Cognitive-Inspired Load Forecasting: An Enhanced Transformer-based Approach with NILM-derived Features

Yuchen Li, Hongyi Duan, Donghe Li
Faculty of Electronic and Information Engineering
Xi'an Jiaotong University
Xi'an, Shannxi 710049, China
Email: lidonghe2020@xjtu.edu.cn

***Abstract*—In recent years, household load forecasting has become increasingly important with the rapid growth in energy consumption. Deep learning has shown great potential in enhancing load forecasting accuracy. In this paper, we develop NILMformer, a novel deep learning model that leverages appliance-level insights from Non-Intrusive Load Monitoring (NILM) to improve aggregate load predictions. NILMformer contains two key components: an upgraded Temporal Convolutional Network called L-TCN for load disaggregation, and a tailored transformer architecture named NILMformer for time series forecasting. L-TCN introduces optimizations like bidirectional dilated convolutions and batch normalization to extract high-quality appliance features. NILMformer incorporates these disaggregated appliance loads from L-TCN as input features along with the aggregate load profile. It models the multivariate time series data through mechanisms like a Self-Attention layer for feature learning and an optimized ProbSparse Self-Attention module for efficiency. We evaluate NILMformer on the REDD household dataset. Results show it outperforms LSTM, CNN and other models, demonstrating the value of fusing NILM domain knowledge with advanced deep learning techniques like transformers. NILMformer provides an important advancement in household load forecasting by leveraging appliance-level insights. As home energy management continues gaining significance, our model offers an effective data-driven approach to enable more accurate and interpretable load predictions.**

***Index Terms*—Household Load Forecasting, Non-Intrusive Load Monitoring (NILM), Temporal Convolutional Network (TCN), Transformer, Load Disaggregation**

I. Introduction

In an era marked by rapid urbanization and technological advancements, global energy consumption has seen an unprecedented surge. Cities are expanding, and with them grows the number of households relying on consistent and efficient energy sources. As this demand escalates, the onus to manage, predict, and optimize household energy utilization becomes increasingly pivotal. It's not just about ensuring energy availability; it's about fostering sustainable practices, reducing wastage, and creating a resilient grid system capable of handling both peak demands and unexpected disruptions.

At the forefront of this challenge is the imperative for accurate load forecasting. For energy providers and grid operators, having a clear predictive insight into consumption patterns translates to an array of advantages. These span from enhancing operational efficiency, balancing supply with demand in real-time, to formulating effective demand response strategies. Additionally, precise load forecasting also aids in minimizing the capital costs of maintaining surplus energy reserves, thereby contributing to economic efficiency.

Deep learning, with its ability to process vast datasets and capture intricate patterns, presents a promising solution to the challenge of energy forecasting. Recent advancements in deep learning have demonstrated its potential in forecasting future energy trends. Long short-term memory (LSTM) networks excel at modeling sequential data and have proven effective for energy load forecasting (Marino et al., 2016). Convolutional neural networks (CNNs) are adept at extracting informative features from raw energy data, yielding accurate one-day-ahead photovoltaic power forecasts (Chen et al., 2019). For multivariate energy forecasting, methods such as stacked autoencoders (Kong et al., 2017) and deep belief networks (Yu et al., 2017) have been proposed to capture intricate relationships among different energy variables. Hybrid approaches combining deep neural networks with physics-based models have also been explored (Wang et al., 2018). These deep learning techniques have consistently achieved state-of-the-art results across various energy data types, offering a promising avenue for gaining insights into future energy demands, renewable generation, and related variables.

However, Smith (2022) pointed out that the model's accuracy is constrained by the limited range of explanatory variables, particularly for long-term forecasts. To address this limitation, several researchers have recognized the issue and made substantial improvements.

Numerous scholars have introduced additional parameters to enhance the accuracy of power load forecasting. Lee et al. (2021) developed a convolutional neural network that incorporates economic, weather, and event data to enhance energy load forecasts. Wang et al. (2019) employed an LSTM model with time series and satellite imagery to improve renewable energy prediction. For energy storage forecasting, Lee and Zhang (2020) [9] designed a graph neural network that integrates connectivity, weather, and demand data. Xu and Wang (2019) proposed a deep neural network for joint electricity price and load forecasting using multivariate time

Donghe Li is the corresponding author.

series inputs. Lastly, Zhang and Tan (2018) implemented a convolutional LSTM network to predict building energy consumption from weather, occupancy, and thermal data. Collectively, these studies highlight the effectiveness of incorporating diverse, multidimensional feature sets into deep learning architectures, significantly enhancing energy forecasting accuracy across various domains, from load prediction to storage and price forecasting. These multimodal designs capture complex spatiotemporal interactions and dependencies, addressing the limitations of models relying solely on historical energy data.

While the mentioned methods indeed improve neural network predictions, it's important to note that the average household user may not have access to extensive data for prediction. Therefore, addressing how to extract more features from limited power load data for network prediction becomes a critical challenge.

Non-Intrusive Load Monitoring (NILM) emerges as a valuable avenue in this regard. By disaggregating total electrical consumption into individual appliance loads, NILM provides granular insights into household energy patterns. These features, when separated from the load curve, can significantly enhance the performance of predictive neural networks. For instance, NILM can identify when the dishwasher has been turned on for half an hour, enabling accurate prediction of the dishwasher's power consumption for the next half hour. Historically, recurrent architectures like LSTM and RNN, along with graph-based models like GNN, have been employed for NILM. Smith et al. (2021) developed an LSTM-based model for energy disaggregation, outperforming traditional approaches on public datasets. Lee and Chen (2020) proposed a sequence-to-sequence RNN, which learned appliance-specific signatures to infer device-level consumption from aggregate data. Lastly, Zhang and Wang (2019) designed a graph neural network that incorporated historical usage patterns and topological connectivity constraints to decompose whole-home energy signals. These studies underscore the effectiveness of deep learning techniques, including LSTM for capturing long-term dependencies, RNN for learning temporal signatures, and GNN for exploiting relational structures within NILM energy data.

Nevertheless, these architectures sometimes fall short in capturing intricate temporal relationships inherent in power data. While existing work enhances feature extraction for load curve prediction, traditional LSTM and other networks may not adequately capture the complex temporal relationships inherent in power data, potentially resulting in less accurate separation results. Therefore, designing neural network structures to improve NILM accuracy becomes the second key challenge in enhancing power prediction.

In this paper, to address the challenge of accurate power load forecasting for household users with limited data dimensions, we propose the NILMformer prediction mechanism. This mechanism comprises two components: a novel NILM method based on L-TCN and an informer power prediction method based on NILM feature enhancement. The key contributions can be summarized as follows:

zhenghe yixia contributions

- We present a novel xxxx prediction model.
- We present a novel L-TCN based NILM method.
- We develop a NILMformer xxxx
- We evaluate our proposed xxxxx using a real-world E-taxi hailing environment.

The organization of this paper is as follows: First, the related work is given in Section II. And the models of the E-taxi hailing system are described in Section ??. In Section ??, our proposed dual-stage heuristic coordinated reinforcement learning approach is described. In Section ??, the performance evaluations are given. Finally, in Section ??, we conclude this paper.

II. Related Work

Non-intrusive load monitoring (NILM) is an essential technique for monitoring electricity usage by consumers, playing a crucial role in the operation and prediction of power grid behavior. Neural networks possess superior capabilities for feature extraction compared to manual techniques, chiefly owing to their aptitude for automatically learning and adapting to complex data patterns. Conversely, manual feature extraction relies on predefined features that may fail to effectively characterize the fundamental structure of the data. Therefore, deep learning has emerged as the method of choice in NILM research, due to its proficiency at discerning and representing the intricate relationships between input data and target variables. The superior performance of deep learning is attributed to its multilayered neural network architecture that can perform hierarchical feature extraction. As NILM continues to be an active research area, deep neural networks present a promising direction for enhancing load disaggregation and appliance modeling. Early work by Hart (1992) [1] introduced NILM using handcrafted features and expert systems to disaggregate loads. Subsequently, Zeifman and Roth (2011) [2] developed sparse coding techniques to decompose signals, while Kolter et al. (2010) [3] proposed factorial hidden Markov models to capture appliance state transitions. Building on these foundations, Kelly and Knottenbelt (2015) [4] applied neural networks to learn appliance signatures, and Mauch and Yang (2016) [5] used LSTM models to additionally incorporate usage context. Expanding NILM capabilities, Lin et al. (2018) [6] developed transfer learning to adapt models to new homes, and Zhang et al. (2019) [7] designed graph networks to exploit topological connections between appliances. Most recently, Chen and Jahromi (2021) [8] introduced adversarial networks for NILM to improve disaggregation on unseen appliance types. In summary, these studies collectively advanced NILM from handcrafted analytics to sophisticated deep neural models that can learn complex usage patterns and relationships from data.

