A Generalizable Load Recognition Method in NILM Based on Transferable Random Forest

Zhongzong Yan, Pengfei Hao, Matteo Nardello, *Member, IEEE*, Davide Brunelli, *Senior Member, IEEE*, and He Wen, *Senior Member, IEEE*

***Abstract*—Practical applications of nonintrusive load monitoring (NILM) require load recognition models that generalize to unseen data from new houses and operate efficiently on edge devices. However, existing NILM approaches, particularly deep learning (DL) models, are computationally intensive and prone to performance degradation when deployed to new houses due to domain shifts. To address these challenges, this article proposes a weighted transferable random forest (WTRF) approach for load recognition. Based on the random forest (RF) framework, WTRF incorporates a transfer learning (TL) mechanism to swiftly adapt the model to new houses using only one to three labeled samples per appliance. The model is lightweight, with a memory size under 300 kB. Case studies on three datasets demonstrate its effectiveness, including a macro $F1$-score of 97.0% ± 2.6% when transferring from PLAID to WHITED, a significant improvement over 5.7% ± 1.7% achieved by source-only models. Deployed on a Raspberry Pi 4, WTRF achieves update times as low as 3.1 ± 0.3 s and testing times of approximately 3 ms per house. These results highlight WTRF's efficiency in addressing domain shifts and its suitability for real-time NILM in resource-constrained edge environments.**

***Index Terms*—Load recognition, nonintrusive load monitoring (NILM), random forest (RF), transfer learning (TL).**

I. Introduction

UNDERSTANDING energy usage is essential for optimizing energy management and ultimately promoting more efficient energy consumption [1]. Smart grids enhance energy strategies by utilizing smart meters to collect data and provide real-time feedback on users' energy consumption. Appliance-level feedback allows users to monitor individual devices and make timely adjustments, significantly improving energy efficiency [2]. Pilot studies have reported that providing appliance-level feedback can reduce electricity consumption by 15% to 40% across dozens of monitored devices [3].

To achieve these goals, a straightforward approach to monitoring appliance energy usage is to install a measurement sensor for each device [4]. However, this method is costly and impractical for large-scale deployment due to extensive hardware requirements. As an alternative, a technique known as nonintrusive load monitoring (NILM) was proposed by Har [5]. NILM aims to identify the operational states and energy consumption of individual appliances by analyzing aggregate signals collected from main meters, thereby eliminating the need for per-appliance sensors.

Recent machine learning (ML)- and deep learning (DL)-based NILM methods have been successfully developed, effectively addressing basic load recognition problems when the training and testing data come from the same distribution. However, in practical deployments, the data used to train the model are often collected from households or regions that differ from the testing environment, leading to notable differences in load characteristics and discrepancies in the feature space. As a result, domain shifts can significantly degrade model performance, particularly when the feature distributions of the two domains vary widely. For example, Liu et al. [6] reported that DL models trained on the PLAID dataset with an accuracy of 94.04% ± 1.27% were applied to classify the same 11 types of appliances in the WHITED dataset. As a result, the classification accuracy dropped significantly to 23.82% ± 3.63%. Similarly, the study in [7] reported that a DL-based model achieved a macro $F1$-score of 77.60% on the PLAID dataset and 75.45% on the WHITED dataset when evaluated using leave-one-house-out cross-validation (LoHoCV).[1] These results are notably lower than those achieved with K-fold cross-validation on the same datasets, highlighting the challenges posed by domain shifts. To address these challenges, transfer learning (TL) techniques have been employed to mitigate distribution shifts caused by differences in load characteristics [8].

TL techniques aim to improve model generalization in the target domain by leveraging knowledge from the source domain and related tasks. As outlined in [9], TL methods can be categorized into four groups: instance-, feature-, parameter-, and relational-based approaches. Among these, parameter- and feature-based methods are the most commonly used in NILM applications. Parameter-based fine-tuning (F-T) adjusts the parameters of pretrained models using data from the target domain. For instance, the work in [1] studied the generalization of convolutional neural networks (CNNs) in energy disaggregation and demonstrated that F-T only the fully

Received 4 February 2025; revised 31 March 2025; accepted 27 April 2025. Date of publication 15 May 2025; date of current version 3 June 2025. This work was supported by State Grid Jiangxi Electric Power Company Ltd., through Researching Key Science and Technology Projects under Grant 521852230006. The Associate Editor coordinating the review process was Dr. Yau Chung. *(Corresponding author: He Wen.)*

Zhongzong Yan, Pengfei Hao, and He Wen are with the College of Electrical and Information Engineering, Hunan University, Changsha 410012, China (e-mail: yanzhongzong@hnu.edu.cn; haopengfei0408@126.com; he_wen82@126.com).

