Nonintrusive Load Monitoring Based on Self-Supervised Learning

Shuyi Chen, Bochao Zhao, Member, IEEE, Mingjun Zhong, Member, IEEE, Wenpeng Luan, Senior Member, IEEE, and Yixin Yu, Life Senior Member, IEEE

Abstract—Deep learning models for nonintrusive load monitoring (NILM) tend to require a large amount of labeled data for training. However, it is difficult to generalize the trained models to unseen sites due to different load characteristics and operating patterns of appliances between datasets. For addressing such problems, self-supervised learning (SSL) is proposed in this article, where labeled appliance-level data from the target dataset or house are not required. Initially, only the aggregate power readings from target dataset are required to pretrain a general network via a self-supervised pretext task to map aggregate power sequences to derived representatives. Then, supervised downstream tasks are carried out for each appliance category to fine-tune the pretrained network, where the features learned in the pretext task are transferred. Utilizing labeled source datasets enables the downstream tasks to learn how each load is disaggregated, by mapping the aggregate to labels. Finally, the fine-tuned network is applied to load disaggregation for the target sites. For validation, multiple experimental cases are designed based on three publicly accessible REDD, U.K.-DALE, and REFIT datasets. Besides, the state-of-the-art neural networks designed based on three publicly accessible REDD, U.K.-DALE, and REFIT datasets. For addressing, multiple experimental cases are designed based on three publicly accessible REDD, U.K.-DALE, and REFIT datasets. Thus, SSL generally outperforms zero-shot learning in improving load disaggregation performance without any submetering data from the target datasets.

Index Terms—Deep neural network (DNN), nonintrusive load monitoring (NILM), self-supervised learning (SSL), sequence-to-point learning.

I. INTRODUCTION

In recent years, energy shortage and environmental pollution worldwide have become increasingly serious. Therefore, the approaches of efficient energy utilization and carbon emissions reduction are being explored [1], [2]. Meanwhile, with the global deployment of smart meters, benign interaction between power suppliers and users has been established for enhancing demand side management and optimizing power grid operation [3]. As one of the energy conservation applications, electricity consumption detail monitoring has attracted extensive attention around the world [4]. In general, load monitoring technology is mainly categorized into intrusive way and nonintrusive way. Note that intrusive load monitoring requires extra sensor installation for submetering. Alternatively, the concept of nonintrusive load monitoring (NILM) was proposed by Hart [5] as identifying power consumed by each individual appliance via analyzing aggregate power readings using only software tools. NILM offers appliance-level power consumption feedback to both demand and supply sides economically and efficiently, contributing to power system planning and operation [1], energy bill savings [6], demand side management [7], and energy conservation and emission reduction [3], [6], [8].

NILM is a single-channel blind source separation problem, aiming to disaggregate the appliance-level energy consumption from the aggregate measurements [9]. Combinatorial optimization (CO) is initially applied to perform NILM in [5], searching for the best combination of operational states of individual appliances at each time instance. However, CO relies on the power range of each operational state as prior knowledge, making it unavailable to the newly added appliances [10]. Benefiting from the technology development in recent years on big data, artificial intelligence, and edge computing, plenty of NILM approaches have been proposed based on machine learning, mathematics, and signal processing [8], [11]. Factorial hidden Markov model (FHMM) and its variants [12], [13], [14] are popular in carrying out NILM. Given an aggregate power signal as the observation, such FHMM-based NILM methods estimate the hidden operational states of each appliance considering their state continuity in time series [15], [16]. Thus, FHMM-based methods usually achieve good results in disaggregating loads with periodic operation such as refrigerators. However, their performance is limited for the loads with short-lasting working cycles and the ones with less frequent usage. Note that FHMM-based methods are regarded as state-based NILM approaches, where the aggregate power measurement at each time instance is assigned to each operational state per appliance [17]. Alternatively, NILM approaches can be event-based, where sudden changes in power signals referring to turn-on, turn-off, and state transition events are featured [17]. Such event-based NILM methods can be carried out via subtractive clustering and the maximum likelihood classifier [18]. Besides, graph signal processing concepts are applied to perform NILM, mapping correlation among samples to the underlying graph structure [19], [20].

Manuscript received 23 October 2022; accepted 27 January 2023. Date of publication 22 February 2023; date of current version 2 March 2023. This work was supported in part by the Joint Funds of the National Natural Science Foundation of China under Grant U2066207 and in part by the National Key Research and Development Program of China under Grant 2020YFB0905904. The Associate Editor coordinating the review process was Dr. Guglielmo Frigo. (Corresponding author: Wenpeng Luan.)

Shuyi Chen, Bochao Zhao, Wenpeng Luan, and Yixin Yu are with the School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: wenpeng.luan@tju.edu.cn).
Mingjun Zhong is with the Department of Computing Science, University of Aberdeen, AB24 3UE Aberdeen, U.K. (e-mail: mingjun.zhong@abdn.ac.uk).
Digital Object Identifier 10.1109/TIM.2023.3246504

1557-9662 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
Although such event-based NILM approaches can achieve high load identification accuracy, they tend to suffer from measurement noises.

Deep neural networks (DNNs), performing well in computer vision, speech recognition, and natural language processing, have been employed in load disaggregation since 2015 [21]. Since then, DNN structures have become more and more popular to perform NILM, including long short-term memory (LSTM) [15], [21], [22], gated recurrent unit (GRU) [10], [23], denoising autoencoder (dAE) [21], [24], and convolutional neural network (CNN) [25], [26], [27], showing competitive performance against traditional NILM methods. Although LSTM is suitable for long time-series-related tasks due to avoiding the vanishing gradient problem, it underperforms in NILM task compared to CNN [28], [29]. Attention mechanism is applied to LSTM in [22] for improving load disaggregation performance. As a variant of LSTM, GRU can also remember data patterns. In addition, GRU contains fewer parameters, thus requires shorter training time, which is suitable for online application in NILM [10], [30]. Note that the bidirectional GRU (Bi-GRU) is employed to perform NILM in [10], where the network can be trained simultaneously in positive and negative time directions. Besides, dAE is applied to NILM by recovering the power signal of target appliances (clean signal) from the aggregate (noisy signal) [8], [21], where CNN layers are usually embedded [21], [24]. A state-of-the-art CNN-based NILM method, S2p, is proposed in [9] and claimed to outperform benchmarks in NILM task [9], [31], benefiting from meaningful latent features learned from sub-metering data. Recently, more improved network structures are applied to NILM, such as generative adversarial networks (GANs) [32] and transformer [33]. GAN framework helps to improve appliance profile estimation when performing NILM, as more outputs close to the ground truth (GT) are generated in network training. In terms of transformer, attention mechanism is advantageous to NILM via capturing long-range dependency from sequential data [30].

