# Harnessing Temporal Awareness: Dual-CNN Model for Non-Intrusive Load Monitoring

Hengxin Wu, Yechun Ruan, and Qianyi Huang, *Member, IEEE*

***Abstract*—Thoroughly monitoring the energy consumption of each appliance in a household is essential to assist users in better engaging in power-saving practices. To get per-appliance electrical profile, non-intrusive load monitoring (NILM), also known as energy disaggregation, is a blind source separation task, that decomposes the total household electricity consumption into the load profile of each individual appliance. In recent years, deep learning models have demonstrated superior performance in NILM tasks. However, when encountering an unseen household, the model performance degrades seriously. In order to improve the performance on unseen households, we observe that temporal information is beneficial for the model generalization ability, as the ON/OFF patterns of electrical appliances follow a similar time schedule across different households. Thus, in this paper, we propose a time-aware dual-CNN architecture for NILM (TimeNILM), which fully exploits the temporal information and employs feature fusion based on attention mechanism to improve model performance. Experimental results on two datasets (REDD and UK-DALE) demonstrate that our model outperforms state-of-the-art models, achieving 8%-27% mean absolute error (MAE) gain and 8%-35% signal aggregated error (SAE$_\delta$) gain for unseen households. As appliance usage patterns are private data, in order to protect user privacy, we extend TimeNILM to federated settings and propose a scheme for aggregation during the learning process. Furthermore, we have successfully deployed TimeNILM on edge devices, and performance evalutation indicates that the model is capable of real-time load monitoring on a Raspberry Pi.**

***Index Terms*—Non-intrusive load monitoring, temporal information, dual-CNN architecture, feature fusion, federated learning**

## I. Introduction

THE residential sector accounts for a major portion of the overall electricity usage in society. According to recent research, residential buildings account for 40% of total energy consumption in the United States [1]. It is disturbing that at least 30% of electricity in residential consumption is wasted [2]. Although the governments of the United States and the United Kingdom have deployed a substantial number of smart meters in residential settings, these smart meters can only measure whole-home electricity consumption, without detailed information of each appliance. Thoroughly monitoring the energy consumption of each appliance in a household is essential to assist users in better engaging in power-saving practices, so that users can identify power-hungry appliances and make corresponding power-saving efforts.

Non-intrusive load monitoring (NILM), was first proposed by Hart in 1992. In NILM, the aggregated whole-home power consumption is decomposed into per-appliance power consumption, which saves us from deploying a dedicated sensor for each appliance. In recent years, deep learning has emerged as a popular approach for NILM [3], [4]. Considering the characteristics of electrical appliances, state-of-the-art models are proposed. In Subtask Gating Network (SGN) [5], the model consists of a classification sub-network and a regression sub-network, where the classification sub-network predicts the ON/OFF state of the appliance and the regression network predicts the power level of the appliance. More recently, Multi-State Dual CNN (MSDC) considers appliances with multiple states and utilizes Conditional Random Fields (CRF) to capture state transitions [6], leading to further improvements in power consumption prediction accuracy.

Existing deep learning models for NILM are typically trained using a substantial amount of labeled data [7]. However, when encountering unseen households, the model's generalization ability becomes unsatisfactory. This task is highly challenging as the power consumption level of an appliance is influenced by a number of factors. Take the fridge as an example. Fridges are made by different manufacturers and even the same manufacturer can produce various models. Even users' configurations (temperature settings) will also affect the power consumption level. Thus, the model performance degrades when encountering an unseen household.

In order to improve the performance on unseen households, we observe that temporal information can help improve model generalization ability. The active patterns of appliances are similar across different households. For example, the microwave is more active during mealtime while the dishwasher is usually run at night. By considering the temporal information in the model design, we can identify the appliances' states more accurately. Furthermore, it can also help to identify key moments when the power level (i.e., state) changes, so that we can estimate power level more accurately. Unfortunately, most state-of-the-art NILM models [5], [8], [9] do not explicitly incorporate timestamps as model input.

Though some studies use RNN or Transformer architecture to extract temporal information [14], [15], they primarily capture the trending patterns in time series data and extract temporal dependencies among samples, making them more suitable for capturing global correlations and time dependencies. In our NILM task, we are mainly concerned with the relationship between time information at each specific time point and

Hengxin Wu and Qianyi Huang are with the School of Computer Science, Sun Yat-Sen University, Guangzhou 510006, China; Yechun Ruan is with the Institute of Future Networks, Southern University of Science and Technology, Shenzhen 518055, China. Q. Huang is the corresponding author (e-mail: huangqy89@mail.sysu.edu.cn).

This work was supported in part by the National Natural Science Foundation of China (Grant No. 62472452); in part by Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515010262)

the corresponding electrical appliance power. Therefore, our goal is to utilize CNN to identify the appliances' ON/OFF schedule across days through convolutional operations, aiming to effectively capture local patterns at each time point and more accurately model and extract information relevant to specific moments.

In this paper, in order to enhance the model's generalization ability, we propose TimeNILM, which incorporates temporal-domain information in NILM task. Different from existing dual-CNN models, in TimeNILM, the classification subnetwork takes time-domain information along with the aggregated power level as the model input. To help the regression subnetwork identify key moments when the power level will change, the two subnetworks will perform feature fusion with attention mechanism so that the regression subnetwork can also benefit from the time-domain information. Furthermore, we use Kullback-Leibler Divergence Loss (KLDivLoss) [16] to minimize the feature distance between the source domain and target domain. Through experiments conducted on the REDD and UK-DALE datasets, we demonstrate that our model improve MAE by $8\%$ to $27\%$ and SAE$_\delta$ by $8\%$ to $35\%$ in power prediction accuracy compared with state-of-the-art models.

As the appliance active patterns and the users' usage habits are quite private and sensitive, users are reluctant to share their household power consumption data. Thus, we also extend TimeNILM to federated settings. We propose Fed-TimeNILM to enable different clients to aggregately improve the classification subnetwork while prevent the hetergenous power levels from other clients degrading the performance of regression subnetwork.

Our main contributions can be summarized as follows:

1) We incorporate the timestamp information into the dual-CNN models for NILM, which can help the model generalize to unseen households.
2) In the dual-CNN architecture, we employ the attention-based feature fusion, which can avoid overfitting of power level. We further extend TimeNILM to federated settings, improving model generalization across multiple households.
3) We demonstrate that our model outperforms state-of-the-art models (8%-27% MAE gain and 8%-35% SAE$_\delta$ gains) with Raspberry Pi deployment, validating real-time feasibility.

## II. Related Works

[17] first proposed NILM, a method that estimates the power usage of individual appliances based on the overall household power consumption. The most commonly used approach in NILM is the Hidden Markov Model (HMM) and its variants. [18] utilized the sparsity of HMM to achieve real-time non-intrusive load monitoring, employing the super-state Hidden Markov Model and sparse Viterbi algorithm for load monitoring. [19] used the Factorial Hidden Markov Model (FHMM) to model multiple appliances and extended the Viterbi algorithm with integer programming for optimal allocation. [20] combined active and reactive power in the Additive Factorial Hidden Markov Model (AFHMM) for non-intrusive load monitoring.