Matteo Nardello and Davide Brunelli are with the Department of Industrial Engineering, University of Trento, 38123 Trento, Italy (e-mail: matteo.nardello@unitn.it; davide.brunelli@unitn.it).

Data is available on-line at https://github.com/zhz-yan/TRFNILM

Digital Object Identifier 10.1109/TIM.2025.3570355

[1] LoHoCV: The training and testing sets are partitioned by household. Data from one household are used for testing, while data from all other households are used for training. This process is repeated for each household.

connected (FC) layers is sufficient for effective TL. In [6], the AlexNet pretrained on image datasets was fine-tuned for load recognition by adjusting the parameters of the last FC layer with electric signal data. In addition, [10] investigated the transferability of Transformer-based models and methods to reduce the computational cost of model pretraining. The work in [11] extended F-T to a one-to-many model for estimating the power consumption of multiple appliances simultaneously. It argued that F-T the FC layer alone is insufficient, and adjustments to the CNN layers are required.

Feature-based methods aim to enhance model transferability by learning transformations that extract invariant feature representations across domains. Prominent approaches include the domain adversarial neural network (DANN) [12], the adversarial deep neural network (ADNN) [13], multikernel maximum mean discrepancy (MK-MMD) [14], and deep correlation alignment (Deep CORAL) [15]. In the context of NILM, Lin et al. [16] applied domain adaptation techniques leveraging unlabeled target data to improve model transferability across domains using MMD and CORAL. Similarly, Liu et al. [17] proposed an unsupervised domain adaptation (UDA) approach based on DANN and ADNN, while Hao et al. [18] improved existing UDA methods by separating domain-specific and shared feature representations. Furthermore, Zhong et al. [19] introduced a source-free domain adaptation technique for NILM that eliminates the need for source data and labeled target data. In addition to domain adaptation, several techniques aim to improve the generalization ability of NILM models, including self-supervised learning [20], knowledge distillation [21], and metalearning [22], [23].

Despite recent progress, several challenges persist in NILM: 1) fine-tuned DL-based models require extensive labeled data to capture all possible operational states of appliances in the real world, making the process costly and time-consuming; 2) these models are computationally intensive, which complicates their deployment on resource-constrained edge devices, particularly due to the need for posttraining updates; and 3) although UDA enables cross-domain learning without labeled target data, its success is inconsistent, and the causes of negative transfer, especially under significant domain shifts, remain poorly understood. Given these challenges, improving the generalization of load recognition models and enabling few-shot recognition of appliances in the target domain have emerged as critical research focuses, especially in scenarios where labeled data for retraining are scarce.

To address these issues, we propose a generalized load recognition method using a transferable random forest (RF) framework. The proposed approach utilizes a low-dimensional feature space in combination with a resource-efficient RF algorithm to enhance load recognition accuracy. Our method requires only a small number of labeled samples per appliance from the new environment, enabling rapid adaptation. Moreover, the proposed approach is significantly more lightweight than DL-based models and can be deployed efficiently on edge devices, ensuring effective updates after deployment. The main contributions are summarized as follows.

1) We propose a weighted transferable RF (WTRF) method for load recognition in NILM. The method incorporates a TL technique to adapt trained models to new houses with as few as one or three labeled samples per appliance, greatly reducing the data and computational requirements compared to DL-based methods.
2) The WTRF method is evaluated across three public datasets, demonstrating its effectiveness under domain shifts involving different appliance brands, houses, and datasets. Our results show that WTRF achieves an $F1$-macro improvement of up to 90% over source-only models and outperforms UDA and fine-tuned DL-based models.
3) We deploy the model on a Raspberry Pi 4 to evaluate its suitability for edge computing. The model achieves update times as low as 3.1 ± 0.3 s and testing times of approximately 3 ms per house, enabling real-time adaptation in resource-constrained environments.

The rest of this article is organized as follows. Section II introduces the problem formulation and fundamentals of RF. Section III details the proposed method, followed by the experimental setup in Section IV. Section V discusses the results, and conclusion are provided in Section VI.