Compared to traditional NILM methods, the advantages of DNN-based NILM approaches include automatic feature extraction from power readings and linearity between computational complexity and appliance amount [4]. However, the promising performance of the aforementioned DNN-based methods relies on a large amount of submetering data from the target set for training [4]. Since such data collection may last for months or even years [4], it is neither user-friendly nor economical in practice.

Alternatively, transfer learning concepts are proposed, where transferable networks can be trained on a source (seen) dataset and applied to the load disaggregation task on a target (unseen) dataset [2], [32], [34], [35]. Depending on whether network fine-tuning is required, transfer learning can be classified as few-shot learning (FSL) and zero-shot learning (ZSL) [34]. For fine-tuning in FSL, a small amount of labeled data from the target set are still required [34]. However, when labels are unable to be captured from the target datasets, ZSL offers proper solutions. In [35], ZSL achieves tiny performance drop compared to baseline when it is employed in load disaggregation by both GRU and CNN networks, showing transferability across datasets. However, in ZSL, it is difficult to generalize networks between datasets with different load characteristics and operating patterns of appliances. The same as ZSL, self-supervised learning (SSL) requires no labeled data from the target set. SSL is an efficient way to extract universal features from large-scale unlabeled data, contributing to robustness enhancement [36]; thus, it performs well in image processing and speech recognition [37]. To the best of our knowledge, SSL has not been used to solve NILM problem.

Driven by such research gaps, in this article, SSL is applied to three state-of-the-art NILM algorithms based on CNN, GRU, and LSTM, as S2p [9], Bi-GRU [10], and LSTM [22]. For performing NILM, a self-supervised pretext task training is initially carried out for learning features from the aggregate power readings from the unlabeled data in the target set. Then, the pretrained network is fine-tuned in the supervised downstream task training based on the labeled data from the source set for transferring the prelearned knowledge to load disaggregation. After pretraining and fine-tuning, the network can be applied to load disaggregation for target sites. The proposed method is validated on the real-world datasets at 1-min granularity, in the scenarios designed for the same dataset or across various datasets. The contributions of this article are clarified as follows.

1) SSL is applied to load disaggregation based on deep learning without submetering on the target set, by setting a pretext task for network pretraining on unlabeled data from the target set with fine-tuning.

2) Experiments are carried out for all combinations of three state-of-the-art DNN-based NILM methods (S2p [9], Bi-GRU [10], and LSTM [22]) and learning frameworks (SSL and ZSL), on three real-world datasets.

3) Six cases differing in data selection are designed for performance evaluation on the data across houses or sets, showing SSL generally outperforms in various metrics and energy consumption estimation results, with comparable training time cost.

The rest of this article is organized as follows. In Section II, the NILM formulation is clarified, followed by introducing the preliminaries for NILM neural networks and SSL. The methodology of SSL for NILM is explained in Section III. Section IV contains datasets, evaluation metrics, and experimental settings, followed by experimental results with discussion illustrated in Section V; eventually, the conclusion is drawn and the future work is prospected in Section VI.

II. Preliminaries

In this section, we first formulate the NILM problem and then clarify seq2seq and seq2point concepts, followed by an introduction for two seq2point network architectures. Finally, the overall structure of SSL is demonstrated.

A. NILM Problem Formulation

Assuming that the aggregate power reading measured in a household at time index \( t \in [1, T] \) is \( y_t \), where \( T \) refers to the total number of samples. Then, the simultaneous power
1) S2p: The utilization of S2p in NILM is based on the assumption that the midpoint of each sliding window acts as its nonlinear regression representation. Namely, S2p makes full use of the past and future information to infer the midpoint, as shown in Table I.

For a defined neural network $f^m$, the input is a power sequence denoted by $y_{t:t+W-1}$ segmented by a sliding window from the aggregate, where $t$ is the time index, and the window size $W$ is set to an odd number. Thus, by mapping each sequence $y_{t:t+W-1}$ to the power $x^m_t$ consumed by appliance $m$ at $\tau = t + (W - 1)/2$, the entire power signal $x^m$ for appliance $m$ can be predicted. Such model can be formulated as

$$x^m_t = f^m(y_{t:t+W-1}) + \varepsilon$$

where $\varepsilon$ is the $W$-dimensional Gaussian random noise. Besides, the loss function in the network training is formulated as follows:

$$L_p = \sum_{i=1}^{T-W+1} \log p(x^m_\tau|y_{t:t+W-1}, \theta_p)$$

where $\theta_p$ is a set of network parameters.

The CNN-based architecture of S2p is illustrated in Table I containing five convolutional layers and one dense layer. In each iteration, the input signal refers to an $n$-length sliding window for aggregate measurements. Then, five convolutional layers are employed for feature extraction through an activation function called ReLU. Eventually, the feature maps are flattened and fed to a dense layer, and an appliance-level classification is performed. This approach is demonstrated in Table II.

### B. Sequence-to-Sequence Versus Sequence-to-Point NILM Frameworks

NILM can be carried out via neural networks with a seq2seq or seq2point framework [25]. In a seq2seq NILM solution, for each appliance, a network learns the nonlinear regression between sequences with the same time stamps, referring to the aggregate and appliance-level power. For an arbitrary aggregate power sequence $y$ covering time instance $t$, the power $x^m_t$ consumed by appliance $m$ at $t$ is predicted by the network, thus can be finalized as the average value of all such predictions [9]. Unlike the seq2seq framework, the seq2point framework predicts the appliance-level power consumed at only one point of each sliding window iteratively. The inputs and outputs of both seq2seq and seq2point frameworks in a NILM task are illustrated in Fig. I.

Note that seq2point frameworks are demonstrated in Fig. I(b) and (c) on two architectures, as S2p proposed in [9] and Bi-GRU proposed in [10], respectively. Compared to the seq2seq framework, the seq2point framework emphasizes the representational power at one element and eases the prediction task. Then, S2p and Bi-GRU are introduced in detail.