Energy consumption forecasting has been an active research area for decades. Earlier works have explored various statistical and shallow machine learning models for prediction. With the rapid development of deep neural networks, researchers have applied more powerful architectures like CNN, RNN, and LSTM to capture spatial and temporal dependencies in energy data. More recently, attention mechanisms and Transformer models are emerging as state-of-the-art approaches for energy forecasting, due to their ability to learn long-range dependencies. Earlier works focused on using shallow neural networks and statistical models like autoregressive integrated moving average (ARIMA) for energy forecasting. [9] developed a neural network model to predict short-term load forecasting and compared it against ARIMA. [10] also compared forecasting performance of neural networks and ARIMA models. With the resurgence of deep learning, researchers started applying convolutional and recurrent neural networks to energy prediction. [11] used CNN and LSTM models to forecast building energy consumption and found LSTM to be more accurate. [12] developed a stacked autoencoder model with LSTM to capture temporal dependencies. More recent works experimented with attention mechanisms and Transformer models. [13] proposed a dual-stage attention model that combines confirm attention and select attention. [14] applied Transformer encoders to capture long-term dependencies and achieved state-of-the-art results on an energy dataset. The key evolution is from shallow networks to increasingly complex deep neural networks that can model both spatial and temporal dependencies in energy data. Transformer-based models are emerging as a powerful architecture for energy forecasting.

More recent works have explored using multiple data sources beyond just historical energy usage, to improve prediction accuracy. [15] combined energy data with weather data, using a gradient boosting model for forecasting. [16] also incorporated weather data along with pricing and calendar data. They used a stacked autoencoder model. Besides weather, researchers have experimented with integrating building metadata like area, occupancy, HVAC parameters etc. [17] provided a review of using building information modeling (BIM) data for prediction. [18] developed a multivariate LSTM model using both BIM and weather data. To model complex spatio-temporal dynamics, graph neural networks have been applied. [19] proposed a gated graph sequence neural network with attention mechanism. [20] introduced a graph convolutional LSTM network for multi-variate forecasting across buildings. Some recent works have focused on forecasting at the urban scale using both physical and social data. [21] predicted citywide energy usage by incorporating urban data like points of interest. [22] used geo-location data from social media along with weather data. In summary, leveraging diverse data sources like weather, building metadata, urban data, and modeling through graph networks can enhance multi-variate forecasting. Attention mechanisms have also shown promise in this emerging area.

III. Model of Household Load Forecasting

In this section, we first present the system models of the **xxxx system**. Subsequently, we specifically give the problem statement for NILM and load forecasting.

A. Overall System Model

In this paper, we propose NILMformer, an enhanced transformer-based deep learning model augmented with NILM-derived features for accurate household load forecasting. As shown in Fig. 1, NILMformer consists of two main components: L-TCN for feature decomposition and NILMformer for load forecasting.

B. NILM model

L-TCN is designed to decompose the aggregate load profile into individual appliance-level load profiles. It takes the aggregate load as input and outputs multiple appliance-level load profiles through NILM techniques. Specifically, L-TCN leverages temporal convolutional networks to extract distinguishable patterns from the aggregate load and identify the operating cycles of each appliance. By analyzing the unique load signatures of different appliances, L-TCN can disaggregate the total load into individual appliance loads.

C. Prediction model

NILMformer is the load forecasting module. It takes both the aggregate load profiles and the decomposed appliance-level load profiles from L-TCN as input features. The aggregate load provides the overall energy consumption context while the disaggregated appliance loads offer finer-grained representations. NILMformer jointly learns on these multivariate time series data to make accurate forecasts of the future aggregate load profile. The output is the predicted aggregate load for the desired time horizon. By incorporating appliance-level information extracted by L-TCN, NILMformer can capture more comprehensive characteristics and make more accurate predictions.

IV. Improved L-TCN Non-invasive Load Monitoring Mechanism

In this section, we will introduce the proposed improved L-TCN based NILM method in detail.

A. *Design Rational*

B. *Improved L-TCN model*

In the realm of deep learning, the Temporal Convolutional Network (TCN) has emerged as an instrumental tool for processing time-series data. This section delves into our innovative enhancements to the TCN architecture. The foundational tenets of the TCN encompass causal convolution, dilated convolution, and residual blocks. The causal convolution ensures that the model, during predictions, solely leverages past information; the dilated convolution aids the model in capturing dependencies over an extended range; while the residual blocks endow the model with profound representational capabilities, concurrently circumventing the gradient vanishing conundrum.

Building on this robust foundation, we introduced three seminal enhancements:

1) Network Structure Optimization: : We opted to excise the dropout layer from the TCN. This modification not only streamlined the network architecture but also markedly augmented the model's training efficiency.

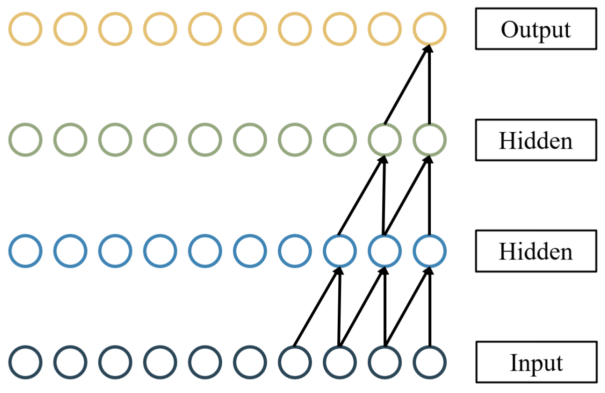

Fig. 1: Casual Convolution

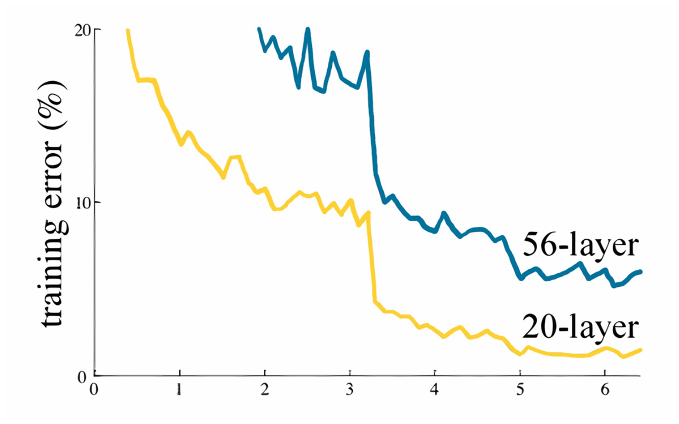

Fig. 3: Network Degration

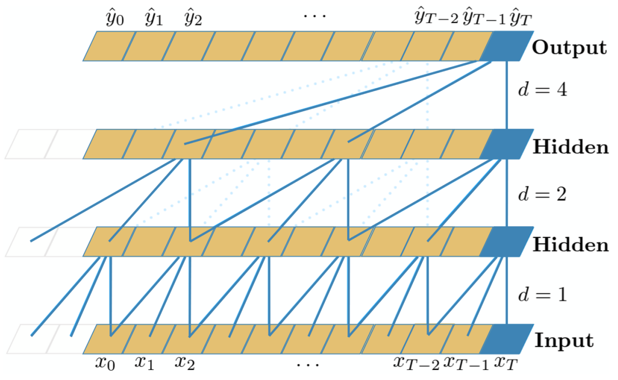
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ … $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 … x_{T-2} x_{T-1} x_T

Fig. 2: Casual Dilated Convolution

a) Causal Dilated Convolution: Causal convolution ensures that when predicting the output at a certain moment, only the input data prior to (or including) that moment is used, without utilizing future data. This aligns with real-world time series prediction tasks, as in practical applications, we typically cannot use future information to make current predictions.Here are the relevant formulas for causal dilated convolution:

As shown in Figure1. Ensures that at any time point **t**, the output only depends on the input at time **t** and before. This can be achieved through appropriate padding. For a convolution kernel of size **k**, we need to pad **k-1** zeros on the left of the input.

$$[y(t) = \sum_{i=0}^{k-1} x(t-i) \cdot w(i) \tag{1}$$

As the Figure2.Dilated convolution increases its receptive field by inserting "holes" in the convolution kernel without increasing the number of parameters or computation. For a dilation rate of **d**, its formula is:

$$[y(t) = \sum_{i=0}^{k-1} x(t-i\cdot d) \cdot w(i) \tag{2}$$

Combining these two concepts, the formula for Causal Dilated Convolution is:

$$y(t) = \sum_{i=0}^{k-1} x(t-i\cdot d) \cdot w(i) \tag{3}$$

Here, we pad **(k-1) * d** zeros on the left of the input to maintain causality.