In recent years, with the popularity of deep learning, a series of new methods [21], [22] have been introduced to address the NILM problem, mainly applied to low-frequency NILM methods. Recurrent Neural Networks (RNNs) can leverage the temporal dependencies of power signal, making them suitable for NILM problems [23]. Convolutional Neural Networks (CNNs) are popular deep learning models that have achieved state-of-the-art performance in various tasks [24]–[27]. In the NILM problem, various CNNs have been used for regression and classification. [9] introduced the Sequence-to-Point model, where the input is the aggregated power sequence and the output is the midpoint in the sequence for the target appliance. Inspired by the Sequence-to-Point model, [28] applied pruning algorithms to reduce the number of weights, aiming to reduce the parameter number without sacrificing performance, making the model deployable on mobile devices. Inspired by multi-task learning, [5] proposed the Subtask Gating Network (SGN), which employs two subnetworks that simultaneously perform regression and classification, allowing the neural network to learn multiple tasks by combining the regression and the classification output to construct the final representation. [8] proposed the Multi-State Dual CNN (MSDC), which considers multiple appliance states while utilizing CRF to capture patterns of state transitions.

However, when encountering unseen households, the model performance suffers. There are two possible reasons. On one hand, state-of-the-art models [5], [8], [9] overlook the timestamp information in the collected data and rely solely on the aggregated power sequence to predict the per-appliance power consumption. Without time-domain information, it is hard for the knowledge that learned from the source domain to transfer to the target domain. On the other hand, although SGN and MSDC propose a dual-CNN architecture, they do not further investigate the connection between the regression subnetwork and the classification subnetwork.

In recent years, with the advancements in the field of deep learning, federated learning (FL) has emerged to address data privacy concerns. Federated learning is a distributed machine learning technique that revolves around the idea of decentralized model training across multiple data sources, each having its own local dataset. It achieves the goal of data privacy protection and collaborative computation by constructing a global model based on virtual federated data, all without the need to exchange individual or sample data [10], [11]. [12] introduced FedNILM, which employs FL to protect user data privacy while achieving accurate personalized energy disaggregation. Additionally, [13] proposed DPFL model for training and testing NILM using FL, which exhibits greater robustness in Federated Learning compared to other non-privacy models.

## III. Problem Formulation

Given appliances in a household, we denote their aggregated power consumption at time $t$ by $X_t$. The task of NILM is to infer the power consumption for individual appliance $i$

TABLE I
SUMMARY OF NOTATIONS

| Notation | Description |
|---|---|
| **Variables** | |
| $t$ | Time step index |
| $X_t$ | Aggregated power consumption at time $t$ |
| $x_t^i$ | Power consumption of appliance $i$ at time $t$ |
| $u_t$ | Power consumption of unknown appliances at time $t$ |
| $\epsilon_t$ | Noise term at time $t$ |
| $\tau$ | Time-dimension input |
| $M$ | Number of appliance states |
| $\gamma$ | State classification threshold |
| $y_{i,j}$ | Ground truth label indicating if $i$-th sample belongs to state $j$ |
| $p_{i,j}$ | Predicted probability that $i$-th sample belongs to state $j$ |
| $y_i$ | Ground truth value for $i$-th training sample |
| $\hat{y}_i$ | Predicted value for $i$-th training sample |
| $l_i$, $l_o$ | Input / Output sequence lengths |
| **Model Inputs and Outputs** | |
| $\hat{\boldsymbol{y}}_{reg}$, $\hat{\boldsymbol{y}}_{cls}$ | Output of regression and classification subnetworks |
| $\boldsymbol{x}_{reg}$ | Input to the regression subnetwork: aggregated power |
| $\boldsymbol{x}_{cls}$ | Input to the classification subnetwork: concatenated input of power and temporal data |
| $\boldsymbol{f}_r$, $\boldsymbol{f}_c$ | Feature vectors from regression/classification subnetwork |
| $\boldsymbol{W_q}$, $\boldsymbol{W_k}$, $\boldsymbol{W_v}$ | Trainable parameters in attention mechanism |
| $\boldsymbol{\alpha}$ | Attention weight for feature fusion |
| $\hat{\boldsymbol{y}}$ | Predicted power of the appliance |
| $s_m$ | Predicted probability of being in state $m$ |
| $p_m$ | Predicted power level of state $m$ |
| $\theta_{reg}$, $\theta_{cls}$ | Parameters of regression/classification subnetwork |
| **Loss Functions** | |
| $L_{power}$ | Power prediction loss (MSE) |
| $L_{state}$ | State prediction loss (cross-entropy) |
| $L_{KLDiv}$ | Kullback-Leibler divergence loss |
| $L_{transfer}$ | Feature alignment loss |
| $L_{similarity}$ | Cosine similarity loss for federated learning |
| $L_1$ | Joint loss in training stage: $L_{power} + L_{state}$ |
| $L_2$ | Retraining loss: $L_{power}$ only |
| $L_3$ | Transfer learning loss |
| **Evaluation Metrics** | |
| $MAE$ | Mean Absolute Error |
| $SAE_\delta$ | Signal Aggregated Error over time period $\delta$ |

($i = 1, ..., N$) at each time step, i.e., $(x_1^i, ..., x_t^i)$. Typically, we are more interested in power-hungry appliances, which consume the majority of energy. Thus, to make things easier, we treat other appliances as unknown factors, and their power consumption is denoted by $U = (u_1, ..., u_t)$. At each time step, $X_t$ is the sum of individual appliances plus a noise term $\epsilon_t$, which can be represented as

$$X_t = \sum_{i=1}^{N} x_t^i + u_t + \epsilon_t \tag{1}$$

Similar to MSDC [8], we assume that home appliances have multiple active states. Specifically, we first divide the states of an appliance into ON/OFF states, representing whether the appliance is turned on or off. Then, within the ON state, we further divide the appliance into multiple active states, corresponding to different levels of power consumption. To facilitate understanding of the symbols used throughout this paper, we summarize the notations in Table I.

## IV. TimeNILM Model

### A. Overview

As shown in Figure 1, We train a TimeNILM model for each appliance, which consists of two subnetworks: a regression subnetwork and a classification subnetwork. For both subnetworks, the fundamental architecture comprises six convolutional layers followed by two fully connected layers, which are used to extract and integrate features before mapping them to the output. The regression subnetwork takes aggregated power consumption as input and predicts the appliance power consumption. The classification subnetwork takes both temporal data and aggregated power data as input and predicts the appliance state. We only incorporate temporal data in the classification subnetwork, but not in the regression subnetwork. This is because that although their state transitions may exhibit certain similarities, the power level for the same type of appliances (different manufacturers/models) may vary. Therefore, incorporating temporal data into the regression subnetwork may potentially compromise the model's capacity for generalization. For the regression subnetwork to better take advantage of temporal data, after the model has been trained for multiple epochs, we employ feature fusion based on attention mechanism to transfer knowledge from the classification subnetwork to the regression subnetwork. It helps the regression subnetwork to identify key moments (when appliances have state transitions) while avoiding the overfitting of power levels. In TimeNILM, the final predicted value is the element-wise multiplication of the regression subnetwork result and the classification subnetwork result.

### B. The classification subnetwork

Figure 2 shows the active patterns of washing machines in different households collected over 7 days. We can observe that between 10:00am and 12:00am, the washing machines are quite active. Similar observations can be found in other appliances. It implies that, by considering the temporal domain information in model design, the model can better generalize to unseen households.