### TABLE I

| Network specifications |
|------------------------|
| **Architectures**      |
|                        |
| Input                  |
|                        |
| Filter size: 10        |
| # filters: 30          |
| Stride: 1              |
| Activation: ReLU       |
|                        |
| Conv. layer1           |
| Filter size: 8         |
| # filters: 30          |
| Stride: 1              |
| Activation: ReLU       |
|                        |
| Conv. layer2           |
| Filter size: 6         |
| # filters: 40          |
| Stride: 1              |
| Activation: ReLU       |
|                        |
| Conv. layer3           |
| Filter size: 5         |
| # filters: 50          |
| Stride: 1              |
| Activation: ReLU       |
|                        |
| Conv. layer4           |
| Filter size: 5         |
| # filters: 50          |
| Stride: 1              |
| Activation: ReLU       |
|                        |
| Conv. layer5           |
| Flatten                |
|                        |
| Dense layer            |
|                        |
| Output                 |
| Units: 1024            |
| Activation: ReLU       |
|                        |
| Units: 1               |
| Activation: Linear     |

1. For a defined neural network $f^m$, the input is a power sequence denoted by $y_{t:t+W-1}$ segmented by a sliding window from the aggregate, where $t$ is the time index, and the window size $W$ is set to an odd number. Thus, by mapping each sequence $y_{t:t+W-1}$ to the power $x^m_t$ consumed by appliance $m$ at $\tau = t + (W - 1)/2$, the entire power signal $x^m$ for appliance $m$ can be predicted. Such model can be formulated as

$$x^m_t = f^m(y_{t:t+W-1}) + \varepsilon$$

2. For a defined neural network $f^m$, the input is a power sequence denoted by $y_{t:t+W-1}$ segmented by a sliding window from the aggregate, where $t$ is the time index, and the window size $W$ is set to an odd number. Thus, by mapping each sequence $y_{t:t+W-1}$ to the power $x^m_t$ consumed by appliance $m$ at $\tau = t + (W - 1)/2$, the entire power signal $x^m$ for appliance $m$ can be predicted. Such model can be formulated as

$$x^m_t = f^m(y_{t:t+W-1}) + \varepsilon$$

where $\varepsilon$ is the $W$-dimensional Gaussian random noise.
TABLE II
NETW ORK SPECIFICATIONS FOR Bi-GRU [10]

| Architectures | Network specifications |
|---------------|------------------------|
| Input         | N-length sequence      |
| Conv. layer   |                        |
|              | Filter size: 4         |
|              | # filters: 16          |
|              | Stride: 1              |
|              | Activation: ReLU       |
| Bi-GRU layer1 |                        |
|              | Size: 64               |
|              | Merge: concat          |
|              | Activation: ReLU       |
| Dropout      |                        |
| Bi-GRU layer2 |                        |
|              | Size: 128              |
|              | Merge: concat          |
|              | Activation: ReLU       |
| Dropout      |                        |
| Dense layer  | Units: 128             |
|              | Activation: ReLU       |
| Dropout      |                        |
| Output       | Units: 1               |
|              | Activation: Linear     |

Fig. 2. SSL framework.

As shown in Table II, after each aggregate power sequence is input to the network, a convolutional layer is used for feature extraction. Then, two Bi-GRU layers are applied to enhance the memory for the data patterns based on the extracted features, followed by a dense layer as in the S2p network. Note that dropout performs overfitting prevention for such layers.

Note that convolutional layers are used in both seq2point architectures, with all layers activated by ReLU functions except the output. As the network layers go deeper, larger number of filters in convolutional layers in S2p and more units in Bi-GRU layers in Bi-GRU help to enrich features.

C. Self-Supervised Learning

SSL performs promisingly in computer vision applications without labeled data [36], where visual features are learned from large-scale unlabeled data to avoid the cost of data annotations. In SSL, a pretext task should be predesigned for the network with a loss function to learn features from unlabeled dataset followed by supervised downstream tasks for enriching features. The general pipeline of SSL is shown in Fig. 2.

In Fig. 2, by solving the self-supervised pretext task, the network is pretrained on unlabeled dataset for mapping the relationship between the input and output in the pretext task. Then, benefiting from the downstream task, the predefined network parameters are fine-tuned based on the labeled dataset. Thus, knowledge is transferred between tasks in SSL procedures.

III. SELF-SUPERVISED LEARNING FOR NILM

Motivated by the drawbacks of existing DNN-based NILM methods requiring labeling for the target dataset, in this article, SSL is applied to various neural networks for load disaggregation. The scheme of SSL application in NILM is shown in Fig. 3, taking the S2p NILM method as an example. It can be observed that the proposed load disaggregation approach is composed of three stages: self-supervised pretext task training, supervised downstream task training, and load disaggregation as testing. For given aggregate power measurements, the S2p network can be constructed for mapping each W-length sliding window as a sequence \( y_{r:t+W-1}^{pre} \) to its midpoint \( y_{r:T-1}^{pre} \) through pretraining. Note that self-supervised pretraining is treated as initialization for the general neural network \( f \) for each appliance category, contributing to performance improvement and overfitting mitigation [37]. Then, for each appliance \( m \), a load disaggregation network \( f^m \) can be built by fine-tuning the pretrained network \( f \). That is, the network parameters are fine-tuned by using the aggregate power sequence \( y_{r:t+W-1}^{pre} \) as input and \( x^m_r \) as target for the neural networks, which results in \( f^m \).

Finally, the predicted power reading \( \hat{x}^m \) is the output of \( f^m \) using the mains reading \( y_{r:T-1}^{pre} \) as input. In the following part of this section, we clarify both self-supervised pretext task training and supervised downstream task training proposed for load disaggregation, while testing procedure is executed as in [2].

A. Self-Supervised Pretext Task Training

Firstly, self-supervised pretext task training is set for extracting features from the unlabeled aggregate data from the target set. For matching the pretraining task with the network architectures of S2p and Bi-GRU, a nonlinear loss function is defined as

\[
L_{SSL} = \min_{\theta_{SSL}} \frac{1}{I-W+1} \sum_{t=1}^{I-W+1} \text{loss}(y_{r:t+W-1}^{pre}, y_{r:t+W-1}^{pre})
\]

(4)

where \( \theta_{SSL} \) is a set of network parameters, and \( y_{r:t+W-1}^{pre} \) is its midpoint for S2p or the ending point for Bi-GRU. Equation (4) maps each aggregate power sequence referring to a sliding window of the mains to its midpoint or ending point via a nonlinear regression. It can be inferred from [2] and [9] that mapping the aggregate power sequence to its midpoint can help networks to learn meaningful features, such as appliance-level operational power ranges. Therefore, the midpoint elements are believed as crucial signatures in the pretext task for performing energy disaggregation.