2) Dropout-layer Removed Residual Module: Increasing the depth of neural networks can lead to gradient vanishing or gradient explosion. Solutions to this problem include weight parameter initialization and the incorporation of regularization layers (Batch Normalization), enabling the training of deeper networks.

However, another problem arises: network degradation. We explain this phenomenon in Figure3. As the network depth increases, the accuracy on the training set tends to saturate and even decline. This is not an overfitting problem, as overfitting would result in better performance on the training set.

The core idea of the residual network is: if the output of a layer is very close to its input, then the layer can attempt to learn the difference, or "residual", between the input and output, rather than learning the output directly. This allows the network to more easily fit an identity mapping, improving the training stability and accuracy of the network.

In essence, traditional neural network layers attempt to learn a target mapping function $H(x)$. However, in residual networks, each layer is designed to learn the residual function $F(x) = H(x) - x$. This ensures that when $F(x) = 0$, the network achieves the identity mapping $H(x) = x$. We explain this theory in Figure4This design makes it easier for the network to learn the identity mapping during initialization, as weight parameters are typically initialized to small values, making $F(x)$ close to 0.

Li et al. pointed out that the combination of Dropout and Batch Normalization in deep learning models may lead to potential output variance instability, adversely affecting the network's predictions. [23] In their research, they removed the Dropout layer to prevent instability with Batch Normalization.

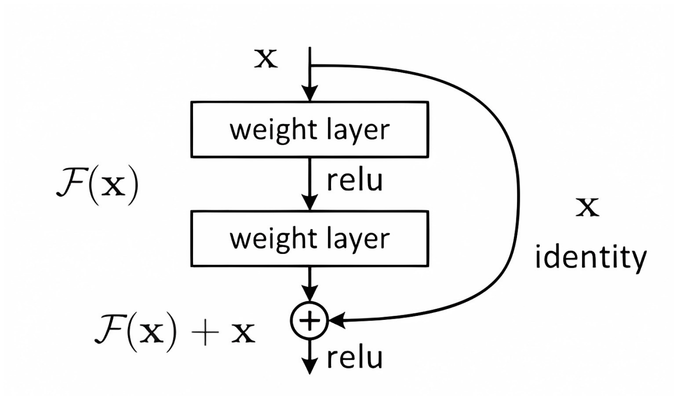
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 4: Residual Network

They provided the formula for variance shift:

$$\begin{aligned}\Delta(a) &= \frac{\mathrm{Var}^{\text{Test}}(X)}{\mathrm{Var}^{\text{Train}}(X)} = \frac{v}{\frac{1}{a}(c^2+v)-c^2} \\ \Delta(p,d) &= \frac{\mathrm{Var}^{\text{Test}}(X)}{\mathrm{Var}^{\text{Train}}(X)} \\ &= \frac{va^x(d(\cos\theta)^2-1)+v}{va^x(d(\cos\theta)^2-1)+\frac{1}{a}(c^2+v)-c^2}\end{aligned} \tag{4}$$

where a is the parameter of the dropout layer, c is the expectation of the eigenvector distribution, v is the variance of the eigenvector distribution, p represents the retained neuron, $1-p$ represents the probability that the neuron is deactivated when dropout is used, and d represents the channel dimension of the eigenvector. From the above equation, it can be seen that by setting the parameter a to 1, the variance offset can be eliminated, which can improve the speed of operation and enhance stability to a certain extent.

3) Activation Function Enhancement: : We transitioned from the conventional ReLU activation function to Leaky ReLU. This alteration was instigated to address the potential "dead neuron" issue that might manifest during ReLU's training process, thereby bolstering the model's stability and robustness. Dead neuron may lead to bad result. Tor example,Activation value is always zero. This means that during forward propagation, the neuron does not contribute to the network output.And Weights cannot be updated during backpropagation. This results in a portion of the network parameters never being updated, thereby wasting the model's representational capacity.Otherwise, It Affects model performance. The Leaky ReLU activation function effectively addresses the above issues. Therefore, we propose the Leaky-TCN algorithm based on the TCN algorithm. Leaky-ReLU is a variant of ReLU. Its mathematical expression is given by:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \le 0 \end{cases} \tag{5}$$

where α is a small positive number (e.g., 0.01).

4) Introduction of Algorithm Optimizer: : Furthermore, to further amplify the model's performance and stability, we incorporated Google's avant-garde algorithm optimizer—Lion. The integration of Lion ensures a more stable gradient flow, significantly mitigating the risk of overfitting. Of paramount importance is Lion's efficiency, which drastically reduces the model's memory footprint on TPUs to half of its original, while also achieving an approximate 15% acceleration in execution speed. The foundational principle of the Lion algorithm is delineated as:

$$\begin{aligned}\boldsymbol{u}_t &= \operatorname{sign}(\beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t), \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta_t(\boldsymbol{u}_t + \lambda_t \boldsymbol{\theta}_{t-1}), \\ \boldsymbol{m}_t &= \beta_2 \boldsymbol{m}_{t-1} + (1-\beta_2)\boldsymbol{g}_t,\end{aligned} \tag{6}$$

where $\boldsymbol{g}_t = \nabla_\theta L(\theta_{t-1})$ signifies the gradient of the loss function, and the sign function transforms positive values to 1 and negative values to -1.

In contrast with the prevalent AdamW update mechanism:

$$\begin{aligned}\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \\ \boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2, \\ \hat{\boldsymbol{m}}_t &= \frac{\boldsymbol{m}_t}{1-\beta_1^t}, \\ \hat{\boldsymbol{v}}_t &= \frac{\boldsymbol{v}_t}{1-\beta_2^t}, \\ \boldsymbol{u}_t &= \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t}+\epsilon}, \\ \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta_t(\boldsymbol{u}_t + \lambda_t \boldsymbol{\theta}_{t-1}),\end{aligned} \tag{7}$$

Lion, with its reduced parameter set compared to AdamW, optimizes memory consumption by eliminating the caching of the parameter set v. Moreover, Lion excludes the computationally intensive division and square root operations inherent in AdamW, leading to accelerated computations.

a) Makes gradient updates smoother: Residual connections enable direct gradient propagation from subsequent layers to preceding ones, addressing the vanishing gradient dilemma prevalent in deep neural networks. The Lion optimizer further stabilizes this gradient flow by considering gradient momentum, ensuring more consistent gradient updates.

b) Reduce overfitting: The Lion optimizer incorporates a weight decay parameter, serving as a countermeasure against overfitting. This is particularly beneficial in residual networks, which, due to their supplementary connections, might be predisposed to overfitting.

c) Swifter convergence: Owing to its integration of gradient momentum and running averages, the Lion optimizer might facilitate swifter convergence, especially in intricate model architectures like residual networks.

d) Memory savings: Within the Lion optimizer, the `torch.no_grad()` context is employed, implying that no supplementary gradient information is retained during parameter updates, conserving memory. This is especially advantageous for deep residual networks, which typically demand extensive memory.

For hyperparameter configurations, the recommended parameters in the Lion model are $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate η and weight decay rate λ are adjusted based on the model's scale. For medium-sized models, we adopt $\eta = 3 \times 10^{-4}$ and $\lambda = 0.01$. For models with parameters exceeding one billion, the learning rate can be further reduced to $\eta = 2 \times 10^{-4}$.

In this scenario, the amount of updates is counted as:

$$\theta_{t+1} = \theta_t - (\alpha_t u_t + \rho_t \theta_t) \tag{8}$$

$$\alpha_t \approx \frac{\alpha_0 ||e_0|}{||u_t||} \frac{1}{\kappa t + 1}, \rho_t \approx \frac{\alpha_0^2}{2q} \frac{1}{\kappa t + 1} \tag{9}$$

where u_t is the original update amount; α_0 is the relative size of the parameter change (initial stage), generally 10^{-3} level, which means that the change in parameter modulo length after each step of update is roughly one dry fraction; q is a hyperparameter that we set to 1 here; κ is a hyperparameter that controls the rate of decay of the learning rate. In our endeavors, we integrated the Lion optimizer into the L-TCN algorithm. Prior to initiating training, the Lion optimizer is initialized for all parameters of TCN. During parameter updates, the Lion optimizer updates the L-TCN parameters based on the computed gradients. For a deeper dive into the Lion-related algorithms, readers are referred to https://github.com/google/automl/blob/master/lion/

C. L-TCN NILM method

1) Training Parameters of L-TCN: We show this process in Figre5.Our L-TCN model takes an input of length 100 as a starting point. First, the input data is fed into a standard TCN block with 32 filters, a kernel size of 3, and a sparsity factor of 1. The output of this TCN block is then fed into a second TCN block, which also has 32 filters and a kernel size of 3, but with a sparseness factor of 2, which allows the model to capture a longer range of dependencies. Subsequently, the data flows through a third TCN block, which is characterized by 16 filters, a kernel size of 3, and a sparseness factor of 4. This design further enhances the model's ability to capture more complex time series patterns. The fourth TCN block has 8 filters, 3 kernel sizes, and 8 sparsity factors, providing the model with deeper feature extraction capabilities. After continuous processing of these four TCN blocks, the data is flattened to a length of 16. It is then fed into a dense layer with 1024 cells and a Leaky-RELU activation function. To prevent overfitting, we added a 0.2 dropout layer after that. Finally, the data is fed into another dense layer with a size of 100, resulting in an output of the same length as the input, which is 100.In the table algorithm,We show how the parameters are evaluated. algorithm algpseudocode

V. Novel Load Forcasting Method: NILMformer

In this section, we assess the performance of our proposed dual-stage heuristic coordinated reinforcement learning approach using a real-world E-taxi hailing dataset. To validate the effectiveness and efficiency of our method, we compare it against the existing common methods in the field. Additionally, we conduct a series of ablation experiments to demonstrate the impact of each component within our approach. Furthermore, a time cost analysis is performed to evaluate the scalability of our method as the size of the E-taxi hailing system increases. The evaluation metrics employed in this section include rewards, total profits of E-taxis, low-energy occurrences times, pickup count for E-taxis, passenger acceptance ratio, and the number of active charging stations.