In the classification subnetwork, we utilize aggregated power consumption and temporal data as inputs, and the output is the appliance's state probability distribution. The classification subnetwork can be expressed as:

$$\hat{\boldsymbol{y}}_{cls} = f_{cls}(\boldsymbol{x}_{cls}; \theta_{cls}) \tag{2}$$

where $f_{cls}(\cdot\,; \theta_{cls})$ is the classification subnetwork parameterized by $\theta_{cls}$, $\boldsymbol{x}_{cls} \in \mathbb{R}^{l_i \times 2}$ and $\hat{\boldsymbol{y}}_{cls} \in \mathbb{R}^{l_o \times M}$, with $l_i$ is the input sequence length, $l_o$ is the output sequence length and $M$ is the number of appliance states. In our experiments, we set $l_i$ = 432 and $l_o$ = 32.

As for the temporal data, the original format in the dataset is timestamps. Considering the periodicity and continuity of time, i.e., the 24th hour of the first day corresponds to 0th hour of the next day, we use the sine function to represent the time-domain information. Specifically, the time-dimension input can be expressed as:

$$\tau = \sin(\frac{2\pi(h \cdot 3600 + m \cdot 60 + s)}{3600 \cdot 24}) \tag{3}$$

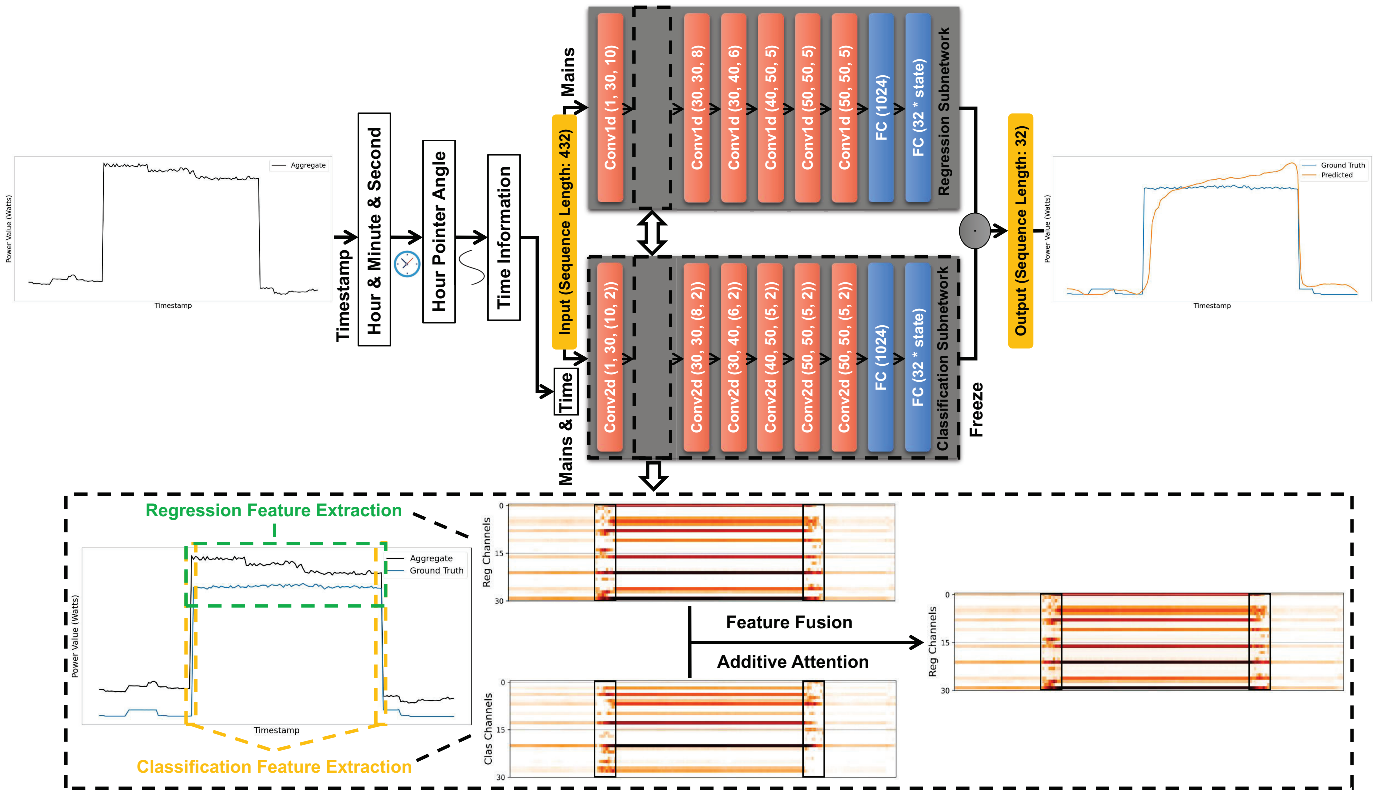

Fig. 1. TimeNILM model architecture.

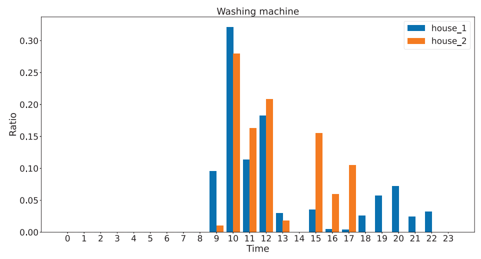

Fig. 2. The washing machine usage in different houses. The horizontal axis represents hours in a day, and the vertical axis indicates the ratio of ON states within the corresponding hour.

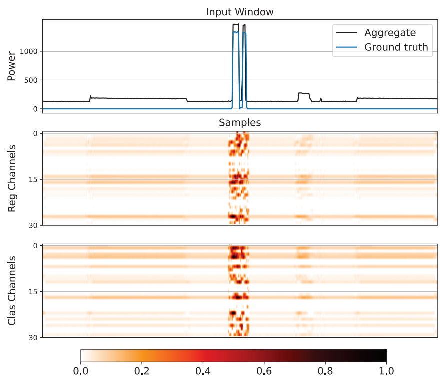

Fig. 3. The feature maps generated by the convolutional networks. The topmost section represents the input and output, while the middle section displays the feature maps in the regression subnetwork and the bottom section depicts the feature maps in the classification subnetwork. The colorbar accompanying the feature maps illustrates the weight of the features, with lighter colors representing lower weights and darker colors representing higher weights.

where $h$, $m$, and $s$ represent the hour, minute, and second, respectively. We are actually transforming the timestamps into the angle of hour hand in the clocks, which are quite representative. Therefore, the input to the classification subnetwork is a two-tuple item, the temporal data and the aggregated power.

## C. The regression subnetwork

We use the aggregated power consumption as the input for the regression subnetwork, and the output is the time-series of the appliance's power consumption. The regression subnetwork can be expressed as:

$$\hat{\boldsymbol{y}}_{reg} = f_{reg}(\boldsymbol{x}_{reg}; \theta_{reg}) \tag{4}$$

where $f_{reg}(\cdot; \theta_{reg})$ is the regression subnetwork parameterized by $\theta_{reg}$, $\boldsymbol{x}_{reg} \in \mathbb{R}^{l_i}$ and $\hat{\boldsymbol{y}}_{reg} \in \mathbb{R}^{l_o \times M}$.

The regression subnetwork employs multiple layers of convolution to extract features from aggregated power consumption, while the fully connected layers integrate these extracted features and map them to the output.