It is known that sudden changes within an aggregate power sequence correspond to appliance state transitions and switching ON/OFF events. Thus, the intuition behind setting such pretext task is to enhance general feature representation from the aggregate in the target dataset, mainly referring to power change points, usage duration, and power levels [9]. Therefore, the network parameters are initialized via iterative updating in the self-supervised pretraining stage, with each sequence as input. The schematic of the pretext task proposed for load disaggregation is shown in Fig. 4.
As shown in Fig. 4, aggregate measurement sequences as sliding windows are iteratively imported into the networks, while the network outputs are their midpoints or ending points.

**B. Supervised Downstream Task Training**

For transferring the features extracted from the “unseen” target dataset without labels to load disaggregation task, the network is updated iteratively. That is, the network parameters are further fine-tuned on the labeled data from the source set, as a supervised downstream task training. By carrying out the downstream task, the general feature representation learned via the pretext task contributes to predicting individual appliance power. As described in Section II, such network is fine-tuned with the mains power in each sequence window as input and the appliance-level power estimation at its midpoint or ending point as output. Thus, the downstream task training for a typical appliance \( m \) is illustrated in Fig. 5.

Note that parameter fine-tuning can be carried out on either partial network layers or all layers, defined as partial fine-tuning and full fine-tuning. For performing partial fine-tuning on both S2p and Bi-GRU architectures, only their dense layers and output layers are fine-tuned, while the other layers are frozen. In DNN, general low-level features can be captured in shallow layers, while in deep layers, high-level features related to the task tend to be extracted [37]. Therefore, partial fine-tuning is usually applied to the tasks with knowledge transfer for sharing common features and thus guaranteeing efficiency. However, it is claimed in [38] that fine-tuning all network...
parameters after self-supervised pretext outperforms partial fine-tuning in some tasks. As fine-tuning parameters in only dense layers may be insufficient in feature extraction and transference from pretext task to downstream task.

C. Justification for Using SSL

We now justify why our scheme of using SSL could be able to learn generalized features. The purpose of the pretext task is to enable networks to learn general features from the aggregate in the target dataset when neither submetering nor user survey is available for labeling loads in the target dataset. The purpose of the downstream task is then to help the network to learn the features of the load disaggregation task from the source dataset with labels and transfer general features learned from the pretext task to load disaggregation.

Denote the midpoint of the aggregated power sequence $y_{t:t+W−1}$ as $y_t$, and the midpoint of the appliance $x_{t:t+W−1}$ as $x_t^m$. Then, a model in pretext task can be trained to predict $y_t$ using two nonlinear functions: $u = f_0(y_{t:t+W−1})$ and $y_t = g_0(u)$, where $u$ is the representation variable. This could be represented by the following probabilistic model:

$$p(y_t|y_{t:t+W−1}) = \int p_0(y_t|u)p_0(u|y_{t:t+W−1})du.$$  \tag{5}

The function $f_0$ is used to learn a feature representation from the aggregate in the target dataset and $g_0$ as a regressor. For predicting $x_t^m$, we could simply learn a function $h_\beta$, such that $x_t^m = h_\beta(y_t)$. Then, we can have

$$p(x_t^m|y_{t:t+W−1}) = \int p_\beta(x_t^m|y_t)p_0(y_t|u)p_0(u|y_{t:t+W−1})du.$$  \tag{6}

We can see that the fine-tuning procedure in downstream task given a learned representation $u$ is to use $u$ to predict $x_t^m$, which means $u$ could be used to predict $x_t^m$ by the following probabilistic model:

$$p(x_t^m|u(y_{t:t+W−1})) = \int p_\beta(x_t^m|y_t)p_0(y_t|u(y_{t:t+W−1}))dy_t.$$  \tag{7}

The network learns the features from the source dataset with labels and transfers general features learned from the pretext task to load disaggregation via updating network parameters in downstream task. It should be noticed that since the values of $y_t$ are all the possible values of the aggregate and the values of $x_t^m$ are only all the possible values of an appliance, $y_t$ has a larger number of possible values than $x_t^m$. Therefore, such $h_\beta$ is an undercomplete representation for $y_t$, and so, $x_t^m$ compresses $y_t$ by using $h_\beta$. This indicates that: 1) we can use $u$ to predict $x_t^m$ and 2) $u$ contains more information to predict $x_t^m$ than the model directly trained using $(y_{t:t+W−1}, x_{t:t+W−1})$.

D. Data Utilization in SSL Against FSL and ZSL

For clearly distinguishing data utilization in load disaggregation based on SSL, FSL, and ZSL, a comparison is demonstrated in Fig. 6.

In an FSL-based NILM method, the network is pretrained on the aggregate with submetering data in the source set and then fine-tuned on labeled data from the target set [34]. Namely, unlike in SSL, limited submetering data are required in FSL. However, in a ZSL-based NILM method, the network trained on the source dataset is only directly tested on the target dataset, requiring no submetering measurements from the target set as in SSL. That is, differing from SSL, no knowledge about power consumption for the target dataset can be learned in ZSL.
from 20 U.K. houses from 2013 to 2015. Since the data from REDD, U.K.-DALE, and REFIT datasets are collected by standard monitoring equipment and widely used in validating NILM approaches, its measurement uncertainty is acceptable with negligible influence on NILM performance. Compared to the other two datasets, REFIT dataset is noisier due to more unknown loads. Note that all data are downsampled to 1 min as low-rate measurements. Since the REFIT dataset is the largest [2], containing more houses with more appliances, it is chosen for fine-tuning in SSL and training in ZSL. The experimental data selected for pretext task training, downstream task training, and testing in SSL are clarified in Table III, as well as that for both training and testing in ZSL.

For exploring the performance boundary of SSL, six experimental cases are settled. In both Cases 1 and 2, pretraining, fine-tuning, and testing in SSL are carried out among houses from the same REFIT dataset. Cases 1 and 2 share the same data from the same house for testing, while they differ in houses for building the pretraining dataset. Data for single House 2 are used for both pretraining and testing in Case 1, while more houses other than House 2 are used for pretraining in Case 2. Similar settings are employed for Cases 3 and 4; however, the data for both pretraining and testing are from REDD houses. For further investigating the impact of pretraining house selection on load disaggregation, we set the size of pretraining data from various houses in Case 6 equivalent to that from single House 1 in Case 5. In terms of ZSL benchmarks, the datasets for training and testing are the same as those for fine-tuning and testing for SSL, respectively. In consideration of the appliance types across datasets, five appliances with the most frequent usage are chosen to be disaggregated, including fridge (F), microwave (M), dishwasher (DW), washing machine (WM), and kettle (K). For each appliance, the data for fine-tuning its network come from houses heuristically picked based on [30].