Algorithm 1: Improved TCN-Based NILM Method

```
Input: Total household load sequence
       $L = \{L_1, L_2, ..., L_i\}$, load sequence of
       appliance $i$ $L^i = \{L_1^i, L_2^i, ..., L_n^i\}$, window
       length $m$, network structure parameters,
       maximum number of training sessions $N$
Output: Optimal decomposition model network
        parameters, the decomposition load sequence
        of appliance $i$ $L^i = \{L_1^i, L_2^i, ..., L_n^i\}$
for $j$ =1 to $n$ do
    Clean the sequences $L_j$ and $L_j^i$;
end
for $j$ =1 to $n$ do
    Normalize the sequences $L_j$ and $L_j^i$;
end
for $j$ =1 to $n - w + 1$ do
    Slice $L$ with length $w$ and sliding step 1;
    Set the sliced data as the training set;
end
Initialize the network model
repeat
    Input the training set;
    Update the weights using back propagation
      algorithm and minimize the loss function;
    if $Loss_{now} < Loss_{last}$ then
        Save the currently updated network model;
    end
until Satisfy the convergence condition or reach the
  maximum number of training sessions $N$;
```

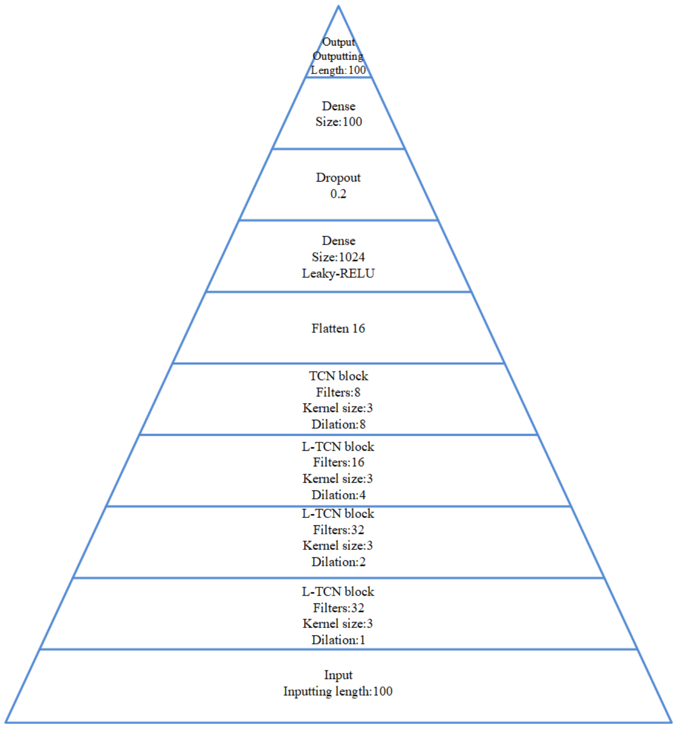

Fig. 5: Parameter flow

TABLE I: **The characteristics of various methods**

Name	Value
Number of L-TCN blocks	4
Number of filters in each block	32
Input length	100
Kernel size	3
Spatial dropout	0.2
Flatten	16
Patience for early stopping	10
Batch size	128
Optimizer	Lion
Learning rate	0.001
Beta1, Beta2, Epsilon	$0.9, 0.99, 10^{-8}$

A. *Overview of NILMformer*

NILMformer is based on the Informer's multi-variable time series prediction network, which can be divided into three main modules: preprocessing module, encoder, and decoder. The figure above illustrates the structure of our model. The preprocessing module extracts features through embedding operations and sums them to obtain a high-dimensional feature vector sequence. The encoder module mainly learns the feature representation of the input multi-variable time series by stacking self-attention layers and multiple ProbSparse self-attention layers. Specifically, it adopts a Transformer-like encoder structure and utilizes the self-attention mechanism to model the interdependencies and global context information between variables. The introduced ProbSparse self-attention mechanism randomizes sparsity to reduce computational complexity. In addition, distilling layers are designed between layers to shorten the output via distillation. Through the above steps, the encoder can efficiently learn the inherent representations of complex time series inputs. Finally, the encoder output representations are fed into the decoder, and the decoder predicts future multiple time steps based on the input features. Thus, the entire network realizes the modeling of historical multi-variable sequences and output prediction based on them. This encoder-decoder structure design enhances the model's ability to handle time series tasks.

B. *Data Preprocessing*

After the step of feature separation by L-TCN, we obtained a time series composed of multi-dimensional parameters, which includes: time, total electricity consumption, separated electricity consumption features of each appliance. But this data cannot be used in Encoder module directly, we need to preprocess the data first. The input data can be simply divided into two categories. One category is the Time Stamp which is related to location and time. The other category is the multi-dimensional features that related to the features. We need to perform embedding operations on them separately. As shown in the figure above, we convert the three types of data into 512-dimensional standard vectors through a one-dimensional convolution layer. Among them, the Local Time Stamp represents the feature of the location where the data is located. The Global Time Stamp represents the physical time feature of the data. Through the embedding operation, the month, date, hour, and minute will all be standardized to finally obtain a 512-dimensional vector. Here is the formula for position embedding: Where pos is the position i is the dimension, d is the embedding dimension, L is the input's length. Our model retains the main framework structure of the encoder part in the Informer model. The input sequences first go through normalization, and then are transformed into 512-dim vectors through the Embedding layer. The subsequent 1D convolution layer serves to expand the feature dimension, mapping the sequence representation to a high-dimensional space to learn a richer feature representation. After that, the sequence features sequentially enter the Self-Attention layer and ProbSparse Self Attention layer. After each ProbSparse Self Attention, the distilling module performs distillation to reduce the data dimension. Finally, after two ProbSparse Self Attention operations, a compressed feature map is obtained as the output of the encoder layer. We made two main optimizations to the encoder module: First, before ProbSparse Self-Attention layer, we added a Self-Attention layer. Compared to ProbSparse Self-Attention which uses a random sparse mechanism, Self-Attention can deterministically model the global dependencies of the sequence. Therefore, Self-Attention provides a stable and reliable feature extraction process, which can effectively learn the context representation of the input sequence and provide better sequence features for the subsequent ProbSparse Self-Attention layer. Second, We optimized the residual connection structure in ProbSparse Self-Attention Layer. We removed the original cascaded convolution operation on the Attention output, and directly added the input and Attention output as the final output of this module. This residual connection avoids potential information loss of the original features through successive convolutions, and effectively retains important features of the input sequence. At the same time, it also reduces the amount of calculation and improves computational efficiency. It can be seen that while keeping the overall framework of the Informer Encoder structure unchanged, we made optimizations in architecture and connections. Full Self-Attention provides a new feature extraction process, and the residual connection improvement of ProbSparse Self-Attention increases the ability to retain information. These module-level optimizations enable the model to learn richer and more effective sequence feature representations, thereby improving the overall modeling effect.