## D. Feature fusion and retraining

Figure 3 shows the feature maps extracted by the last convolutional layer of the input data, including the feature maps of the regression subnetwork and the classification subnetwork. For each data sample, there are 50 corresponding feature weights, where brighter colors indicate larger weights. We observe that when there is a significant change in power levels, some features in both the regression subnetwork and the classification subnetwork have larger weights. This suggests that both subnetworks focus on the features of transition points, indicating a certain degree of similarity in their attention patterns, while the features in the classification subnetwork are more obvious than the other. If we can leverage this information, it can be helpful for power level predictions. As we mentioned above, to avoid the regression subnetwork overfitting to the source domain data, the regression subnetwork does not take the time-domain information as model input. In order for the regression subnetwork to better identify key moments (when appliance switches states), we perform feature fusion between two subnetworks.

We divide the model training into two stages: training stage and retraining stage. In the training stage, we do not perform feature fusion because the regression and classification subnetworks both have weak feature extraction capabilities due to a limited number of training iterations. After training for at most 100 epochs with the early stopping strategy, we proceed with the retraining stage. During this stage, we freeze the parameters of classification subnetwork and retrain regression subnetwork by integrating the data features extracted by the classification subnetwork into regression subnetwork through feature fusion based on attention mechanism. The objective of attention mechanism is to identify features that are similar between both subnetworks and focus attention on these features. Specifically, we consider the features of regression subnetwork as query vectors and the features of classification subnetwork as key vectors, then employ the additive attention mechanism to calculate attention weight $\boldsymbol{\alpha}$ and apply it to the features of classification subnetwork, which can be expressed as:

$$\boldsymbol{\alpha} = \boldsymbol{W_v}^\top(\boldsymbol{W_q f_r} + \boldsymbol{W_k f_c}) \tag{5}$$

$$\boldsymbol{f_r} = \boldsymbol{\alpha} \cdot \boldsymbol{f_c} + \boldsymbol{f_r} \tag{6}$$

where $\boldsymbol{W_v}$, $\boldsymbol{W_q}$ and $\boldsymbol{W_k}$ are learnable parameters, $\boldsymbol{f_c}$ is the features of classification subnetwork and $\boldsymbol{f_r}$ is the features of regression subnetwork. The purpose is to enhance the feature extraction capability of the regression subnetwork and improve the prediction performance.

## E. Prediction results

Assume that the appliance has $M$ states. At each time step, the output from the classification subnetwork is the probability distribution over $M$ states, i.e., $(s_1, s_2, ..., s_m)$; the output from the regression subnetwork is the $M$ power levels corresponding to the $M$ states, i.e., $(p_1, p_2, ..., p_m)$. Then, the predicted power of the appliance is the sum of the element-wise multiplication of the power and the corresponding probability, which can be expressed as the mathematical expectation:

$$\hat{\boldsymbol{y}} = \sum_{m=1}^{M} s_m \cdot p_m. \tag{7}$$

## F. Loss function

We use separate loss functions for the training stage and the retraining stage. In the training stage, we use the following loss function for joint optimization:

$$L_1 = L_{power} + L_{state}. \tag{8}$$

This loss function represents the sum of the power prediction loss and the state prediction loss. For the power prediction loss, we employ mean squared error (MSE) loss to compare the difference between the final predicted value and the ground truth. Assuming there are $n$ training data, we use $y_i$ and $\hat{y}_i$ to represent the true value and the predicted value, respectively. The power prediction loss incurred by the model over the $n$ training data can be defined as follows:

$$L_{power} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{9}$$

For the state prediction loss, we use cross-entropy loss to compare the difference between the predicted states and the actual states, which can be expressed as:

$$L_{state} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j}), \tag{10}$$

where $y_{i,j}$ denotes whether the ground truth of the $i$-th sample is in the $j$-th state, and $p_{i,j}$ denotes the probability that the model predicts the $i$-th sample to belong to the $j$-th state.

In the retraining stage, since we have frozen the parameters of the classification subnetwork, we only optimize the power prediction loss. Similar to the training stage, we employ MSE loss to decrease the difference between the final predicted value and the ground truth. Therefore, the loss function is defined as:

$$L_2 = L_{power}. \tag{11}$$

As we can see from Figure 2, although the active pattern of washing machines is similar between 10:00am and 12:00am, there are dissimilarities between 15:00 and 22:00 across the two households. To improve the model generalization ability, we apply KLDivLoss to both the classification subnetwork and the regression subnetwork. KLDivLoss can be expressed as:

$$L_{KLDiv}(y_{pred}, y_{true}) = y_{true} \cdot (\log y_{true} - \log y_{pred}), \tag{12}$$

where $y_{pred}$ is the input and $y_{true}$ is the target. Specifically, we perform training on the source domain $D_s$ (labeled dataset) and extract its features from the classification subnetwork ($d_s^c$) and the regression subnetwork ($d_s^r$). Simultaneously, we extract the features of the target domain $D_t$ (unlabeled dataset) using the classification subnetwork ($d_t^c$) and the regression subnetwork ($d_t^r$). Subsequently, we use KLDivLoss to minimize the distances between $d_s^c$ and $d_t^c$, as well as between $d_s^r$ and $d_t^r$, which can be expressed as

$$L_{transfer} = L_{KLDiv}(d_s^c, d_t^c) + L_{KLDiv}(d_s^r, d_t^r). \tag{13}$$

Therefore, in the scenario of unsupervised transfer learning, the loss function is defined as:

$$L_3 = L_{power} + L_{state} + L_{transfer}. \tag{14}$$

In summary, in TimeNILM, we enhance the classification subnetwork's state prediction by incorporating temporal domain information. After training both subnetworks for multiple epochs, we retrain the regression subnetwork using feature fusion based on attention mechanism to incorporate information from the classification subnetwork. Finally, we utilize KLDivLoss to align data representations from different domains, leading to improvement of model performance on unseen households.

## *G. Fed-TimeNILM: Extending TimeNILM to federated settings*

As the appliance active patterns and the users' usage habits are quite private and sensitive, users are reluctant to share their household power consumption data. Thus, we extend TimeNILM to federated settings. Sharing knowledge of the classification subnetwork might lead to performance improvements since the usage patterns and state transitions from the same appliance category could be similar across different households. However, sharing knowledge of the regression subnetwork may negatively impact performance, as the same type of appliance from different manufacturers/brands/models can demonstrate inconsistent power levels. Thus, Fed-TimeNILM only performs knowledge sharing among classification subnetworks.

The conventional federated aggregation methods present the challenge that they disrupt the original connections between the two subnetworks during the knowledge sharing process among clients. This challenge arises because the parameters of the regression subnetwork are frozen throughout the process, while the parameters of the classification subnetwork are updated. Consequently, this leads to a deviation in the collaborative outcomes of the dual subnetworks.

To address the challenge, we assume that there is an available dataset on the server (which could be open datasets like REDD or UK-DALE). We aim to dynamically maintain the connectivity of the two subnetworks by updating the classification subnetworks in a federated manner while use the server-side dataset to update the regression subnetwork. Specifically, we employ cosine similarity loss to measure the distance of classification subnetworks, it can be expressed as:

$$L_{similarity} = 1 - \frac{A \cdot B}{max(\|A\|_2 \cdot \|B\|_2 , \epsilon)} \tag{15}$$

where A and B are the classification subnetwork outputs of different clients, enabling the sharing of client classification subnetwork knowledge. Therefore, the loss function in federated settings is defined as:

$$L = \sum_{i}^{N} L^i_{power} + \sum_{i}^{N} L^i_{state} + L_{similarity}, \tag{16}$$

where $L^i_{power}$ and $L^i_{power}$ are the power prediction loss and state prediction loss of client $i$, respectively.