### B. Evaluation Metrics

Since various metrics have been applied to evaluate NILM performance in the previous works, three widely used metrics are selected in this article. At first, mean absolute error (MAE) [4] is applied to quantify the absolute error between the predicted signal and the GT on average. Namely, MAE is formulated as

\[
MAE = \frac{1}{T} \sum_{t=1}^{T} |\hat{x}_t - x_t| \tag{8}
\]

where the variable \(\hat{x}_t\) denotes the power prediction at time index \(t \in [1, T]\), and \(x_t\) is the simultaneous actual power measurement. Thus, \(T\) is the total number of samples in the signal.

The second metric is normalized signal aggregate error [30], which is defined as

\[
SAE = |\hat{r} - r|/r \tag{9}
\]

where \(r\) and \(\hat{r}\) refer to the actual total energy consumption and its prediction, respectively. SAE measures the relative error of the total energy.

Furthermore, metric energy per day (EpD) [2] is adopted, which offers the average error of daily appliance-level energy consumption prediction. EpD is defined as

\[
EpD = \frac{1}{D} \sum_{n=1}^{D} |\hat{e}_n - e_n|/D \tag{10}
\]

where \(D\) denotes the total number of days, \(e_n\) is the actual daily energy consumption, and \(\hat{e}_n\) refers to the prediction. Not like MAE, where errors are calculated sample-by-sample, corresponding to power, SAE and EpD are calculated for energy consumption. However, EpD reflects daily errors while SAE is normalized for the entire period.

### C. Experimental Settings

Our program is implemented in Python using TensorFlow. The networks were trained on machines with NVIDIA GeForce RTX 2080 Ti. Networks are trained by ADAM optimizer algorithm, with an early-stopping mechanism to prevent overfitting as in [2], where the optimal network is picked with a validation loss lower than those in the following

### TABLE III

| Case | pretext task training data set for SSL | Samples (×10³) | Downstream task training data set for SSL | Training data set for ZSL | Testing data set for both SSL and ZSL | Samples (×10³) |
|------|------------------------------------|----------------|------------------------------------------|--------------------------|-------------------------------------|----------------|
| 1    | 2                                  | 2013-09-17 to 2015-02-06 | 0.54                                      |                          |                                     |                |
| 2    | REFIT                             | 16              | 0.72                                      |                          |                                     |                |
| 3    | 1                                  | 2011-04-18 to 2011-05-11 | 0.02                                      |                          |                                     |                |
| 4    | REDD                              | 2               | 0.02                                      |                          |                                     |                |
| 5    | 1                                  | 2012-12-14 to 2017-04-26 | 2.24                                      |                          |                                     |                |
| 6    | UK-DALE                           | 5               | 1.77                                      |                          |                                     |                |

**B. Evaluation Metrics**

Since various metrics have been applied to evaluate NILM performance in the previous works, three widely used metrics are selected in this article. At first, mean absolute error (MAE) [4] is applied to quantify the absolute error between the predicted signal and the GT on average. Namely, MAE is formulated as

\[
MAE = \frac{1}{T} \sum_{t=1}^{T} |\hat{x}_t - x_t| / T \tag{8}
\]

where the variable \(\hat{x}_t\) denotes the power prediction at time index \(t \in [1, T]\), and \(x_t\) is the simultaneous actual power measurement. Thus, \(T\) is the total number of samples in the signal.

The second metric is normalized signal aggregate error [30], which is defined as

\[
SAE = |\hat{r} - r| / r \tag{9}
\]

where \(r\) and \(\hat{r}\) refer to the actual total energy consumption and its prediction, respectively. SAE measures the relative error of the total energy.

Furthermore, metric energy per day (EpD) [2] is adopted, which offers the average error of daily appliance-level energy consumption prediction. EpD is defined as

\[
EpD = \frac{1}{D} \sum_{n=1}^{D} |\hat{e}_n - e_n| / D \tag{10}
\]

where \(D\) denotes the total number of days, \(e_n\) is the actual daily energy consumption, and \(\hat{e}_n\) refers to the prediction. Not like MAE, where errors are calculated sample-by-sample, corresponding to power, SAE and EpD are calculated for energy consumption. However, EpD reflects daily errors while SAE is normalized for the entire period.

**C. Experimental Settings**

Our program is implemented in Python using TensorFlow. The networks were trained on machines with NVIDIA GeForce RTX 2080 Ti. Networks are trained by ADAM optimizer algorithm, with an early-stopping mechanism to prevent overfitting as in [2], where the optimal network is picked with a validation loss lower than those in the following
six epochs. The batch size is set to 512 for REDD dataset, as in Cases 3 and 4, while in other cases, it is set to 1024 for REFFIT and U.K.-DALE datasets to accelerate data processing. Meanwhile, we set the learning rate to 0.001 as in [2]. The data normalization used in our experiments is the same as in [9]. In Bi-GRU, dropout rate is set to 0.5. Since the data sampling rates in this article differ from those in [10] and [22], the window lengths for Bi-GRU and LSTM are heuristically characterized as ten for WMs due to their longer operational duration and five for other appliances, for guaranteeing the same temporal settings. For similar reason, the general window length is set to 79 for all appliances in S2p, based on that set in [9].

V. EXPERIMENTAL RESULT

In this section, the overall experimental results are initially demonstrated, as for all cases, SSL is evaluated against ZSL on S2p, Bi-GRU, and LSTM networks in three metrics for each appliance. Then, the total energy consumption per appliance estimated in all experiments is illustrated, followed by a series of examples of disaggregated power signals. Finally, SSL and ZSL are compared for their total training time and execution time per sample. For simplification, SSL with full fine-tuning is abbreviated as SSL. The abbreviations of such appliances are: F for fridge; M for microwave, DW for dishwasher, WM for washing machine, and K for kettle.

A. NILM Performance Metrics Comparison

The comparison between load disaggregation performance of ZSL and that of SSL is demonstrated in Table IV for S2p, Table V for Bi-GRU, and Table VI for LSTM. For a clear expression, hyphens are added for combining disaggregation networks and the approaches they are trained.

1) Within the Same Dataset: Since only REFFIT dataset is used in both Cases 1 and 2, their results demonstrate load disaggregation performance of SSL and ZSL on S2p, Bi-GRU, and LSTM networks in three metrics for each appliance. Then, the total energy consumption per appliance estimated in all experiments is illustrated, followed by a series of examples of disaggregated power signals. Finally, SSL and ZSL are compared for their total training time and execution time per sample. For simplification, SSL with full fine-tuning is abbreviated as SSL. The abbreviations of such appliances are: F for fridge; M for microwave, DW for dishwasher, WM for washing machine, and K for kettle.