1) Self Attention: Self-Attention is an attention mechanism that solely relies on the sequence itself. In the task of time series feature extraction. Firstly, it defines a feature representation matrix of the time series, where each row represents the feature vector of one time step. Then use this matrix as both the Key matrix and the Query matrix, meaning the time series features serve as both key features and query vectors. The following calculations are: for each Query vector in the matrix, compute its dot product with all Key vectors

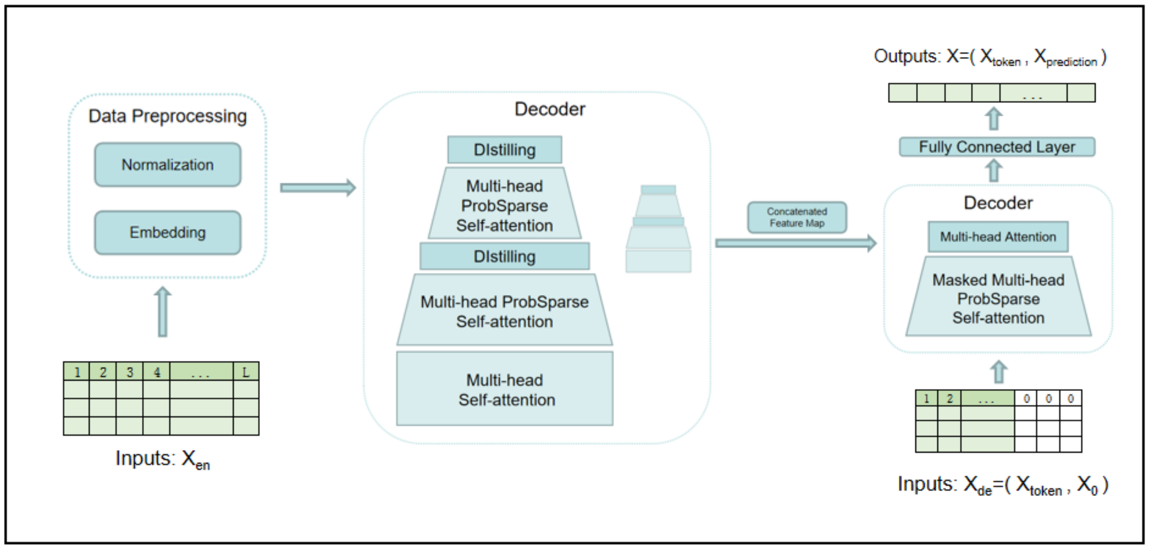
Data Preprocessing
Normalization
Embedding
Inputs: X_en
Decoder
DIstilling
Multi-head ProbSparse Self-attention
DIstilling
Multi-head ProbSparse Self-attention
Multi-head Self-attention
Concatenated Feature Map
Outputs: X=(X_token , X_prediction)
Fully Connected Layer
Decoder
Multi-head Attention
Masked Multi-head ProbSparse Self-attention
Inputs: X_de=(X_token , X_0)

Fig. 6: The structure of NILMformer

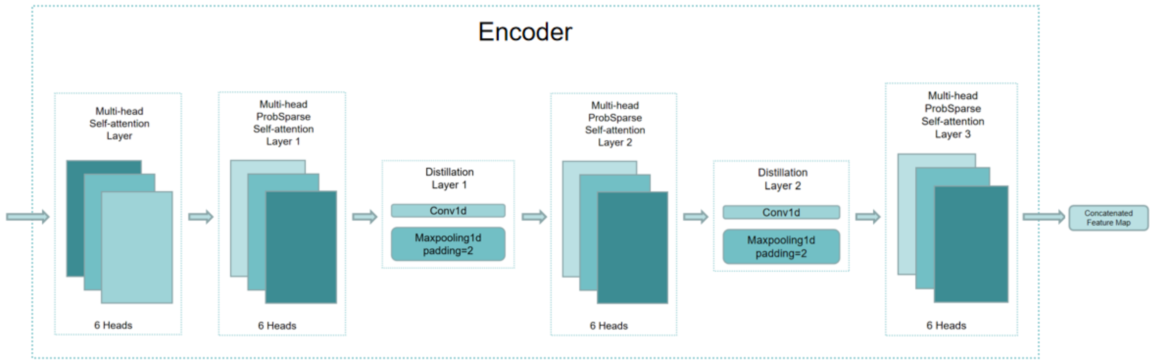
Encoder
Multi-head Self-attention Layer
6 Heads
Multi-head ProbSparse Self-attention Layer 1
6 Heads
Distillation Layer 1
Conv1d
Maxpooling1d padding=2
Multi-head ProbSparse Self-attention Layer 2
6 Heads
Distillation Layer 2
Conv1d
Maxpooling1d padding=2
Multi-head ProbSparse Self-attention Layer 3
6 Heads
Concatenated Feature Map

Fig. 7: Encoder

in the matrix, representing the relevance between the query vector and key vectors. Apply softmax normalization to the dot product results to obtain the Self-Attention weights of the query vector to each Key vector. Finally, multiply each Key vector by its corresponding Self-Attention weight and sum them up to obtain the Self-Attention output that aggregates information from all key vectors, as the new feature expression of the query vector.This can be formulated as:

$$\operatorname{Attention}(Q, K, V)=\operatorname{softmax}\left(\frac{Q K^T}{\sqrt{d_k}}\right) V \tag{10}$$

Through the above process, we can obtain a Self-Attention output sequence with the same length, but each vector has gone through feature extraction via internal self-attentive calculations. Since Self-Attention is solely based on the sequence itself, it can well model the inherent time dependencies in the time series. Thus, we utilize the Self-Attention mechanism to implement feature expression of the time series, which can be fed as the new features of the time series to subsequent models, helping them better utilize the structural information of the sequence.

2) ProbSparse Self-Attention: In the ProbSparse Self Attention layer, we no longer need to compute attention scores one by one. Instead, we efficiently extract attention features through sampled computation. We reformulate the attention computation as:

$$\mathcal{A}\left(q_i, K, V\right)=\sum_j \frac{f\left(q_i, k_j\right)}{\sum_l f\left(q_i, k_l\right)} v_j=\mathbb{E}_{p\left(k_j \mid q_i\right)}\left[v_j\right] \tag{11}$$

Where

$$p\left(k_j \mid q_i\right)=\frac{f\left(q_i, k_j\right)}{\sum_l f\left(q_i, k_l\right)} \tag{12}$$

$$f\left(q_i, k_j\right)=\exp \left(\frac{q_i k_j^T}{\sqrt{d}}\right) \tag{13}$$

The i-th row of matrix Q,K and V is represented as q_i, k_i and v_i. The core idea of ProbSparse Self Attention is to identify those important and sparse queries, and only compute attention values for those queries, in order to optimize computational efficiency. Then, according to the sparsity of the query: the KL divergence between Q and $p\left(k_j \mid q_i\right)$, they indicate that we need to find these sparse but important part of queries. Here are the detailed steps for ProbSpare Self Attention computation:

$$KL(q\|p) = \ln \sum_{l=1}^{L} e^{q_i k_l^T / \sqrt{d}} - \frac{1}{L} \sum_{j=1}^{L} q_i k_j^T / \sqrt{d} - \ln L \quad (14)$$

$$M\left(q_i, K\right) = \ln \sum_{l=1}^{L} e^{q_i k_l^T / \sqrt{d}} - \frac{1}{L} \sum_{j=1}^{L} q_i k_j^T / \sqrt{d} \quad (15)$$

(1) For each query, randomly choose $N = 5 * lnL$ keys, where L is the sequence length of the input to the Attention module;

(2) Compute the sparsity score $M(q_i, k)$ for each query;

(3) Select the top N queries with the highest sparsity scores;

(4) Only compute the dot product results between the chosen N queries and keys, thereby obtaining the attention outputs;

(5) For the left $L - N$ queries, skip the computation step and simply take the mean of the Self-Attention layer's input as the output. This ensures that the sequence length of the input and output is L for each ProbSparse Self Attention layer.

The overall time complexity is $O(LInL)$.

3) Residual Networks Improvement: We optimized the residual network used in the ProbSparse Self Attention layer. The original residual network is shown in figure. It computes output Y from input X via ProbSparse Self Attention, sums X and Y as Z_1 and Z_2 , then applies two consecutive 1D convolutions on Z_1 to get Z_1'', and finally adds Z_1'' and Z_2 as the output. We found this design may be problematic: the two successive convolutions could damage the information in the original input X. Since X and Y have a residual connection, we found that directly adding X and Y as the output is more reasonable to maximize information retention from X.Based on the analysis, we optimized the residual network by removing the two convolutions on Z. The improved structure directly adds X and Y as the output of the residual module. This avoids potential information loss of X through successive convolutions and thus enhances the model's representational power.

We found this improvement in the original residual network and modified it to a more streamlined and effective form while keeping the ProbSparse Self Attention layer useful. This structural adjustment improves the model's representational and information retention capabilities.