## V. Experiments

### *A. Datasets*

We have evaluated the proposed method on two real-world datasets, namely REDD [29] and UK-DALE [30]. The REDD dataset consists of the aggregated power consumption and individual appliance power consumption data from six households in the United States. The power consumption data for mains and appliances are recorded at intervals of 1 second and 3 seconds, respectively. The UK-DALE dataset comprises the aggregated power consumption and individual appliance power consumption data from five households in the United Kingdom, with power consumption data for mains and appliances recorded at 6-second intervals. We used the interpolation method of backfill to fill in missing values in each sequence. Considering the scenario of transfer learning in real-world setting, we have trained and tested on separate households. Moreover, we also included households from different regions in the training set. Specifically, we used house 1 from the UK-DALE dataset, house 2 and house 3 from the REDD dataset as the training set. House 2 from the UK-DALE dataset and house 1 from the REDD dataset were used as the test set. This approach aims to enable the model to learn useful knowledge even from datasets in different regions and achieve good prediction performance in a completely new household. As the households come from different regions worldwide, we adjust the timestamp information of the REDD dataset to align it with UK-DALE dataset. Considering the appliances commonly used in multiple households, we focused on four main appliances: dishwasher, washing machine, microwave, and fridge, which are consistent with previous works.

### *B. Data preprocessing*

For the REDD and UK-DALE datasets, we performed $z$-score normalization on the power readings. Let $o^i = (o^i_1, o^i_2, ..., o^i_t)$ represent the state sequence of appliance $i$. To make things simple, we assume that each appliance has three states ($o^i_t \in \{0, 1, 2\}$), which is consistent with our observation of the data sets:

$$o^i_t = \begin{cases} 0 & x^i_t < \gamma_0 \\ 1 & \gamma_0 < x^i_t < \gamma_1 \\ 2 & x^i_t > \gamma_1 \end{cases} \tag{17}$$

where $\gamma_0$ is the ON/OFF state threshold value and $\gamma_1$ is the state-switching threshold value. The value of $\gamma_1$ is appliance-dependent and determined by the state divisions identified during K-Means clustering. For example, when microwave power clusters yield intervals (15, 800) and (1000, 2000), we set $\gamma_1 = 900$ - the median between adjacent cluster boundaries.

As for the labels of the classification subnetwork (i.e., appliance states), we distinguish between continuously-on appliances and non-continuously powered-on appliances. As for the non-continuously powered-on appliances (e.g., dishwasher, washing machine, and microwave), we first set the $\gamma_0$ as 15 watts to separate the ON/OFF states of appliances. Then, we employed the K-Means clustering algorithm [31] to further divide the ON state of the appliance into different active levels. As for continuously powered-on appliances (e.g., fridge), it can skip the first step.

### *C. Baselines*

We compare our model with three baselines:

TABLE II
RESULTS FOR THE MEAN ABSOLUTE ERROR (MAE) (WATT) AND THE SIGNAL AGGREGATED ERROR OVER A TIME PERIOD OF $\delta$ (SAE$_\delta$) IN REDD DATASET. BEST RESULTS ARE SHOWN IN BOLD.

| Model | Dishwasher | Washing machine | Microwave | Fridge | Average | Improvement |
|---|---|---|---|---|---|---|
| **S2P** | 18.65/**14.94** | 36.24/33.34 | 18.46/12.69 | 39.05/24.05 | 28.10/21.26 | -/- |
| **SGN** | 19.02/18.84 | 12.90/11.66 | 15.92/15.16 | **35.99**/23.75 | 20.96/17.35 | -/- |
| **MSDC** | 19.43/19.32 | 13.12/11.61 | 13.42/12.81 | 37.60/**23.21** | 20.89/16.74 | 0.00%/0.00% |
| **TimeNILM** | 19.65/19.54 | 7.84/**6.12** | 13.03/12.05 | 37.30/24.60 | 19.46/15.58 | 6.85%/6.93% |
| **TimeNILM-FF** | 19.44/19.33 | **7.61**/6.15 | 13.23/12.17 | 36.53/23.89 | 19.20/15.39 | 8.09%/8.06% |
| **TimeNILM-FF-UTL** | **18.61**/18.44 | 8.64/6.97 | **11.85/11.03** | 37.03/25.11 | **19.03/15.39** | **8.90%/8.06%** |

TABLE III
RESULTS FOR THE MEAN ABSOLUTE ERROR (MAE) (WATT) AND THE SIGNAL AGGREGATED ERROR OVER A TIME PERIOD OF $\delta$ (SAE$_\delta$) IN UK-DALE DATASET. BEST RESULTS ARE SHOWN IN BOLD.

| Model | Dishwasher | Washing machine | Microwave | Fridge | Average | Improvement |
|---|---|---|---|---|---|---|
| **S2P** | 34.36/27.89 | 17.80/16.75 | 15.73/11.45 | 28.34/20.38 | 24.06/19.12 | -/- |
| **SGN** | 17.59/14.31 | 17.39/16.72 | 8.83/6.66 | 14.09/8.94 | 14.48/11.66 | -/- |
| **MSDC** | 16.12/7.98 | 16.41/15.54 | 8.86/8.06 | 12.83/7.54 | 13.56/9.78 | 0.00%/0.00% |
| **TimeNILM** | 14.12/6.58 | 8.28/6.68 | 8.74/7.94 | 13.88/8.32 | 11.26/7.38 | 16.96%/24.54% |
| **TimeNILM-FF** | 13.44/7.34 | **7.51/5.79** | **8.70/7.91** | 13.53/7.94 | 10.80/7.25 | 20.35%/25.87% |
| **TimeNILM-FF-UTL** | **11.76/5.61** | 7.84/6.02 | 9.22/7.58 | **10.61/6.01** | **9.86/6.31** | **27.29%/35.48%** |

- S2P [9]. S2P is a sequence-to-point model, where the input is a window of the mains data, and the output is a single point corresponding to the power consumption level of the target appliance.
- SGN [5]. SGN introduces a dual-CNN architecture, with a subnetwork for ON/OFF state classification and a subnetwork for power level regression. Evaluation results show that such a dual-CNN model fits the NILM problem well.
- MSDC [8]. MSDC can be considered as the state-of-the-art model. Instead of binary ON/OFF states in SGN, the classification subnetwork in MSDC classifies the appliance into multiple active levels. MSDC has demonstrated superior performance than RNN and transformer-based models, and thereby we have not compared TimeNILM with them.

## D. Training details and Evaluation metrics

The convolutional neural network was implemented in Python with PyTorch 1.13.0 and trained on an NVIDIA GeForce RTX 4090. Our model is trained with Adam optimizer at a learning rate of $10^{-4}$. The early stopping strategy is employed during the training process to prevent overfitting, with the patience set to 5. The input data length is 432 and the output data length is 32. We use the sliding window method to process data, with a stride length of 200.