II. Preliminaries

A. Notations and Problem Formulation

Let $\mathcal{D} = (\mathcal{X}, \mathcal{Y}, p)$ denote a domain, where $\mathcal{X}$ is the feature space, $\mathcal{Y}$ is the label space, and p is a joint probability distribution over $(\mathcal{X}, \mathcal{Y})$. In TL, we consider two domains: a source domain $\mathcal{D}^S = (\mathcal{X}^S, \mathcal{Y}^S, p^S)$ and a target domain $\mathcal{D}^T = (\mathcal{X}^T, \mathcal{Y}^T, p^T)$. We assume that $\mathcal{X}^S = \mathcal{X}^T$ and $\mathcal{Y}^S = \mathcal{Y}^T$, indicating that both domains contain the same appliance types and electrical features. However, the joint distributions differ: $p^S(x, y) \neq p^T(x, y)$.

A dataset $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is sampled from $\mathcal{D}$, where $\mathbf{x}_i \in \mathbb{R}^M$ represents the features, M is the feature dimension, and $y_i \in \mathcal{Y}$ represents the corresponding label. Here, $\mathcal{Y}$ corresponds to a set of class labels $\{1, 2, \ldots, C\}$. The source domain dataset $X^S = \{(\mathbf{x}_i^S, y_i^S)\}_{i=1}^{n_s}$ contains n_s samples, while the target domain dataset $X^T = \{(\mathbf{x}_j^T, y_j^T)\}_{j=1}^{n_t}$ contains n_t samples. In this study, a small subset $X^u = \{(\mathbf{x}_i^u, y_i^u)\}_{i=1}^{n_u}$, where $X^u \subseteq X^T$, is used for model updates, typically including $1-3$ labeled samples per appliance. We assume that X^S and X^T are sampled independently of the joint distributions $p^S(x, y)$ and $p^T(x, y)$, respectively. We follow the definition provided in [24], where *appliance type* refers to the general category (e.g., fridges, TV), while the term *appliance* or *appliance instance* denotes a specific brand or model within that category.

Prior few-shot NILM studies have shown that even a few samples per appliance instance can yield competitive results, supporting our choice of using one to three samples per appliance. For example, [25] achieved an accuracy of 86.77% with just ten samples per appliance type in the PLAID dataset, [23] reported $F1$-macro scores exceeding 92% using only one to three samples per appliance on the WHITED dataset, and [26] reported an $F1$-macro of 76% using five samples per appliance on the PLAID dataset. These findings confirm the feasibility of our experimental design, where our goal is to

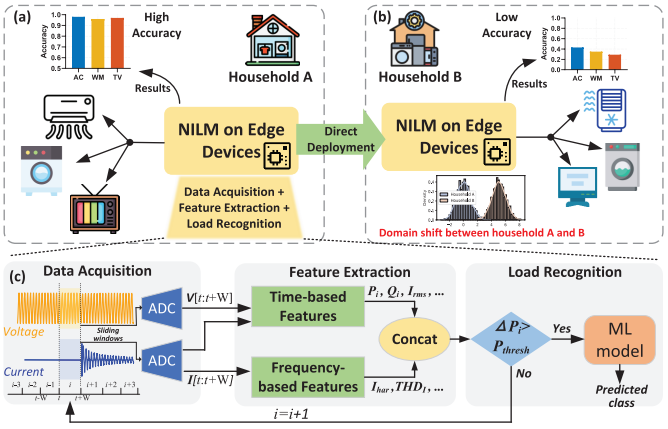

Fig. 1. NILM workflow and challenges under domain shift in edge deployment. (a) NILM models perform well in Household A when training and testing data originate from the same domain. (b) Direct deployment to Household B suffers due to domain shift caused by variations in appliances or usage conditions. (c) General framework of the NILM approach.

demonstrate effective NILM adaptation with minimal data, reflecting realistic practical constraints.

Fig. 1 illustrates an NILM framework designed for deployment on edge devices. As shown in Fig. 1(c), the system consists of three key stages: data acquisition, feature extraction, and load recognition [27]. In the data acquisition stage, voltage and current waveforms are sampled using analog-to-digital converters with a sliding window. Then, the feature extraction stage derives both time- and frequency-based features. If a variation in active power (ΔP_i) exceeds a predefined threshold (P_{thresh}), a switching event is detected, and the system triggers load recognition via an ML model.