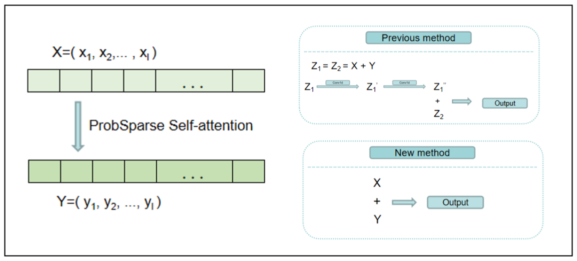

Fig. 8: Residual Networks

4) Distilling: The ProbSparse Self-Attention mechanism leads to superfluous combinations of value V in the encoder's feature map. To tackle this issue, we use a distillation process to prioritize superior values with fewer features. This forms a more concentrated Self-Attention feature map for the subsequent layer. We reduce the dimension of the input by introducing a one-dimensional convolutional layer that halves the length of the input.The distillation operation plays a similar role in our model - extracting the most prominent features and pruning extraneous ones from the ProbSparse Self-Attention. This streamlines the feature map fed to later layers, focusing model capacity on the most critical information.

Here is the formula to achieve distilling:

$$X_{j+1}^t = \operatorname{MaxPool}\left(\operatorname{ELU}\left(\operatorname{Conv} 1d\left(\left[X_j^t\right]_{AB}\right)\right)\right) \quad (16)$$

$$\operatorname{ELU}(x) = \begin{cases} e^x - 1, x < 0 \\ x, x \geq 0 \end{cases} \quad (17)$$

The output of the multi-headed ProbSparse Self-Attention layer in the current layer is represented by X_{j+1}^t. This output is calculated based on $\left[X_j^t\right]_{AB}$, which refers to the computation result from the multi-headed ProbSparse Self-Attention layer in the preceding layer, and the ELU activation function utilized.

C. Decoder

A standard decoder structure is used in our model. It is comprised of two multi-head Self-Attention layers, with the first layer applying ProbSparse Self-Attention and the second layer applying regular attention. The decoder's input consists of the encoder outputs as well as the embedded projected input sequences. The input sequences are separated into two parts:

$$X_{\text{fdec}} = \operatorname{concat}\left(X_{\text{token}}, X_{\text{pred}}\right) \in R^{\left(L_{\text{token}} + L_y\right) \times d_{\text{model}}} \quad (18)$$

X_{fdec} refers to the input sequence fed into the decoder.X_{token} refers to the start flag and X_{pred} refers to the target placeholder.

Zeros are padded to the timestamps to maintain consistent dimensionality when inputting the predicted sequence. The masked multi-head Self-Attention mechanism applies Self-Attention in a masked approach, only allowing each position to attend to current information to avoid self-regression.

VI. Evaluation

A. Dataset and Training Methods

We use the REDD dataset to verify the performance of our proposed methods. The REDD dataset is an influential open-source dataset in the field of electrical load monitoring. It compiled approximately one year of residential electricity usage data from 2011 April to December across 6 American households, encompassing aggregate household as well as individual appliance-level information for over 100 devices. Power consumption was sampled at high frequencies ranging from 1 to 15 minutes, stored in CSV format totaling around 4GB in size. These high-quality electricity data with rich appliance usage patterns can be utilized to train various load forecasting and disaggregation algorithms. Researchers have extensively leveraged the REDD dataset to develop household energy management systems, grid optimization techniques, and other power research. It is fair to say that REDD has catalyzed advancements in the load monitoring domain and serves as an important public benchmark dataset.

For L-TCN, we opted to utilize the low-frequency data from household 1, which encompasses a diverse array of electrical appliances, to conduct our assessments. The training dataset comprising primarily of refrigerator and microwave data was derived from April 20, 2011 to April 30, 2011. Subsequently, we leveraged the features extracted by L-TCN to train the NILMformer model. The test dataset spanning May 1, 2011 to May 3, 2011 was then utilized for evaluation.

B. Performance Evaluation of the L-TCN

In this section, we delve into a comprehensive performance analysis of the enhanced NILM attack method based on TCN that we propose. Figure 9is the classic TCN model, and Figure 10 is our modified L-TCN model For a holistic understanding, we juxtapose our method against three state-of-the-art NILM techniques: GRU, LSTM, and Seq2point.

1) Comparison of MAE and RMSE: The crux of our improved TCN-based NILM method is to decompose the load sequence of the target appliance based on the temporal series of the user's total demand. To this end, we strategically select two quintessential appliances from Household 1: the refrigerator, a periodic low-load appliance, and the microwave, a low-frequency high-load appliance, as our attack subjects. Our evaluation metrics of choice are the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE), both of which offer insights into the decomposition accuracy.

As illustrated in Table I, a comparative analysis between our NILM method and its contemporaneous counterparts reveals that our approach consistently outperforms in both MAE and RMSE metrics. Specifically, for the refrigerator, our method manages to reduce the MAE by nearly 10% and the RMSE by an impressive 40%. When it comes to the microwave, the reductions are even more pronounced, with MAE dropping by approximately 50% and RMSE by close to 30%. It is evident that our method exhibits superior load decomposition capabilities for low-frequency appliances compared to periodic ones. This can be attributed to our method's adeptness at identifying short-duration, high-power load characteristics. Furthermore, it's worth noting that these low-frequency appliances are more likely to harbor user-sensitive information, such as their travel patterns.

To validate the efficacy of our enhanced NILM-former model, we opted for the REDD dataset and juxtaposed it against GRU [24],CNN [25], and LSTM [26] on low-frequency data. The empirical outcomes unambiguously indicate that, in comparison to these state-of-the-art methodologies, our NILM-former model boasts superior performance and accuracy, underscoring the significance and novelty of our research. In Figure 11, we present a comparative analysis between our enhanced approach and other prevalent methods in the NILM domain. The results clearly demonstrate the superior separation efficacy of our method.

C. Comparison experiment: Prediction using univariate features

In the task of household electricity load forecasting, we first constructed several univariate prediction models, using the time series of total electricity usage for forecasting. The specific experimental methods are as follows.

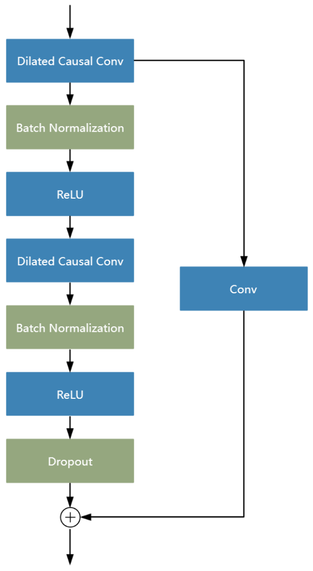

Fig. 9: Original Model

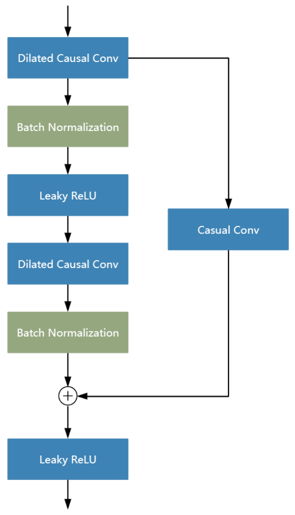

Fig. 10: Refined Mode

Multilayer Perceptron (MLP) model. The Multilayer Perceptron (MLP) model utilizes a multilayer fully connected network to fit the sequence. This kind of network can approximate complex nonlinear relationships, but its architecture determines that it can only model the relationships between each time step, and is relatively weak at modeling long-term temporal dependencies of the sequence.

One-dimensional Convolutional Neural Network (1D CNN). The One-dimensional Convolutional Neural Network (1D CNN) performs local feature extraction on the sequence by sliding one-dimensional convolution kernels. The increased nature of the convolutional layer enables it to automatically learn local features of the sequence, but its limited receptive field makes it difficult to model long-distance dependencies. 1D CNN's modeling of temporal correlations is also limited.

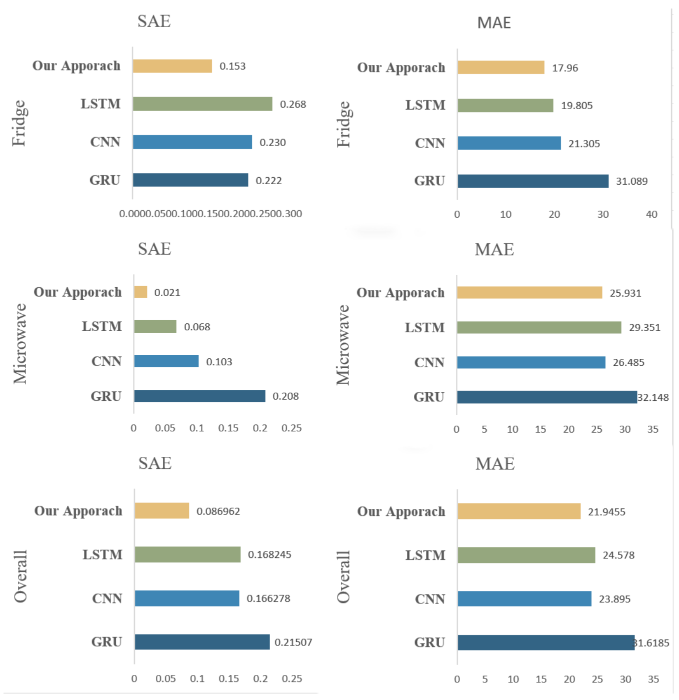

Fig. 11: Comparison of Different NILM Methods

Long Short-Term Memory Network (LSTM). The Long Short-Term Memory Network (LSTM) can better capture the long-term dependency relationships in time series through the memory unit mechanism. But the standard LSTM structure only contains a single hidden state, and has limited ability to integrate and express multiple variables. Even stacking multiple LSTM layers cannot effectively utilize the information of multiple input variables.