The mean absolute error (MAE) and the signal aggregated error over a time period of $\delta$ (SAE$_\delta$) were used as evaluation metrics. MAE is a commonly used measure for regression problems, which quantifies the average absolute difference between the predicted and actual power consumption of an appliance. For appliance $i$, it can be expressed as:

$$MAE^i = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t^i - y_t^i| \tag{18}$$

where $T$ is the length of the predicted output sequence, $\hat{y}_t^i$ is the predicted value and $y_t^i$ is the ground truth.

SAE$_\delta$ represents the average overall error within a time period $\delta$, taking the difference between the sum of aggregated predicted value and the ground truth within the period. It mainly focuses on the cumulative power consumption rather than the instantaneous power consumption. For appliance $i$, SAE$_\delta$ can be expressed as:

$$SAE_\delta^i = \frac{1}{T_\delta}\sum_{n=1}^{N_\delta}\frac{1}{N_\delta}|c_n^i - r_n^i| \tag{19}$$

where $T_\delta$ is the duration of time periods, $N_\delta$ is the number of data points in the time period $\delta$, $c_n^i$ is the sum of predicted value in the $n$-th period and $r_n^i$ is the sum of the groundtruth value in the period:

$$c_n^i = \sum_{t=1}^{N_\delta}\hat{y}_{N_\delta\cdot n+t}^i, \quad r_n^i = \sum_{t=1}^{N_\delta} y_{N_\delta\cdot n+t}^i \tag{20}$$

In our experiments, we set $\delta$ = 1 hour and $N_\delta$ = 450.

## E. Performance

Table II and III show the comparison results of our methods and baselines on the REDD and UK-DALE test sets. In our methods, TimeNILM is trained by incorporating the temporal dimension, TimeNILM-FF retrains the TimeNILM with feature fusion, and TimeNILM-FF-UTL is trained with the additional KLDivLoss based on TimeNILM-FF. Independent comparisons have been performed among these methods. In both tables, bold font indicates the best-performing algorithm for each appliance. Compared to the baseline, TimeNILM achieved an average improvement of 8% in MAE and 8% in SAE$_\delta$ on the REDD test set, and an average improvement of 27% in MAE and 35% in SAE$_\delta$ on the UK-DALE test set. This indicates that by incorporating the temporal dimension, even data from different regions can be used to train a model that performs well in unsupervised transfer learning scenario. Furthermore, with the assistance of feature fusion, the features from the classification subnetwork can enhance the feature

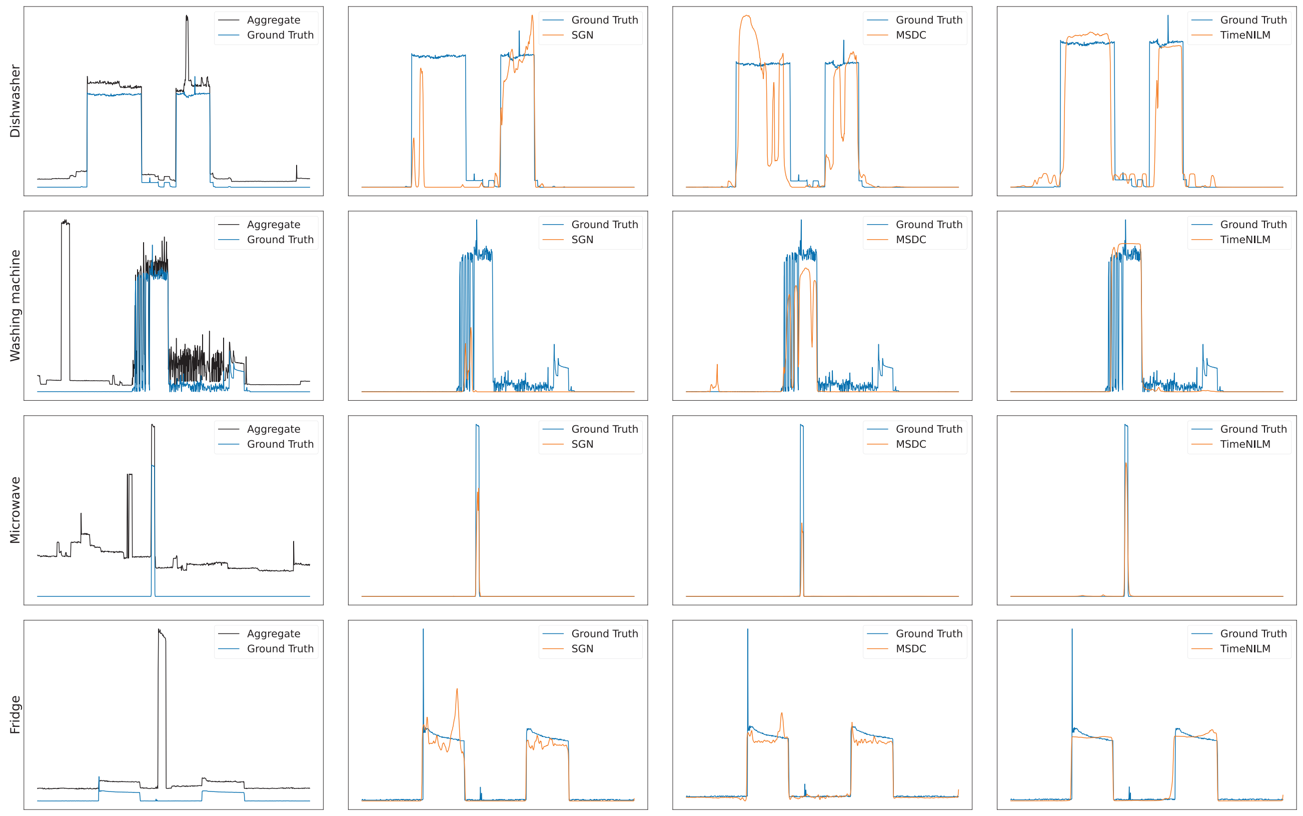

Fig. 4. Some snippets of power prediction for four appliances in the UK-DALE dataset.

extraction capability of the regression subnetwork, thereby improving prediction performance.

As shown in Figure 4, we present the visualized results of power prediction for four appliances in the UK-DALE dataset by different models within one time cycle. The leftmost column represents the mains input and the ground truth of each appliance. The subsequent columns show the differences between the prediction and the ground truth for SGN, MSDC, and TimeNILM models. Among these models, TimeNILM demonstrates the smallest prediction error. From the observation in the figure, we notice that TimeNILM model outperforms SGN and MSDC models in these appliances, thereby proving its superiority in practical applications.

Furthermore, Table IV benchmarks computational complexity across all baselines and TimeNILM. Unlike dual-CNN architectures (SGN, MSDC, TimeNILM), S2P employs a single regression network, reducing its complexity to 50% of dual-subnetwork models. Among dual-CNN implementations, TimeNILM matches SGN's MFLOPS while significantly outperforming it in prediction accuracy, demonstrating an optimal efficiency-accuracy trade-off.