2) Across Datasets: In both Cases 3 and 4, the dataset for both training in ZSL and fine-tuning in SSL remains the same as in Cases 1 and 2, while the data in other stages are switched to REDD dataset. The abbreviations of such appliances are: F for fridge; M for microwave, DW for dishwasher, WM for washing machine, and K for kettle.
in the REFIT dataset. Such distinction in load characteristics also affects the disaggregation results for M. MAE results for both M and DW are increased by applying SSL to Bi-GRU, while in Case 3, the superiority of S2p-SSL in both SA and EpD is caused by lower power estimation for misidentified loads. In Case 3, although EpD results are not improved by LSTM-SSL comparing to LSTM-ZSL, the improvements on the other two metrics are significant, while in Case 4, dominating performance is achieved by LSTM-SSL. Since SSL generally outperforms others in Cases 3 and 4, the results show the knowledge transferability between REDD dataset and REFIT dataset.

For the remaining experiments in Cases 5 and 6, data for self-supervised pretraining in SSL and testing in both ZSL and SSL are selected from U.K.-DALE dataset. From Table IV, applying SSL to S2p generally achieves the best disaggregation performance in both cases. Similar experimental results can be found in Cases 5 and 6 on LSTM framework, showing general superiority of LSTM-SSL. Slightly poor performance for M in Case 5 is because of waveformwise and temporal pattern distinction between datasets for pretraining and testing. However, improvement is tiny for Bi-GRU. Moreover, replacing the data collected from a single house by that from multiple houses while maintaining the data size seems not to have a certain impact on load disaggregation performance, e.g., close performance for fridges in Cases 5 and 6 is due to the high similarity in power characteristics of their operational cycles. Significant performance improvement is achieved by SSL for M from Cases 5 and 6, due to less misidentification. Bi-GRU-SSL outperforms Bi-GRU-ZSL for DW in energy estimation; however, Bi-GRU-ZSL achieves better results in sample-by-sample disaggregation results.

It can be concluded from the whole experimental results that self-supervised pretraining helps NILM performance improvement against ZSL, especially for S2p. Furthermore, S2p-SSL generally performs better than Bi-GRU-SSL and LSTM-SSL across metrics. With respect to transferability between datasets, the overall results show that U.K.-DALE and REFIT datasets, both collected from the U.K., have stronger transferable potential than the datasets from distinct regions. Moreover, the increase in pretraining data size leads to performance improvement for SSL. In terms of disaggregation performance for each appliance, the poor results for M are achieved in various cases, as little information can be captured
due to its short duration and low usage frequency. For more complex WM loads with multiple operational states, their characteristics differ a lot across houses and datasets, resulting in underestimation for power ranges while overestimation for operation times.

### B. Energy Consumption Estimation

The power consumption estimation results for each appliance are demonstrated as in Fig. 7. In both Cases 1 and 2, power consumption close to the GT is estimated by various methods for F, M, and K, with ON/OFF operational states. However, they differ in power consumption estimation for multistate DW and WM, respectively, where Bi-GRU and LSTM suffer from overestimation. Around 8% and 16% samples in power signals for DW and WM in the fine-tuning sets are in stand-by states below 10 W, collected from both REFIT House 5 and House 7. The stand-by power samples are estimated by Bi-GRU and LSTM for DW and WM, while they cannot be observed in the testing set. Similar results can be observed in Cases 3 and 4. Moreover, the generally worse performance of Bi-GRU and LSTM than S2p in estimating power consumption for F in Cases 3 and 4 is also caused by the distinction between the power characteristics of F in the fine-tuning set and those for the testing set. The worse performance of Bi-GRU and LSTM as underestimation for K in Cases 5 and 6 can be explained by the same reason. Thus, we can conclude that Bi-GRU and LSTM are more sensitive to the distinction between distribution in training and testing sets. Note that unknown loads consume 73% in Cases 1 and 2, 65% in Cases 3 and 4, and 56% in Cases 5 and 6 of the total energy; thus, overestimation is more serious in Cases 1 and 2 due to more misidentification.
Among all six cases, S2p-SSL generally outperforms LSTM-SSL and Bi-GRU-SSL in total energy estimation, in line with the results shown in Table IV–VI. On the whole, SSL helps to improve such networks in power consumption estimation performance.

C. Load Power Disaggregation

Furthermore, Case 6 is selected as an example for demonstrating load disaggregation results. The estimation of power consumed by five target appliances in Case 6 is shown in Fig. 8. From Fig. 8, the operating periods for each target appliance are generally identified by these methods. Furthermore, the superiority of NILM methods based on S2p and LSTM over those based on Bi-GRU can be observed, with disaggregated power signals closer to the GT for most appliances. Among these methods, underestimation tends to be common for most appliances. Possible reasons can be inferred as the mapping construction is affected by short-lasting and less-frequent usage of M and multistate WM operating in various power ranges. An exception is DW, which is overestimated by both S2p-ZSL and LSTM-ZSL, as DWs from REFIT houses operate in higher power ranges than those from U.K.-DALE houses.

D. Training Time Analysis

In this section, the total training time in both Cases 4 and 5 is demonstrated in Fig. 9 as instances. In Case 4, the total training time of ZSL-based methods and that of SSL-based methods are close except for F in LSTM-ZSL. As the network convergence duration of LSTM is more sensitive to the distinction between the operational patterns of F in the training dataset, including power ranges and periodicity. Since M operation usually lasts within several minutes, its distinction on power profile across datasets and houses is trivial. Therefore, close training time is achieved for M. However, the amount of data for pretraining in Case 5 is over 20 times of that in Case 4, resulting in longer training time across models. It is noteworthy that applying SSL to both Bi-GRU and LSTM requires longer training time for multistate appliances such as WM, due to its longest sliding window compared to other appliances. In conclusion, the amount of data for pretraining should be carefully chosen as it balances NILM performance and efficiency.

For evaluating the influence of labeled data amount on execution time, the execution time per sample for all methods is shown in Table VII.

From Table VII, generally close execution time per sample can be observed for SSL and ZSL on all networks, especially for Bi-GRU. For S2p, the mean execution time per sample of
SSL is longer than that of ZSL by about 49 s, while for LSTM, the mean execution time per sample of ZSL is longer mainly contributed by F, as its power characteristics varying across houses in the training dataset result in longer convergence duration. However, in SSL, the pretext task helps to accelerate convergence in the downstream task.