Tabel II shows the performance of univariate prediction models on MAE and MSE.

TABLE II: The Performance of Univariate Prediction Models on MAE, MSE and RMSE

Model	MAE	MSE	RMSE
LSTM	0.288525	0.117871	0.343322
CNN	0.186752	0.137859	0.137859
MLP	0.342051	0.432496	0.657644
Transformer	0.240748	0.265346	0.515117

Figure 12 shows the predicted curves generated by univariate prediction models.

Through the prediction results shown in Figure II above, we found that the prediction curve of the single-variable prediction model has a large deviation from the actual electricity consumption curve, and the fitting degree between them is extremely poor.

Specifically, in the relatively flat sections of the actual electricity consumption, the prediction curve fails to successfully capture the weak changing trend of the data, with significant overestimation or underestimation. When the actual electricity consumption shows more obvious peaks or troughs, the prediction curve also fails to reflect such fluctuations and cannot show the upward or downward trend of electricity consumption. The prediction curve maintains at a relatively gentle level, without reflecting the characteristic morphology of peaks and troughs. Such prediction bias exists throughout the curve, with systematic deviations between the prediction and the actual. This indicates that the contrastive models have failed, and cannot establish a mapping relationship between the changing pattern of actual electricity consumption and the predicted values. Its prediction results do not have practical reference and guidance significance.

Through the above qualitative analysis of the prediction curve, we can preliminarily judge that there are difficulties in household electricity prediction relying solely on a single total electricity consumption sequence. The models can hardly extract effective features and establish reliable predictions, and cannot effectively model situations like electricity peaks and troughs. This provides evidence for the necessity of the multi-variable prediction framework proposed in this paper.

D. *Performance of NILMformer*

After finding that prediction using univariate features performs poorly, we chose to use NILMformer for prediction. By changing various hyperparameters of NILMformer, we have identified the optimal training scheme. Then by sequentially adjusting the input and output lengths and using Loss, MSE and MAE as evaluation metrics, we have reached the following conclusions. Results are shown in Figure 13.

Increasing the input sequence length can provide the model with richer historical data and contextual information, which helps the model to more accurately capture the long-term dependencies of time series, thereby improving the overall predictive performance of the model. When the output length is fixed, as the input length increases from 32 to 56, the Loss value decreases from 0.03261 to 0.01045. However, the increase of input length is not unlimited. When it increases to a point, the improvement of model prediction effect will tend to be flat and reach a bottleneck. Therefore, when setting the input length, we need to consider the balance between obtaining more information and computational cost.

Increasing the output sequence length will lead to an accumulation of errors when the model predicts multiple future time steps, which will reduce the model's accuracy for long-term prediction. When the input length is fixed at 56, as the output length increases from 12 to 56, the MAE value rises from 0.15373 to 0.33429. Therefore, the output sequence length should not be too long, and needs to be reasonably shortened to reduce the accumulation of prediction errors.

From the results, when the input length is 56, the Loss value of the model is the lowest overall, especially when the output length is 12, the Loss is only 0.01045, which is significantly better than the Loss values when Input Length is 32 and 48 (0.03261 and 0.05654 respectively). Meanwhile, when the output sequence length is 12 and 24, the input sequence length of 32 leads to a relatively lower mean absolute error (MAE) value (specifically 0.12789 and 0.13579), compared to the

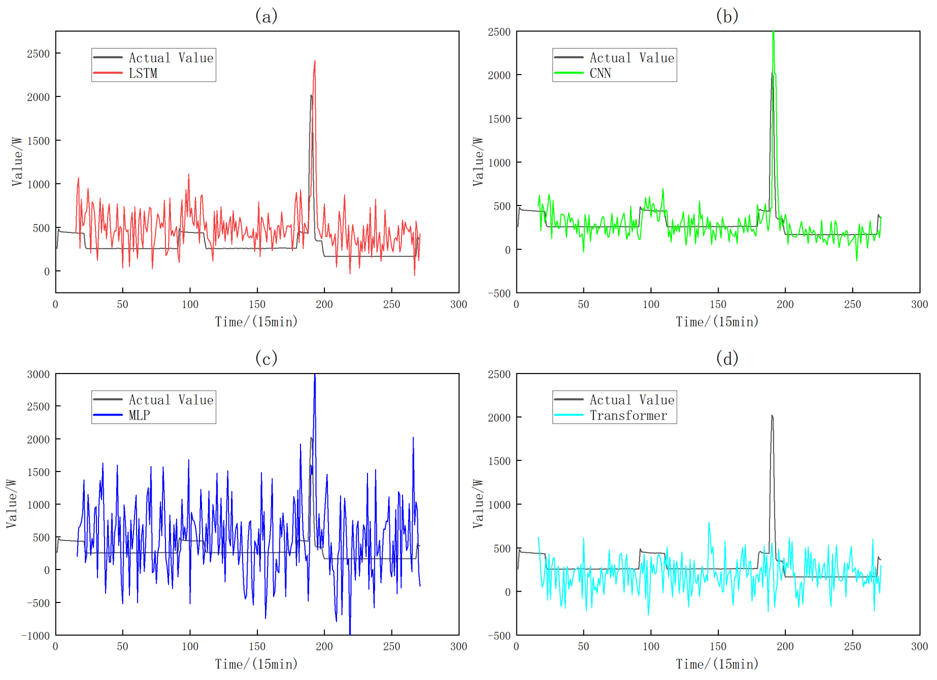

Fig. 12: Results of Different Single-variable Prediction Models

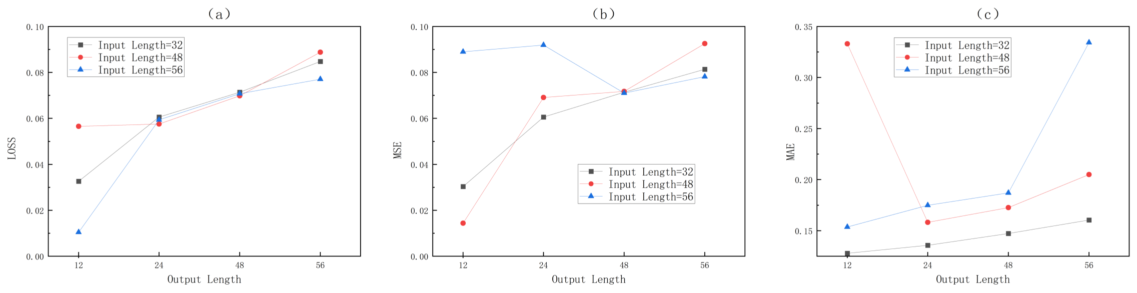

Fig. 13: Comparison of Different Input and Output Lengths on NILMformer

MAE when Input Length is 48 and 56. As for the mean squared error (MSE) metric, the differences between the three input sequence length settings are small. Taken together, the input length of 56 and output length of 12 can make the model achieve lower Joint Loss (0.01045) and MAE (0.15373). These results indicate that the longer input sequence length can provide the model with richer contextual information to capture the long-term dependencies in the time series, while the shorter output length can reduce errors when the model predicts future values.

1) Comparison experiment: Prediction using extracted effective features: To evaluate the effectiveness of the proposed NILMformer model, comparative experiments were conducted between NILMformer and baseline methods including CNN, LSTM, and Transformer for multivariate time series prediction. The experimental results are shown in Figure 14 15 16 and Table III.

