When sufficient target training data are available for training, BlackInter is better than PatchTST and Autoformer, while underperforming STGCN and GW. However, BlackInter is defined for few-shot and zero-shot learning when limited target data are available for training. The results in Table II and Table III demonstrate for the few-shot task, the proposed pre-trained LLM-based BlackInter does not require similar source load data for pre-training like other methods to achieve satisfying forecasting performance. When similar source load data is accessible, the proposed method still outperforms the other four methods for the few-shot task.

TABLE III
COMPARISONS ON FEW-SHOT HOTEL LOAD FORECASTING

Methods	From scratch ($\mathbf{F_1}$)		From dissimilar ($\mathbf{F_2}$)		From similar ($\mathbf{F_3}$)		**Baseline_F**	
	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE
STGCN	26.22%	29.11%	16.06%	17.67%	9.17%	11.88%	**4.02%**	**4.51%**
GW	17.21%	19.76%	10.93%	13.03%	8.31%	10.26%	5.08%	5.17%
PatchTST	26.01%	30.37%	12.36%	16.28%	10.67%	13.22%	10.05%	12.22%
Autoformer	23.61%	27.08%	14.59%	18.42%	11.45%	14.67%	9.59%	11.76%
BlackInter (ours)	**10.11%**	**11.84%**	**9.38%**	**10.32%**	**7.91%**	**8.97%**	7.82%	8.13%

4) Zero-shot learning evaluation: We employ the forecasting models trained by the load data in WA and HI to predict the load values in CO to evaluate the zero-shot learning ability of these models. Table IV represents the residential load forecasting results of the zero-shot experimental cases described in Section IV-A. The proposed pre-trained LLM-based BlackInter outperforms the other four methods no matter whether the model is transferred from dissimilar source data ($\mathbf{O_1}$) or similar source data ($\mathbf{O_2}$). Even with sufficient target load data for training (**Baseline_O**), the proposed method still achieves the best accuracy for residential load forecasting. Table V represents the hotel load forecasting results, where our method is the best when the prediction model is transferred from other source domain data ($\mathbf{O_1}$ and $\mathbf{O_2}$) for the zero-shot task. With sufficient target load data for training, our method is better than PatchTST and Autoformer, while underperforming STGCN and GW. In particular, under the setting $\mathbf{O_1}$ in Table IV and Table V, the MAPE error by our method is only 82.61% (11.87%/14.37%) and 69.44% (10.16%/14.63%) of the MAPE error by PatchTST in residential and hotel data, respectively, where PatchTST performs the best among other methods for comparison. The zero-shot experimental results show that even without any target load data for fine-tuning, the pre-trained LLM-based BlackInter can still maintain satisfying forecasting performance, demonstrating the generalization ability of the proposed method on the unseen task.

TABLE IV
COMPARISONS ON ZERO-SHOT RESIDENTIAL LOAD FORECASTING

Methods	From dissimilar ($\mathbf{O_1}$)		From similar ($\mathbf{O_2}$)		**Baseline_O**	
	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE
STGCN	36.79%	31.92%	13.23%	12.58%	12.21%	11.01%
GW	38.53%	35.11%	14.36%	14.64%	11.71%	11.54%
PatchTST	14.37%	12.68%	12.37%	10.79%	11.27%	9.99%
Autoformer	26.69%	23.47%	13.92%	13.11%	11.28%	10.07%
BlackInter (ours)	**11.87%**	**11.73%**	**10.74%**	**9.76%**	**10.06%**	**9.32%**

5) Seasonal impact evaluation: In the previous experiments, we evaluate the prediction models on the load data in the winter. Since the load patterns from different seasons are different, this section evaluates our method on summer data, under few-shot settings, including $\mathbf{F_1}$, $\mathbf{F_2}$, and $\mathbf{F_3}$, as well as zero-shot settings $\mathbf{O_1}$ and $\mathbf{O_2}$. Few-shot settings adopt the ten days of load data, from July 22nd to July 31st, as the target data. Zero-shot settings adopt the load data in July as the target testing data. We present the results on hotel load and skip the results on residential load as the conclusions are similar.