VI. CONCLUSION

In this article, SSL is applied to the state-of-the-art NILM solutions based on S2p, Bi-GRU, and LSTM. A pretext task is proposed to pretrain a general network to map aggregate power sequences to derived representatives, where labeled data for the target dataset are not required. By performing a supervised downstream task based on the labeled data from source datasets, the pretrained network is fine-tuned with feature transference and then applied to disaggregate loads for the target sites. Publicly accessible REDD, U.K.-DALE, and REFIT datasets are utilized for validation, with multiple experimental cases designed to evaluate the performance of SSL using data within the same set and across datasets. For a comprehensive NILM performance demonstration, various metrics are employed, showing SSL generally outperforms ZSL scheme. Specifically, the features learned via SSL help to improve underestimation, reduce misidentification, and mitigate the influence of wrong labels. Such conclusions are also supported by the disaggregated power signals for each appliance and their total power consumption estimation results. Thus, SSL contributes to network refinement and disaggregation performance improvement, with limited increment in execution time per sample.

Future work includes investigation on the influence of activation balancing for compensating the skew of training data, analyzing the impact of sampling frequency of the data for fine-tuning in the source domain on the NILM performance, and applying autoencoder to carry out the pretext task.

REFERENCES

[1] Y. Yu, B. Liu, and W. Luan, “Nonintrusive residential load monitoring and decomposition technology,” Southern Power Syst. Technol., vol. 7, no. 4, pp. 1–5, Aug. 2013.
[2] M. D’Incecco, S. Squarzini, and M. Zhong, “Transfer learning for non-intrusive load monitoring,” IEEE Trans. Smart Grid, vol. 11, no. 2, pp. 1419–1429, Mar. 2020.
[3] B. Liu and W. Luan, “Conceptual cloud solution architecture and application scenarios of power consumption data based on load disaggregation,” Power Syst. Technol., vol. 40, no. 3, pp. 791–796, 2016.
[4] G. Cui, B. Liu, W. Luan, and Y. Yu, “Estimation of target appliance electricity consumption using background filtering,” IEEE Trans. Smart Grid, vol. 10, no. 6, pp. 5920–5929, Nov. 2019.
[5] G. W. Hart, “Nonintrusive appliance load data acquisition method: Progress report,” MIT Energy Lab., Cambridge, MA, USA, Tech. Rep., 1984.
[6] M. Zeifman and K. Roth, “Nonintrusive appliance load monitoring: Review and outlook,” in Proc. IEEE Int. Conf. Consum. Electron. (ICCE), Las Vegas, NV, USA, Jan. 2011, pp. 76–84.
[7] P. Huber, A. Calatroni, A. Rumsch, and A. Paice, “Review on deep neural networks applied to low-frequency NILM,” Energies, vol. 14, no. 9, p. 2390, Apr. 2021.
[8] J. R. Herrero et al., “Non intrusive load monitoring (NILM): A state of the art,” in Proc. Int. Conf. Practical Appl. Agents Multi-Agent Syst. Cham, Switzerland: Springer, 2017, pp. 125–138.
[9] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, “Sequence-to-point learning with neural networks for non-intrusive load monitoring,” in Proc. AAAI Conf. Artif. Intell., 2018, vol. 32, no. 1, pp. 1–8.
[10] O. Krystalakos, C. Nalmpantis, and D. Vrakas, “Sliding window approach for online energy disaggregation using neural network,” in Proc. 10th Hellenic Conf. Artif. Intell., Jul. 2018, pp. 1–6.
[11] B. Zhao, M. Ye, L. Stankovic, and V. Stankovic, “Non-intrusive load disaggregation solutions for very low-rate smart meter data,” Appl. Energy, vol. 268, Jun. 2020, Art. no. 114949.
[12] M. Zhong, N. Goddard, and C. Sutton, “Signal aggregate constraints in additive factorial HMMs, with application to energy disaggregation,” in Proc. 27th Int. Conf. Neural Inf. Process. Syst., vol. 27. Cambridge, MA, USA: MIT Press, 2014, pp. 3590–3598.
[13] J. Z. Kolter and T. Jaakkola, “Approximate inference in additive factorial HMMs, with application to energy disaggregation,” in Proc. 15th Int. Conf. Artif. Intell. Statist., 2012, pp. 1472–1482.
[14] M. Zhong, N. Goddard, and C. Sutton, “Interleaved factorial non-homogeneous hidden Markov models for energy disaggregation,” 2014, arXiv:1406.7665.
[15] L. Mauch and B. Yang, “A new approach for supervised power disaggregation by using a deep recurrent LSTM network,” in Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP), Orlando, FL, USA, Dec. 2015, pp. 63–67.
[16] L. Mauch and B. Yang, “A novel DNN-HMM-based approach for extracting single loads from aggregate power signals,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Shanghai, China, Mar. 2016, pp. 2384–2388.
[17] P. G. Donato, A. Hernandez, M. A. Funes, I. Carugati, R. Nieto, and J. Urena, “Review of NILM applications in smart grids: Power quality assessment and assisted independent living,” in Proc. Argentine Conf. Automat. Control (AADECA), Oct. 2020, pp. 1–6.
[18] N. Henao, K. Agbossou, S. Kelouwani, Y. Dubé, and M. Fournier, “A framework in nonintrusive type I load monitoring using subtractive clustering,” IEEE Trans. Smart Grid, vol. 8, no. 2, pp. 812–821, Mar. 2017.
[19] B. Zhao, L. Stankovic, and V. Stankovic, “Blind non-intrusive appliance load monitoring using graph-based signal processing,” in Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP), Orlando, FL, USA, Dec. 2015, pp. 68–72.
[20] B. Zhao, L. Stankovic, and V. Stankovic, “On a training-less solution for non-intrusive appliance load monitoring using graph signal processing,” IEEE Access, vol. 4, pp. 1784–1790, 2016.
[21] J. Kelly and W. Knottenbelt, “Neural NILM: Deep neural networks applied to energy disaggregation,” in Proc. 2nd ACM Int. Conf. Embedded Syst. Energy-Efficient Built Environ., 2015, pp. 55–64.
[22] J. Zhang, J. Sun, J. Gan, Q. Liu, and X. Liu, “Improving domestic NILM using an attention-enabled Seq2Point learning approach,” in Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput., Int. Conf. Pervasive Grid Syst. Energy-Efficient Built Environ., 2015, pp. 55–64.
[23] T.-T.-H. Le, J. Kim, and H. Kim, “Classification performance using gated recurrent unit recurrent neural network on energy disaggregation,” in Proc. Int. Conf. Mach. Learn. Cybern. (ICMLC), Jul. 2016, pp. 105–110.
[24] R. Bonfigli, A. Felicetti, E. Principi, M. Fagiani, S. Squarzini, and F. Profumo, “Denoising autoencoders for non-intrusive load monitoring: Improvements and comparative evaluation,” Energy Buildings, vol. 158, pp. 1461–1474, Jan. 2018.
[26] J. Chen, X. Wang, X. Zhang, and W. Zhang, “Temporal and spectral feature learning with two-stream convolutional neural networks for appliance recognition in NILM,” *IEEE Trans. Smart Grid*, vol. 13, no. 1, pp. 762–772, Jan. 2022.