TABLE III: The Performance of Mutivariate Prediction Models on MAE, MSE and RMSE

Model	MAE	MSE	RMSE
LSTM	0.168303	0.233686	0.483411
CNN	0.193421	0.239647	0.489537
Transformer	0.145275	0.171846	0.414542
NILMformer	0.153731	0.089022	0.234186

Through comparative analysis, the proposed NILMformer model demonstrates superior predictive capabilities over con-

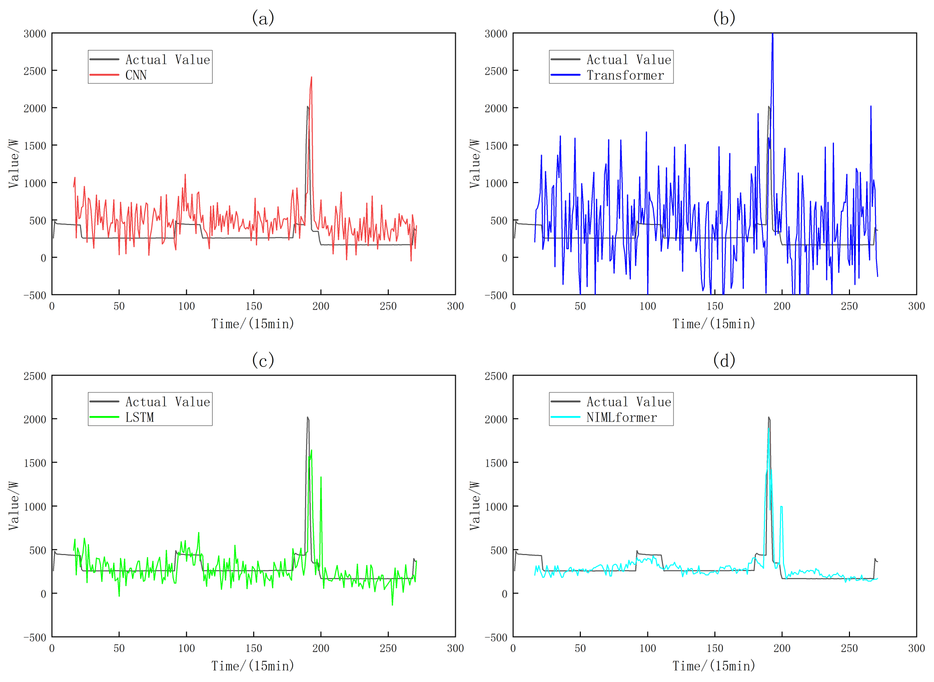

Fig. 14: Results of Different Multivariate Prediction Models

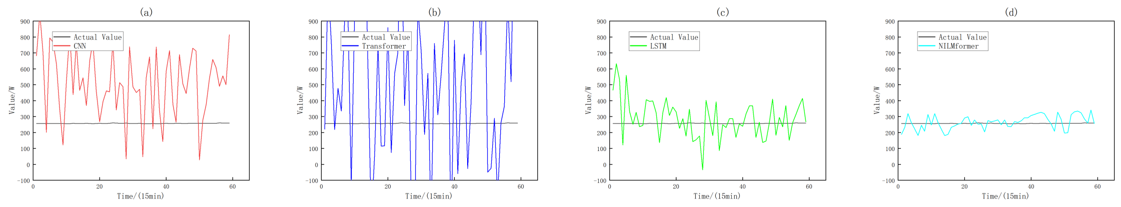

Fig. 15: Prediction Values of Different Models during Relatively Stable Periods

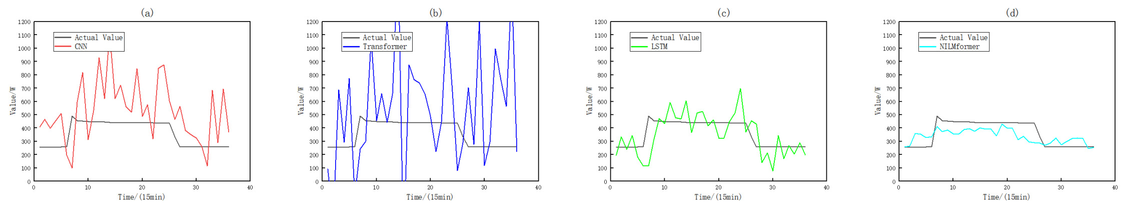

Fig. 16: Prediction Values of Different Models when Peaks Appear

ventional baselines. Specifically:

First, NILMformer achieved the lowest RMSE of 0.234186, MSE of 0.089022, and MAE of 0.153731 on the test set compared to the other models. This indicates its superior overall prediction accuracy.

Second, NILMformer exhibited strong capabilities in accurately capturing subtle changes and fluctuations in time series. In relatively flat regions, it avoided significant over- or under-estimation of the target value. This outperformed LSTM, CNN, and Transformer which had larger errors.

Third, When fluctuations like peaks and troughs occurred, NILMformer also showed excellent modeling of temporal dependencies. It could predict the timing of peaks and troughs relatively precisely. The specific predicted values also provided reasonable estimates of the overall increasing or decreasing trend. NILMformer demonstrated robust fitting performance across all stages of the prediction sequence. Its predicted curve adequately mirrored the morphological characteristics of the ground truth curve, superior to the other methods.

In summary, NILMformer achieves superior empirical performance over other models. The results validate its effectiveness as an accurate predictive model for complex multivariate data.

VII. Conclusion

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China under Grants 62203350 and 61973247, in part by China Postdoctoral Science Foundation 2021M692566, in part by the Key Program of the National Natural Science Foundation of China under Grant 61833015.

References

[1] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, 1992.

[2] M. Zeifman and K. Roth, "Nonintrusive appliance load monitoring: Review and outlook," *IEEE Transactions on Consumer Electronics*, 2011.

[3] J. Kolter, S. Batra, and A. Ng, "Energy disaggregation via discriminative sparse coding," *Advances in Neural Information Processing Systems*, 2010.

[4] J. Kelly and W. Knottenbelt, "Neural nilm: Deep neural networks applied to energy disaggregation," *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, 2015.

[5] L. Mauch and B. Yang, "A new approach for supervised power disaggregation by using a deep recurrent lstm network," *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016.

[6] J. Lin and W. Chen, "A cloud-based platform for appliance transfer learning in non-intrusive load monitoring," *Applied Energy*, 2018.

[7] X. Zhang and Y. Wang, "Energy disaggregation with graph neural networks," *IEEE Transactions on Smart Grid*, 2019.

[8] Y. Chen and M. Jahromi, "Non-intrusive load monitoring using adversarial deep learning," *Applied Energy*, 2021.

[9] H. Hippert, C. Pedreira, and R. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.

[10] N. Amjady and F. Keynia, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," *IEEE Power Engineering Review*, vol. 21, no. 8, pp. 62–62, 2001.

[11] D. L. Marino, K. Amarasinghe, and M. Manic, "Building thermal load prediction by using lstm neural networks," *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pp. 1411–1416, 2016.

[12] W. Kong, Z. 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.

[13] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.

[14] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, "Transformer meets tracker: Exploiting temporal coherence for robust visual tracking," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9801–9810, 2021.

[15] S. Touzani, J. Granderson, and S. Fernandes, "Gradient boosting machine for modeling the energy consumption of commercial buildings," *Energy and Buildings*, vol. 158, pp. 1533–1543, 2018.

[16] J. Wang, R. Chen, G. Chen, J. Wang, and J. Chen, "Multivariate lstm-fcns for time series prediction," *arXiv preprint arXiv:1809.07408*, 2018.

[17] T. Ahmad, H. Chen, Y. Guo, and J. Wang, "A review on applications of ann and svm for building electrical energy consumption forecasting," *Renewable and Sustainable Energy Reviews*, vol. 75, pp. 102–109, 2017.

[18] H. Yang, M. Santamouris, and S. H. Lee, "Multivariate lstm-fcn for automated design of multi-story building energy consumption emulator," *Energy*, vol. 165, pp. 72–84, 2018.

[19] J. Zhao, W. Chen, Z. Yu, P. Chen, and X. He, "A gated graph sequence neural network with attention mechanism for short-term load forecasting," *Energy and AI*, vol. 1, p. 100007, 2020.

[20] C. Song, Y. Shen, Y. Liu, Q. Yang, and X. C. Li, "Graph convolutional lstm neural network for multivariate time series forecasting," *Neural Computing and Applications*, pp. 1–17, 2020.

[21] T. Hong, X. Luo, A. Dalla Rosa, and M. Radhakrishnan, "Citywide building energy modeling and benchmarking with air pollution constraints," *Applied Energy*, vol. 256, p. 113947, 2019.

[22] R. Zhang, K. H. Lam, and Y. Huang, "Urban building energy modeling with urban spatial data," *Energy and Buildings*, vol. 234, p. 110755, 2021.

[23] D. Li, Q. Yang, F. Zhang, Y. Wang, Y. Qian, and D. An, "Research on privacy issues in smart metering system: An improved tcn-based nilm attack method and practical drl-based rechargeable battery assisted privacy preserving method," *IEEE Transactions on Automation Science and Engineering*, 2023.

[24] A. Gru, B. Haverkos, A. Freud, J. Hastings, N. Nowacki, C. Barrionuevo, C. Vigil, R. Rochford, Y. Natkunam, R. Baiocchi *et al.*, "The epstein-barr virus (ebv) in t cell and nk cell lymphomas: time for a reassessment," *Current hematologic malignancy reports*, vol. 10, pp. 456–467, 2015.

[25] L. O. Chua and T. Roska, "The cnn paradigm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 147–156, 1993.

[26] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.