TABLE IV
THE COMPARISON OF COMPUTATIONAL COMPLEXITY

| Model | Computational Complexity (MFLOPS) |
|---|---|
| **S2P** | 35.82 |
| **SGN** | 81.29 |
| **MSDC** | 97.59 |
| **TimeNILM** | 81.42 |

## *F. Ablation Experiment*

In this subsection, we perform an ablation study. Since unsupervised transfer based on the KLDivLoss is a well-known technique, it has the potential to significantly enhance the model performance. Therefore, we conduct the ablation experiments considering KLDivLoss independently. We use MSDC as the baseline in our ablation experiment, while MSDC-UTL is trained with the additional KLDivLoss based on MSDC. In Table V, we can observe that using the KLDivLoss alone results in a performance decline. However, when the model, which incorporates both the temporal dimension and feature fusion, is further optimized with the KLDivLoss, there is a substantial improvement. This indirectly demonstrates the advantages of the temporal dimension in unsupervised transfer scenarios.

## *G. Fed-TimeNILM*

The performance with federated learning is shown in Table VI, where TimeNILM-FF-UTL is the initial model, TimeNILM-FF-UTL-FA is trained with the classical FedAvg algorithm based on TimeNILM-FF-UTL, and TimeNILM-FF-UTL-FL is trained with our proposed method based on TimeNILM-FF-UTL. Compared to the initial model, TimeNILM-FF-UTL-FL achieves 2%-9% MAE gain and 2%-14% SAE$_\delta$ gain. Therefore, the federated strategy of joint learning and aggregation offers a novel solution for knowledge sharing in federated learning. It not only enhances the overall performance of the model but also fosters collaboration and

TABLE V
RESULTS FOR ABLATION EXPERIMENTS IN REDD (TOP) AND UK-DALE (BOTTOM) DATASETS. USING KL-DIVERGENCE ALONE CANNOT BRING PERFORMANCE GAIN.

| Model | Dishwasher | Washing machine | Microwave | Fridge | Average |
|---|---|---|---|---|---|
| **MSDC** | **19.43/19.32** | **13.12/11.61** | **13.42/12.81** | **37.60**/23.21 | **20.89/16.74** |
| **MSDC-UTL** | 23.61/23.19 | 19.20/13.54 | 14.62/13.25 | 45.35/**18.33** | 25.70/17.08 |

| Model | Dishwasher | Washing machine | Microwave | Fridge | Average |
|---|---|---|---|---|---|
| **MSDC** | **16.12/7.98** | **16.41/15.54** | **8.86/8.06** | **12.83**/7.54 | **13.56/9.78** |
| **MSDC-UTL** | 22.21/11.51 | 25.87/25.04 | 13.07/11.15 | 20.55/**6.48** | 20.43/13.55 |

TABLE VI
RESULTS FOR FED-TIMENILM ON REDD (TOP) AND UK-DALE (BOTTOM) DATASETS. WE PRESENT THE PERFORMANCE RESULTS FOR DIFFERENT HOUSEHOLDS. BEST RESULTS ARE SHOWN IN BOLD.

| Model | Dishwasher | Washing machine | Microwave | Fridge | Average |
|---|---|---|---|---|---|
| **TimeNILM-FF-UTL (REDD_1)** | 8.96/8.11 | 7.47/6.32 | 6.80/6.13 | 33.96/24.94 | 14.30/11.38 |
| **TimeNILM-FF-UTL-FA (REDD_1)** | 13.01/11.87 | 5.86/4.36 | 5.83/5.48 | 41.93/26.43 | 16.66/12.04 |
| **TimeNILM-FF-UTL-FL (REDD_1)** | 11.27/10.58 | 3.91/2.97 | 6.62/5.88 | 32.78/23.10 | 13.65/10.63 |
| **TimeNILM-FF-UTL (REDD_2)** | 33.55/33.67 | 20.33/18.88 | 18.74/17.34 | 39.58/26.62 | 28.05/24.13 |
| **TimeNILM-FF-UTL-FA (REDD_2)** | 31.28/31.46 | 30.26/23.77 | 19.65/18.58 | 39.08/21.09 | 30.07/23.73 |
| **TimeNILM-FF-UTL-FL (REDD_2)** | 33.99/34.12 | 13.92/12.83 | 18.90/17.90 | 41.39/29.66 | 27.05/23.63 |

| Model | Dishwasher | Washing machine | Microwave | Fridge | Average |
|---|---|---|---|---|---|
| **TimeNILM-FF-UTL (UK-DALE_1)** | 16.00/13.57 | 8.84/6.66 | 9.59/7.29 | 9.74/6.20 | 11.04/8.43 |
| **TimeNILM-FF-UTL-FA (UK-DALE_1)** | 37.05/26.79 | 16.92/16.51 | 10.34/9.12 | 15.66/11.39 | 19.99/15.95 |
| **TimeNILM-FF-UTL-FL (UK-DALE_1)** | 16.33/10.18 | 5.99/4.23 | 8.49/7.34 | 12.37/8.47 | 10.80/7.56 |
| **TimeNILM-FF-UTL (UK-DALE_2)** | 24.08/21.63 | 8.28/5.29 | 8.90/7.93 | 10.81/6.41 | 13.02/10.32 |
| **TimeNILM-FF-UTL-FA (UK-DALE_2)** | 39.94/28.62 | 11.46/7.83 | 12.32/11.37 | 16.07/8.05 | 19.95/13.97 |
| **TimeNILM-FF-UTL-FL (UK-DALE_2)** | 15.86/13.99 | 8.99/5.37 | 8.83/8.19 | 13.59/7.83 | 11.82/8.85 |

TABLE VII
PERFORMANCE OF DIFFERENT QUANTIZATION SCHEMES

| Scheme | Model size (MB) | Inference time (s) | MAE/SAE | $\Delta$MAE/$\Delta$SAE |
|---|---|---|---|---|
| **Float32** | 254.0 | 0.104 | 15.320/5.882 | 0.000/0.000 |
| **Float16** | 127.0 | 0.110 | 15.325/5.881 | 0.005/0.001 |
| **Int8** | 63.6 | 0.049 | 15.549/6.135 | 0.229/0.253 |

knowledge dissemination among clients with enhanced data privacy.

## H. Deployment

In this subsection, we deploy the TimeNILM model on resource-constrained edge devices. We choose the Raspberry Pi 4 Model B as the hardware platform for deployment. Prior to the actual deployment, we first convert the TimeNILM model into the ONNX format, a cross-platform model representation format. However, since the ONNX Runtime does not support Float16 as the data type [32], we further convert the ONNX format model into the tflite format. Subsequently, we deploy the tflite format model on the Raspberry Pi 4 Model B for real-time, low-latency inference.

In our experiments, we considered various quantization schemes. The objective was to comprehensively understand the impact of different quantization schemes on overall performance, including model size, inference time, and performance accuracy. By selecting appropriate quantization schemes, we aimed to optimize the model to better align with the hardware characteristics while maintaining model accuracy.

As shown in Table VII, we observe that, with the gradual decrease in quantization precision from Float32 to Float16 and then to Int8, the size of the model significantly decreases. However, correspondingly, the model's accuracy drop (including $\Delta$MAE and $\Delta$SAE) in prediction tasks gradually increases. This indicates a trade-off relationship between model size and performance accuracy. Further analysis reveals that the performance accuracy drop of Float16 is relatively small, but when the quantization precision is set to Int8, the error increases significantly. This may be attributed to the inability of integer representation to fully capture minor variations in model parameters, leading to a decrease in the model's sensitivity to input data.