TABLE V
COMPARISONS ON ZERO-SHOT HOTEL LOAD FORECASTING

Methods	From dissimilar ($\mathbf{O_1}$)		From similar ($\mathbf{O_2}$)		**Baseline_O**	
	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE
STGCN	20.58%	22.29%	10.06%	12.56%	5.66%	8.74%
GW	23.57%	18.25%	8.46%	10.58%	**5.35%**	**7.71%**
PatchTST	14.63%	16.07%	11.19%	13.44%	9.99%	12.47%
Autoformer	18.88%	24.03%	12.41%	15.82%	8.18%	10.12%
BlackInter (ours)	**10.16%**	**11.92%**	**8.27%**	**9.59%**	7.89%	8.46%

As Table VI shows, our proposed LLM-based BlackInter can achieve the best forecasting performance in all few zero-shot and zero-shot settings when tested on the summer load data, which is consistent with the results obtained from the winter load data.

To illustrate how these methods capture the load trends, we draw the predicted and real one-day load values for a hotel under few-shot settings. Figs. 9-11 correspond the settings $\mathbf{F_1}$ (from scratch), $\mathbf{F_2}$ (from dissimilar source), and $\mathbf{F_3}$ (from similar source), respectively. The prediction results obtained from our method are the closest to the real load data across all the three settings. The prediction results under the setting $\mathbf{F_1}$ deviate the most from the real data. That is because the setting $\mathbf{F_1}$ has no source load data for pre-training and all the trainable parameters are trained from scratch by few target samples, which leads to insufficient parameter optimization. Even in this case, our model can still capture the trend of the real load data and shows the best prediction accuracy. The prediction results under the setting $\mathbf{F_3}$ closely align with the actual load data, particularly in the valley and peak load values observed at around 4 pm and 9 pm. That is because the setting $\mathbf{F_3}$ involves similar source load data for pre-training, which enhances the prediction performance.

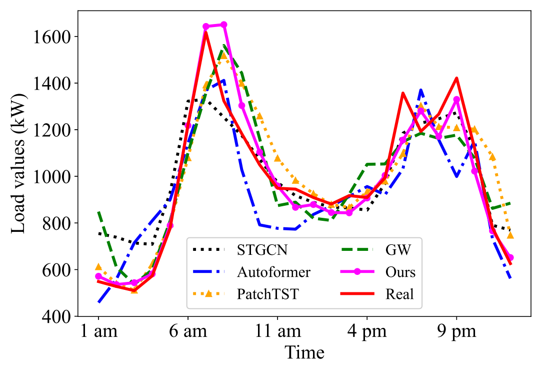


Fig. 9. Load visualization under the setting $\mathbf{F_1}$ (from scratch).

6) Inductive ability evaluation: Here, we verify the inductive ability of pre-trained LLM-based BlackInter when considering the spatial correlations among time-series load values.

TABLE VI
PREDICTION EVALUATION FOR HOTEL'S SUMMER LOAD DATA

Methods	Few-shot						Zero-shot			
	From scratch ($\mathbf{F_1}$)		From dissimilar ($\mathbf{F_2}$)		From similar ($\mathbf{F_3}$)		From dissimilar ($\mathbf{O_1}$)		From similar ($\mathbf{O_2}$)	
	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE	MAPE	NRMSE
STGCN	15.29%	17.63%	11.56%	13.28%	8.42%	9.54%	14.23%	16.43%	8.62%	9.25%
GW	15.17%	15.50%	9.60%	13.54%	7.59%	9.77%	13.04%	15.98%	8.71%	11.19%
PatchTST	15.08%	17.10%	13.44%	14.02%	9.85%	10.64%	13.58%	13.93%	10.32%	10.64%
Autoformer	22.73%	22.37%	10.21%	11.59%	8.39%	9.68%	13.82%	15.74%	9.58%	11.21%
BlackInter (ours)	**10.80%**	**9.98%**	**8.45%**	**8.33%**	**7.39%**	**7.47%**	**9.01%**	**11.44%**	**7.87%**	**8.17%**

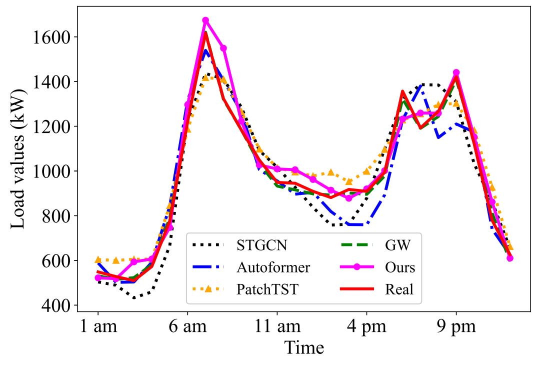


Fig. 10. Load visualization under the setting $\mathbf{F_2}$ (from dissimilar source).