[27] D. Yang, X. Gao, L. Kong, Y. Pang, and B. Zhou, “An event-driven convolutional neural architecture for non-intrusive load monitoring of residential appliance,” *IEEE Trans. Consum. Electron.*, vol. 66, no. 2, pp. 173–182, May 2020.

[28] P. Paulo, “Applications of deep learning techniques on NILM,” Ph.D. dissertation, Dept. Elect. Eng., Federal Univ. Rio de Janeiro, Rio de Janeiro, Brazil, 2016.

[29] D. Kelly, “Disaggregation of domestic smart meter energy data,” Ph.D. dissertation, Comput. Univ. London, Diploma Imperial College, London, U.K., 2016.

[30] L. Wang, S. Mao, and R. M. Nelms, “Transformer for nonintrusive load monitoring: Complexity reduction and transferability,” *IEEE Internet Things J.*, vol. 9, no. 13, pp. 18987–18997, Oct. 2022.

[31] J. Barber, H. Cuayáhuitl, M. Zhong, and W. Luan, “Lightweight non-intrusive load monitoring employing pruned sequence-to-point learning,” in *Proc. 5th Int. Workshop Non-Intrusive Load Monit.*, Nov. 2020, pp. 11–15.

[32] A. M. A. Ahmed, Y. Zhang, and F. Eliassen, “Generative adversarial networks and transfer learning for non-intrusive load monitoring in smart grids,” in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Tempe, AZ, USA, Nov. 2020, pp. 1–7.

[33] Z. Yue, C. R. Witzig, D. Jorde, and H. Jacobsen, “BERT4NILM: A bidirectional transformer model for non-intrusive load monitoring,” in *Proc. 5th Int. Workshop Non-Intrusive Load Monit.*, 2020, pp. 89–93.

[34] L. Wang, S. Mao, B. M. Wilamowski, and R. M. Nelms, “Pre-trained models for non-intrusive appliance load monitoring,” *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 56–68, Mar. 2022.

[35] D. Murray, L. Stankovic, V. Stankovic, S. Luke, and S. Sladojevic, “Transferability of neural network approaches for low-rate energy disaggregation,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Brighton, U.K., May 2019, pp. 8330–8334.

[36] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, “Using self-supervised learning can improve model robustness and uncertainty,” 2019, arXiv:1906.12340.

[37] L. Jing and Y. Tian, “Self-supervised visual feature learning with deep neural networks: A survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, Nov. 2021.

[38] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, “Learning problem-agnostic speech representations from multiple self-supervised tasks,” 2019, arXiv:1904.03416.

[39] J. Z. Kolter and M. J. Johnson, “REDD: A public data set for energy disaggregation research,” in *Proc. Workshop Data Mining Appl. Sustainability (SIGKDD)*, San Diego, CA, USA, 2011, pp. 59–62.

[40] J. Kelly and W. Knottenbelt, “The U.K.-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five U.K. homes,” *Sci. Data*, vol. 2, no. 1, pp. 1–14, 2015.

[41] D. Murray, L. Stankovic, and V. Stankovic, “An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study,” *Sci. Data*, vol. 4, no. 1, pp. 1–12, 2017.

Shuyi Chen received the B.S. degree in electrical engineering from China Agricultural University, Beijing, China, in 2020. She is currently pursuing the M.S. degree in electrical engineering with Tianjin University, Tianjin, China.

Her current research interests include nonintrusive load monitoring and deep learning.

Bochao Zhao (Member, IEEE) received the B.Eng. degree (Hons.) in electronic and electricity engineering and the Ph.D. degree in electronic and electricity engineering from the University of Strathclyde, Glasgow, U.K., in 2014 and 2020, respectively.

He is currently a Lecturer with the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. His current research interests include demand response, energy trading in virtual power plants, energy disaggregation, and graph signal processing applications.

Mingjun Zhong (Member, IEEE) received the B.Sc. and Ph.D. degrees in applied mathematics from the Dalian University of Technology, Dalian, China, in July 1999 and July 2004, respectively.

He was an Associate Professor with Dalian Nationalities University, Dalian. He was a Post-Doctoral Researcher with INRIA Rennes, Rennes, France, from December 2015 to December 2016. He was a Research Assistant with the School of Computing, University of Glasgow, Glasgow, U.K., from February 2008 to October 2011. He was a Research Associate with the School of Informatics, The University of Edinburgh, Edinburgh, U.K., from July 2013 to April 2017. He was an Associate Professor with the Dalian University of Technology. He was a Senior Lecturer with the University of Lincoln, Lincoln, U.K. He is currently a Lecturer with the Department of Computing Science, University of Aberdeen, Aberdeen, U.K. His current research interests include machine learning, computational statistics, and artificial intelligence.

Wenpeng Luan (Senior Member, IEEE) received the B.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 1986, the M.S. degree in electrical engineering from Tianjin University, Tianjin, China, in 1989, and the Ph.D. degree in electrical engineering from Strathclyde University, Glasgow, U.K., in 1999.

After long industry working experiences, he joined Tianjin University, as a Professor, in 2019. His current research interests include smart metering data analytics, nonintrusive load monitoring, distribution system analysis, renewable energy resource integration, and utility advanced applications.

Yixin Yu (Life Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, China, in 1958 and 1963, respectively.

He joined Tianjin University, as a Full Professor, in 1985. He was a Visiting Scholar with the Department of EECS, University of California at Berkeley, Berkeley, CA, USA, from 1980 to 1982, and a Chair Visiting Professor with the Kyushu Institute of Technology, Kitakyushu, Japan, from 1995 to 1997.

His current research interests include power system stability and security, power distribution planning, and smart grid.

Mr. Yu was elected as an Academician of the Chinese Academy of Engineering, in 2005.