In addition, regarding the inference time for each sample, we observe a significant decrease when the quantization precision transitions from Float32 to Int8. However, the inference time for Float16 slightly increases compared to Float32. This could be attributed to the fact that, when running the Float16 quantized model on the CPU of Raspberry Pi 4 Model B, the weight values are dequantized to Float32, introducing additional computational costs in the process, and thus prolonging the inference time. On the whole, regardless of the quantization scheme employed, the inference times are significantly shorter than the recording intervals. Therefore, these models can support real-time inference.

## VI. CONCLUSION

In this paper, we propose TimeNILM, a time-aware dual-CNN architecture. By leveraging the similarity in appliance active schedules across different households, we propose the integration of the temporal dimension into NILM, aiming to capture the relationship between temporal data and appliance states. We further design the feature fusion strategy based on attention mechanism to enhance the prediction performance of the regression subnetwork. To better adapt to unsupervised transfer learning scenarios, we utilize KLDivLoss to minimize the discrepancy between the features of existing household data and unknown households. We combine the data from different houses in the REDD and UK-DALE datasets as the training set, while using unseen houses from two regions as separate test sets. Evaluation results show that we can achieve 8% to 27% MAE improvement and 8% to 35% $\mathrm{SAE}_\delta$ improvement over state-of-the-art deep learning solutions. These results demonstrate that the usage of temporal dimension and the utilization of feature fusion provide promising directions for enhancing the prediction performance of dual-CNN NILM. Through extending TimeNILM to federated settings and employing a federated strategy of joint learning and aggregation, we enhance NILM performance while protecting the data privacy. Furthermore, we have successfully implemented the TimeNILM model on a resource-limited edge device, demonstrating that the model can perform real-time load monitoring on the Raspberry Pi, thereby providing a viable solution for real-world applications.

## REFERENCES

[1] D. Chioran and H. Valean, "Design and performance evaluation of a home energy management system for power saving," *Energies*, vol. 14, no. 6, p. 1668, Mar 2021. [Online]. Available: http://dx.doi.org/10.3390/en14061668

[2] O. Ben-Salha, B. Hkiri, and C. Aloui, "Sectoral energy consumption by source and output in the us: New evidence from wavelet-based approach," *Energy Economics*, vol. 72, pp. 75–96, 2018.

[3] G.-F. Angelis, C. Timplalexis, S. Krinidis, D. Ioannidis, and D. Tzovaras, "Nilm applications: Literature review of learning approaches, recent developments and challenges," *Energy and Buildings*, vol. 261, p. 111951, 2022.

[4] M. Kaselimi, E. Protopapadakis, A. Voulodimos, N. Doulamis, and A. Doulamis, "Towards trustworthy energy disaggregation: A review of challenges, methods, and perspectives for non-intrusive load monitoring," *Sensors*, vol. 22, no. 15, p. 5872, 2022.

[5] C. Shin, S. Joo, J. Yim, H. Lee, T. Moon, and W. Rhee, "Subtask gated networks for non-intrusive load monitoring," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1150–1157.

[6] Y. Mao, K. Dong, and J. Zhao, "Non-intrusive load decomposition technology based on crf model," in *2021 IEEE Sustainable Power and Energy Conference (iSPEC)*, Dec 2021. [Online]. Available: http://dx.doi.org/10.1109/ispec53008.2021.9735837

[7] S. Mari, G. Bucci, F. Ciancetta, E. Fiorucci, and A. Fioravanti, "Advanced architecture for training and testing nilm systems," in *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, 2022, pp. 1–6.

[8] J. He, J. Liu, Z. Zhang, Y. Chen, Y. Liu, B. Khoussainov, and L. Zhu, "Msdc: exploiting multi-state power consumption in non-intrusive load monitoring based on a dual-cnn model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, pp. 5078–5086.

[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 *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[10] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[11] R. Zeng, C. Zeng, X. Wang, B. Li, and X. Chu, "Incentive mechanisms in federated learning and a game-theoretical approach," *IEEE Network*, vol. 36, no. 6, pp. 229–235, 2022.

[12] Y. Zhang, G. Tang, Q. Huang, Y. Wang, K. Wu, K. Yu, and X. Shao, "Fednilm: Applying federated learning to nilm applications at the edge," *IEEE Transactions on Green Communications and Networking*, 2022.

[13] H. Pötter, S. Lee, and D. Mossé, "Towards privacy-preserving framework for non-intrusive load monitoring," in *Proceedings of the Twelfth ACM international conference on future energy systems*, 2021, pp. 259–263.

[14] J. Feng, K. Li, H. Zhang, X. Zhang, and Y. Yao, "Multichannel spatio-temporal feature fusion method for nilm," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8735–8744, 2022.

[15] S. Sykiotis, M. Kaselimi, A. Doulamis, and N. Doulamis, "Electricity: An efficient transformer for non-intrusive load monitoring," *Sensors*, vol. 22, no. 8, p. 2926, 2022.

[16] D. Kim and J. Choi, "Unsupervised representation learning for binary networks by joint classifier learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 9747–9756.

[17] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[18] S. Makonin, F. Popowich, I. V. Bajic, B. Gill, and L. Bartram, "Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring," *IEEE Transactions on Smart Grid*, vol. 7, no. 6, p. 2575–2585, Nov 2016. [Online]. Available: http://dx.doi.org/10.1109/tsg.2015.2494592

[19] M. Aiad and P. H. Lee, "Non-intrusive load disaggregation with adaptive estimations of devices main power effects and two-way interactions," *Energy and Buildings*, p. 131–139, Oct 2016. [Online]. Available: http://dx.doi.org/10.1016/j.enbuild.2016.08.050

[20] R. Bonfigli, E. Principi, M. Fagiani, M. Severini, S. Squartini, and F. Piazza, "Non-intrusive load monitoring by using active and reactive power in additive factorial hidden markov models," *Applied Energy*, vol. 208, pp. 1590–1607, 2017.

[21] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, no. C, may 2021. [Online]. Available: https://doi.org/10.1016/j.cosrev.2021.100379

[22] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys*, p. 1–36, Sep 2019. [Online]. Available: http://dx.doi.org/10.1145/3234150

[23] G. Bejarano, D. DeFazio, and A. Ramesh, "Deep latent generative models for energy disaggregation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, p. 850–857, Sep 2019. [Online]. Available: http://dx.doi.org/10.1609/aaai.v33i01.3301850

[24] X. Bai, B. Shi, C. Zhang, X. Cai, and L. Qi, "Text/non-text image classification in the wild with convolutional neural networks," *Pattern Recognition*, vol. 66, pp. 437–446, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *Computer Research Repository*, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[28] J. Barber, H. Cuayáhuitl, M. Zhong, and W. Luan, "Lightweight non-intrusive load monitoring employing pruned sequence-to-point learning," in *Proceedings of the 5th international workshop on non-intrusive load monitoring*, 2020, pp. 11–15.

[29] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA*, vol. 25, 2011, pp. 59–62.

[30] J. Kelly and W. Knottenbelt, "The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes," *Scientific data*, vol. 2, no. 1, pp. 1–14, 2015.

[31] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[32] C. Dong, T. Z. Li, K. Xu, Z. Wang, F. Maldonado, K. Sandler, B. A. Landman, and Y. Huo, "Characterizing browser-based medical imaging ai with serverless edge computing: towards addressing clinical data security constraints," in *Proceedings of SPIE–the International Society for Optical Engineering*, vol. 12469. NIH Public Access, 2023.