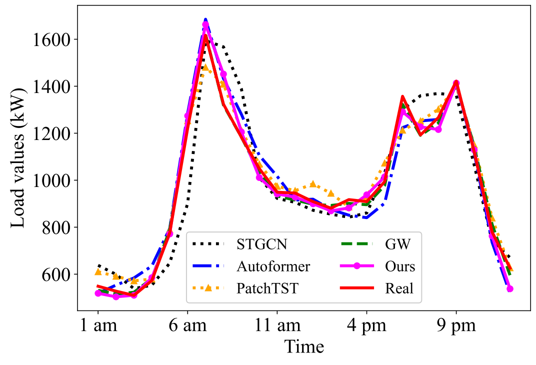


Fig. 11. Load visualization under the setting $\mathbf{F_3}$ (from similar source).

As Table VII shows, the pre-trained LLM-based BlackInter trained on the load values from ten buildings can be used to predict the load values of five buildings and fifteen buildings accurately. Hence, the pre-trained LLM-based BlackInter can handle different numbers of buildings in the training and testing stages. The prediction errors in the setting $\mathbf{I_1}$ are slightly smaller than those in the corresponding baseline setting **Baseline_I$_1$**. That is because in **Baseline_I$_1$**, the training data are from five buildings only, while the models are trained using the data from ten buildings in the setting $\mathbf{I_1}$, resulting in better performance. In contrast, the prediction performance in the setting **Baseline_I$_2$** is more accurate than that in $\mathbf{I_2}$ because there are more training data from fifteen buildings in setting **Baseline_I$_2$**.

7) Ablation study: This section evaluates the effectiveness of the spatial embedding layer, the feature extraction layer, and the optimization method Mix-Adam. Here, we follow the experiment setting $\mathbf{F_1}$ in the few-shot evaluation. To

TABLE VII
INDUCTIVE ABILITY VERIFICATION OF BLACKINTER

Experimental case	Residential		Hotel	
	MAPE	NRMSE	MAPE	NRMSE
Test on five buildings ($\mathbf{I_1}$)	10.93%	9.67%	7.46%	8.42%
Baseline_I$_1$	11.96%	9.98%	7.71%	8.85%
Test on fifteen buildings ($\mathbf{I_2}$)	9.22%	9.11%	7.84%	8.39%
Baseline_I$_2$	8.98%	8.81%	7.56%	8.37%

evaluate the spatial embedding layer and feature extraction layer, we eliminate these two layers in both the training and testing stages. To validate the effectiveness of Mix-Adam, we compare it with the white-box setting, where the parameters of BlackInter are optimized using first-order gradients and Adam (FO-Adam), requiring the knowledge of the structure and weights of the pre-trained LLM Llama-7B. From the results in Table VIII, we can see the prediction performance decreases if the spatial embedding layer or the feature extraction layer is removed, which thus demonstrates the effectiveness of the two layers. As expected, the prediction performance in the white-box setting is slightly higher than that of the pre-trained LLM-based BlackInter, but the performance gap is small. Moreover, BlackInter does not require any knowledge of the structure and parameters of the LLM model.

TABLE VIII
ABLATION STUDY OF BLACKINTER

Variants	Residential		Hotel	
	MAPE	NRMSE	MAPE	NRMSE
w/o Spatial embedding layer	10.61%	11.88%	10.41%	12.45%
w/o Feature extraction layer	10.48%	10.94%	10.37%	12.39%
White-box	9.31%	9.84%	10.05%	10.93%
BlackInter (ours)	9.87%	10.75%	10.11%	11.84%

D. Efficiency analysis

This section is to analyze the efficiency of the proposed pre-trained LLM-based BlackInter. The experiment setting is $\mathbf{F_1}$ in the few-shot evaluation described in Section IV-A. We analyze the memory consumption and computational time of the forecasting models in both the offline training and real-time testing stages in Table IX. Since the pre-trained LLM Llama-7B is accessed through the model owner's service in this paper, and only BlackInter is performed on the user side, we do not consider the memory requirement of the pre-trained LLM in our proposed method. In the offline training stage, the memory overhead of BlackInter is approximately 50% higher than that of the STGCN and GW. However, the memory usage of BlackInter is significantly lower than that of PatchTST and Autoformer. In the real-time testing stage, the memory usage of BlackInter can be comparable to STGCN and GW, and significantly lower than that of PatchTST and Autoformer. In both the offline training and real-time testing stages, the computational time of the pre-trained LLM-based BlackInter, PatchTST, and Autoformer are higher than that of STGCN and GW. That is because the pre-trained LLM-based BlackInter, PatchTST, and Autoformer involve transformer modules and take longer computational time.

TABLE IX
EFFICIENCY COMPARISONS IN THE OFFLINE AND ONLINE STAGES

Methods	Offline training		Online testing	
	Mem(MiB)	Time(s)	Mem(MiB)	Time(s)
STGCN	292	79.3	257	4.2
GW	332	87.6	259	4.37
PatchTST	1296	1250.1	1103	56.94
Autofomer	2801	527.9	1372	61.37
White-box	9688	676.7	9196	47.2
BlackInter (ours)	458	763.5	278	46.1

In addition, we also compare the efficiency of BlackInter to the pre-trained LLM-based method in the white-box setting to verify the superiority of the black-box setting. As Table IX shows, the memory overhead of the proposed BlackInter is significantly lower than that of the model in the white-box setting. That is because storing the weights of the pre-trained LLM and computing the first-order gradients takes up huge memory space. In the offline stage, the computational time of BlackInter is slightly higher than that of the model in the white-box setting. The reason is that the optimization method Mix-Adam employed in BlackInter converges a bit slower due to the estimations of gradients. However, the online inference time of BlackInter and the model in the white-box setting are comparable.

E. LLM's size and type impacts

This section evaluates the impacts of different types and sizes of pre-trained LLMs. Here, we follow the experiment setting $\mathbf{F_1}$ in the few-shot evaluation. To evaluate the impacts of different model sizes, we adopt the pre-trained Llama-3B and Llama-7B as the LLM backbone of BlackInter, respectively. Both Llama-3B and Llama-7B belong to the Llama model family [64] but differ in size. Llama-3B consists of 3.21 billion parameters, while Llama-7B comprises 7 billion parameters. In order to evaluate the impacts of different model types, we also adopt the pre-trained Phi-2 [65] and RedPajama-3B [66] as the LLM backbone, respectively. Phi-2 is developed by Microsoft and contains 2.7 billion parameters. RedPajama-3B is from RedPajama-INCITE family and contains 2.8 billion parameters.

From Table X, we can see that for the results from the same type of model Llama, the performance decreases when the size of the pre-trained model decreases from 7 billion to 3.21 billion. The model size of Phi-2 is slightly smaller than that of RedPajama-3B, but Phi-2 still outperforms RedPajama-3B in our load forecasting task. The superior performance of Phi-2 compared to RedPajama-3B may be attributed to its use of high-quality data for pre-training [65].

TABLE X
LLM'S SIZE AND TYPE IMPACTS

Model	Residential		Hotel	
	MAPE	NRMSE	MAPE	NRMSE
Phi-2	10.79%	11.16%	11.03%	11.44%
RedPajama-3B	11.11%	11.35%	11.21%	12.78%
Llama-3B	10.65%	10.86%	10.49%	12.47%
Llama-7B (ours)	9.87%	10.75%	10.11%	11.84%

V. CONCLUSION

Accurate building-level load forecasting helps to improve the stability and flexibility of power grids. Insufficient historical load data for training poses a challenge to achieving superior prediction performance. Thus, this paper develops a memory-efficient, plug-in, and inductive pre-trained LLM-based adapter, referred to as BlackInter, to empower pre-trained LLMs for building-level load forecasting, especially when few or even no historical data are available. The parameters of BlackInter are optimized by the proposed Mix-Adam using the model inference results without knowing the information of pre-trained LLMs. It can thus be implemented with memory and computation efficiency. The proposed method demonstrates its superior few-shot and zero-shot load forecasting performance over the other methods. The prediction error by our method can be only 42.88% and 69.44% of the error by the best alternative approach in the few-shot and zero-shot cases, respectively. Even with sufficient historical datasets for training, the proposed method still shows satisfying performance. Future work includes how to protect the load data privacy. After building-level load data are sent to LLMs, the owners of LLMs can utilize the data to infer the buildings' information, such as daily routine, appliance usage, and occupancy level [67], which exposes the privacy of the buildings. We plan to explore encryption techniques to resolve this issue. Another future direction is to evaluate the LLM-based BlackInter considering different demographics of building owners or homeowners.

REFERENCES

[1] J. Wang, X. Chen, F. Zhang, F. Chen, and Y. Xin, "Building load forecasting using deep neural network with efficient feature fusion," *Journal of Modern Power Systems and Clean Energy*, vol. 9, no. 1, pp. 160–169, 2021.

[2] N. Tsalikidis, A. Mystakidis, C. Tjortjis, P. Koukaras, and D. Ioannidis, "Energy load forecasting: One-step ahead hybrid model utilizing ensembling," *Computing*, vol. 106, no. 1, pp. 241–273, 2024.

[3] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE transactions on smart grid*, vol. 10, no. 1, pp. 841–851, 2017.

[4] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques," *Applied energy*, vol. 236, pp. 1078–1088, 2019.

[5] Y. Li, F. Zhang, Y. Liu, H. Liao, H.-T. Zhang, and C. Chung, "Residential load forecasting: An online-offline deep kernel learning method," *IEEE Transactions on Power Systems*, 2023.

[6] J.-W. Xiao, P. Liu, H. Fang, X.-K. Liu, and Y.-W. Wang, "Short-term residential load forecasting with baseline-refinement profiles and bi-attention mechanism," *IEEE Transactions on Smart Grid*, 2023.

[7] S. S. Pappas, L. Ekonomou, D. C. Karamousantas, G. Chatzarakis, S. Katsikas, and P. Liatsis, "Electricity demand loads modeling using autoregressive moving average (arma) models," *Energy*, vol. 33, no. 9, pp. 1353–1360, 2008.

[8] C.-M. Lee and C.-N. Ko, "Short-term load forecasting using lifting scheme and arima models," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5902–5911, 2011.

[9] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 900–911, 1971.

[10] J. Munkhammar, D. van der Meer, and J. Widén, "Very short term load forecasting of residential electricity consumption using the markov-chain mixture distribution (mcm) model," *Applied Energy*, vol. 282, p. 116180, 2021.

[11] I. Drezga and S. Rahman, "Short-term load forecasting with local ann predictors," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 844–850, 1999.

[12] J. Che and J. Wang, "Short-term load forecasting using a kernel-based support vector regression combination model," *Applied energy*, vol. 132, pp. 602–609, 2014.

[13] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2017.

[14] G. Chitalia, M. Pipattanasomporn, V. Garg, and S. Rahman, "Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks," *Applied Energy*, vol. 278, p. 115410, 2020.

[15] X. Tang, H. Chen, W. Xiang, J. Yang, and M. Zou, "Short-term load forecasting using channel and temporal attention based temporal convolutional network," *Electric Power Systems Research*, vol. 205, p. 107761, 2022.

[16] S. M. J. Jalali, S. Ahmadian, A. Khosravi, M. Shafie-khah, S. Nahavandi, and J. P. Catalão, "A novel evolutionary-based deep convolutional neural network model for intelligent load forecasting," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8243–8253, 2021.

[17] Y. Jiang, T. Gao, Y. Dai, R. Si, J. Hao, J. Zhang, and D. W. Gao, "Very short-term residential load forecasting based on deep-autoformer," *Applied Energy*, vol. 328, p. 120120, 2022.

[18] W. Lin, D. Wu, and B. Boulet, "Spatial-temporal residential short-term load forecasting via graph neural networks," *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5373–5384, 2021.

[19] H. Zhao, Y. Wu, L. Ma, and S. Pan, "Spatial and temporal attention-enabled transformer network for multivariate short-term residential load forecasting," *IEEE Transactions on Instrumentation and Measurement*, 2023.

[20] Z. Wu, Y. Mu, S. Deng, and Y. Li, "Spatial–temporal short-term load forecasting framework via k-shape time series clustering method and graph convolutional networks," *Energy Reports*, vol. 8, pp. 8752–8766, 2022.

[21] D. Wu, B. Wang, D. Precup, and B. Boulet, "Multiple kernel learning-based transfer regression for electric load forecasting," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1183–1192, 2019.

[22] Y. Lu, Z. Tian, R. Zhou, and W. Liu, "A general transfer learning-based framework for thermal load prediction in regional energy system," *Energy*, vol. 217, p. 119322, 2021.

[23] J. Zhao, Y. Yang, X. Lin, J. Yang, and L. He, "Looking wider for better adaptive representation in few-shot learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 10 981–10 989.

[24] Y. Geng, J. Chen, X. Zhuang, Z. Chen, J. Z. Pan, J. Li, Z. Yuan, and H. Chen, "Benchmarking knowledge-driven zero-shot learning," *Journal of Web Semantics*, vol. 75, p. 100757, 2023.

[25] Z. Zhang, P. Zhao, P. Wang, and W.-J. Lee, "Transfer learning featured short-term combining forecasting model for residential loads with small sample sets," *IEEE Transactions on Industry Applications*, vol. 58, no. 4, pp. 4279–4288, 2022.

[26] D. Wu and W. Lin, "Efficient residential electric load forecasting via transfer learning and graph neural networks," *IEEE Transactions on Smart Grid*, vol. 14, no. 3, pp. 2423–2431, 2022.

[27] C. Peng, Y. Tao, Z. Chen, Y. Zhang, and X. Sun, "Multi-source transfer learning guided ensemble lstm for building multi-load forecasting," *Expert Systems with Applications*, vol. 202, p. 117194, 2022.

[28] J. Lin, J. Ma, and J. Zhu, "A privacy-preserving federated learning method for probabilistic community-level behind-the-meter solar generation disaggregation," *IEEE Transactions on Smart Grid*, vol. 13, no. 1, pp. 268–279, 2021.

[29] M. Ganjouri, M. Moattari, A. Forouzantabar, and M. Azadi, "Spatial-temporal learning structure for short-term load forecasting," *IET Generation, Transmission & Distribution*, vol. 17, no. 2, pp. 427–437, 2023.

[30] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[31] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models (2023)," *arXiv preprint arXiv:2302.13971*, 2023.

[32] 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*, vol. 36, 2024.

[33] 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*, vol. 36, 2024.

[34] 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.

[35] 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.

[36] T. Wang, A. Roberts, D. Hesslow, T. Le Scao, H. W. Chung, I. Beltagy, J. Launay, and C. Raffel, "What language model architecture and pretraining objective works best for zero-shot generalization?" in *International Conference on Machine Learning*. PMLR, 2022, pp. 22 964–22 984.

[37] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[38] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, "Fine-tuning language models with just forward passes," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[39] T. Sun, Y. Shao, H. Qian, X. Huang, and X. Qiu, "Black-box tuning for language-model-as-a-service," in *International Conference on Machine Learning*. PMLR, 2022, pp. 20 841–20 855.

[40] C. Oh, H. Hwang, H.-y. Lee, Y. Lim, G. Jung, J. Jung, H. Choi, and K. Song, "Blackvip: Black-box visual prompting for robust transfer learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 24 224–24 235.

[41] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[42] N. Mohan, K. Soman, and S. S. Kumar, "A data-driven strategy for short-term electric load forecasting using dynamic mode decomposition model," *Applied energy*, vol. 232, pp. 229–244, 2018.

[43] W. Peng, J. Yi, F. Wu, S. Wu, B. Zhu, L. Lyu, B. Jiao, T. Xu, G. Sun, and X. Xie, "Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark," *arXiv preprint arXiv:2305.10036*, 2023.

[44] Y. Fathullah, C. Wu, E. Lakomkin, J. Jia, Y. Shangguan, K. Li, J. Guo, W. Xiong, J. Mahadeokar, O. Kalinli *et al.*, "Prompting large language models with speech recognition abilities," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 13 351–13 355.

[45] B. Zheng, J. Gu, S. Li, and C. Dong, "Lm4lv: A frozen large language model for low-level vision tasks," *arXiv preprint arXiv:2405.15734*, 2024.

[46] M. Huh, B. Cheung, T. Wang, and P. Isola, "The platonic representation hypothesis," *arXiv preprint arXiv:2405.07987*, 2024.

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

[48] U. B. Filik and M. Kurban, "A new approach for the short-term load forecasting with autoregressive and artificial neural network models," *International Journal of Computational Intelligence Research*, vol. 3, no. 1, pp. 66–71, 2007.

[49] J. Park, Y. Park, and K. Lee, "Composite modeling for adaptive short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 6, no. 2, pp. 450–457, 1991.

[50] R. Gunjal, S. S. Nayyer, S. Wagh, A. Stankovic, and N. Singh, "Granger causality for prediction in dynamic mode decomposition: Application to

power systems," *Electric Power Systems Research*, vol. 225, p. 109865, 2023.

[51] W. Li and M. Wang, "Identifying overlapping successive events using a shallow convolutional neural network," *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 4762–4772, 2019.

[52] C. Shi, H. Yang, D. Cai, Z. Zhang, Y. Wang, Y. Yang, and W. Lam, "A thorough examination of decoding methods in the era of llms," *arXiv preprint arXiv:2402.06925*, 2024.

[53] M. Nguyen, A. Baker, C. Neo, A. Roush, A. Kirsch, and R. Shwartz-Ziv, "Turning up the heat: Min-p sampling for creative and coherent llm outputs," *arXiv preprint arXiv:2407.01082*, 2024.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[55] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[56] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, "Fine-tuning language models with just forward passes," *Advances in Neural Information Processing Systems*, vol. 36, pp. 53 038–53 075, 2023.

[57] M. C. Choy, D. Srinivasan, and R. L. Cheu, "Simultaneous perturbation stochastic approximation based neural networks for online learning," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*. IEEE, 2004, pp. 1038–1044.

[58] Y.-Y. Hong, H.-L. Chang, and C.-S. Chiu, "Hour-ahead wind power and speed forecasting using simultaneous perturbation stochastic approximation (spsa) algorithm and neural network with fuzzy inputs," *Energy*, vol. 35, no. 9, pp. 3870–3876, 2010.

[59] N. Dong, X.-S. Han, Z.-K. Gao, Z.-Q. Chen, and A.-G. Wu, "Spsa-based data-driven control strategy for load frequency control of power systems," *IET Generation, Transmission & Distribution*, vol. 12, no. 2, pp. 414–422, 2018.

[60] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[61] Y. Hu, X. Cheng, S. Wang, J. Chen, T. Zhao, and E. Dai, "Times series forecasting for urban building energy consumption based on graph convolutional network," *Applied Energy*, vol. 307, p. 118231, 2022.

[62] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22 419–22 430, 2021.

[63] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

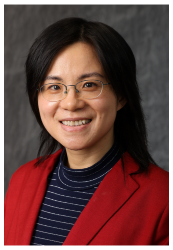
[64] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[65] M. Javaheripi, S. Bubeck, M. Abdin, J. Aneja, S. Bubeck, C. C. T. Mendes, W. Chen, A. Del Giorno, R. Eldan, S. Gopi *et al.*, "Phi-2: The surprising power of small language models," *Microsoft Research Blog*, vol. 1, p. 3, 2023.

[66] M. Weber, D. Y. Fu, Q. Anthony, Y. Oren, S. Adams, A. Alexandrov, X. Lyu, H. Nguyen, X. Yao, V. Adams *et al.*, "Redpajama: an open dataset for training large language models."

[67] J. Lei, L. Wang, Q. Pei, W. Sun, X. Lin, and X. Liu, "Privgrid: Privacy-preserving individual load forecasting service for smart grid," *IEEE Transactions on Information Forensics and Security*, 2024.

Yating Zhou (Graduate Student Member, IEEE) received the B.E. degree in Electrical Engineering and its Automation from Beijing Jiaotong University, China, in 2017, and the M.S. degree in Electrical Engineering from Tsinghua University, China, in 2020. She is currently working toward the Ph.D. degree in electrical engineering at Rensselaer Polytechnic Institute, Troy, NY, USA. Her research interests include load monitoring, spatial-temporal data analytics, and machine learning.



Meng Wang (Senior Member, IEEE) received B.S. and M.S. degrees from Tsinghua University, China, in 2005 and 2007, respectively. She received the Ph.D. degree from Cornell University, Ithaca, NY, USA, in 2012. She is an Associate Professor in the department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY, USA, where she joined in Dec. 2012. Before that, she was a postdoc scholar at Duke University, Durham, NC, USA. Her research interests include high-dimensional data analytics, machine learning and artificial intelligence, power systems monitoring, and synchrophasor technologies. She serves as an Associate Editor for IEEE Transactions on Smart Grid and IEEE Transactions on Signal Processing.