When an NILM model is deployed on the same household that it was trained on, it performs well because the training and testing data share the same domain, as illustrated in Fig. 1(a). However, direct deployment to a different household (e.g., Household B) results in poor accuracy due to domain shift, caused by differences in appliance models, brands, or usage patterns, as shown in Fig. 1(b). To address this issue, we propose an RF-based method enhanced with TL. By updating a subset of trees using a small number of labeled samples from the target household, our method adapts efficiently to new domains while keeping labeling costs and computational overhead low.

B. DTs and RFs

Now, we briefly describe decision trees (DTs) based on the classification and regression tree (CART) algorithm and RFs. The introduced notation will be useful for understanding the rest of this article.

Let v denote a node in a tree and X_v denote the subset of the data that reach node v. The CART algorithm starts with the root node v_0. At each node v, a particular subset X_v is processed (for the root, $X_{v_0} = X$). For a binary DT, the two children of a node v are referred to as the left child v_l and the right child v_r. Each internal node v selects a splitting feature $a(v)$ and a numeric threshold $\tau(v)$ to divide X_v into left (X_{v_l}) and right (X_{v_r}) subsets.

For each feature a^i in X_v, where $i \in \{1, 2, \ldots, M\}$, candidate thresholds τ_k^i are computed as the midpoints between sorted unique values of a^i. Here, $|A^i|$ represents the number of unique values of feature a^i in X_v. These candidate thresholds are denoted as follows:

$$\Theta^i = \left\{ \tau_k^i = \left(x_{j_k}^i + x_{j_{k+1}}^i \right) \Big/ 2 \,\middle|\, k = 1, \ldots, |A^i| - 1 \right\}. \quad (1)$$

The Gini impurity of a split is defined as follows:

$$\text{Ginisplit}\left(X_v; \tau_k^i\right) = \frac{\left|X_{v_l}\left(\tau_k^i\right)\right|}{|X_v|} \cdot \text{Gini}\left(X_{v_l}\left(\tau_k^i\right)\right) + \frac{\left|X_{v_r}\left(\tau_k^i\right)\right|}{|X_v|} \cdot \text{Gini}\left(X_{v_r}\left(\tau_k^i\right)\right) \quad (2)$$

where $\text{Gini}(X_v) = 1 - \sum_{c \in \mathcal{Y}} p_c^2(v)$ and $p_c(v)$ is the proportion of samples in X_v belonging to class c. The splitting feature $a(v)$ and threshold $\tau(v)$ are chosen to minimize the Gini impurity of the split

$$a(v), \tau(v) = \arg\min_{a^i, \tau_k^i \in \Theta^i} \text{GiniSplit}\left(X_v; \tau_k^i\right). \quad (3)$$

Once the optimal split is determined, X_v is divided into two subsets: $X_{v_l} = \{x_j \in X_v \mid x_j^{a(v)} \le \tau(v)\}$ and $X_{v_r} = \{x_j \in X_v \mid x_j^{a(v)} > \tau(v)\}$. Finally, A node becomes a leaf when all samples belong to the same class or the sample count falls below a threshold. The leaf predicts the majority class in X_v

$$y(v) \leftarrow \arg\max_{c \in \mathcal{Y}} \left| \left\{ \mathbf{x}_j \in X_v | y_j = c \right\} \right|. \quad (4)$$

To classify a sample $\mathbf{x}$, the tree is traversed starting from the root v_0. At each node v, the sample is sent to v_l if $x_j^{a(v)} \le \tau(v)$ or to v_r otherwise. This process continues until $\mathbf{x}$ reaches a leaf node, where it is assigned a prediction.

The RF classifier combines multiple DTs, each trained on a randomly sampled subset of data and with randomly selected features for each split. This approach has demonstrated superior performance in the load recognition task [24], [27]. However, when deployed across different households, domain shift issues often arise due to variations in load characteristics, leading to degraded RF performance [28]. Therefore, it is essential to leverage TL techniques to adjust model parameters and adapt the RF model to new environments effectively.

III. Methods

This section describes the proposed methods for appliance recognition in NILM, including data collection, feature extraction, an updated DT strategy for parameter updates, and a transferable RF framework for classification.

A. Data Collection and Feature Extraction

In load recognition problems, current and voltage signals are used for analysis because they effectively reflect the operational characteristics of various appliances. Event-based NILM assumes that only one appliance is switched on or off at a time. When a significant change in active power exceeds a set threshold, one or multiple cycles of voltage and current waveforms are segmented around the event. As depicted in Fig. 2(a), the active power waveform is marked

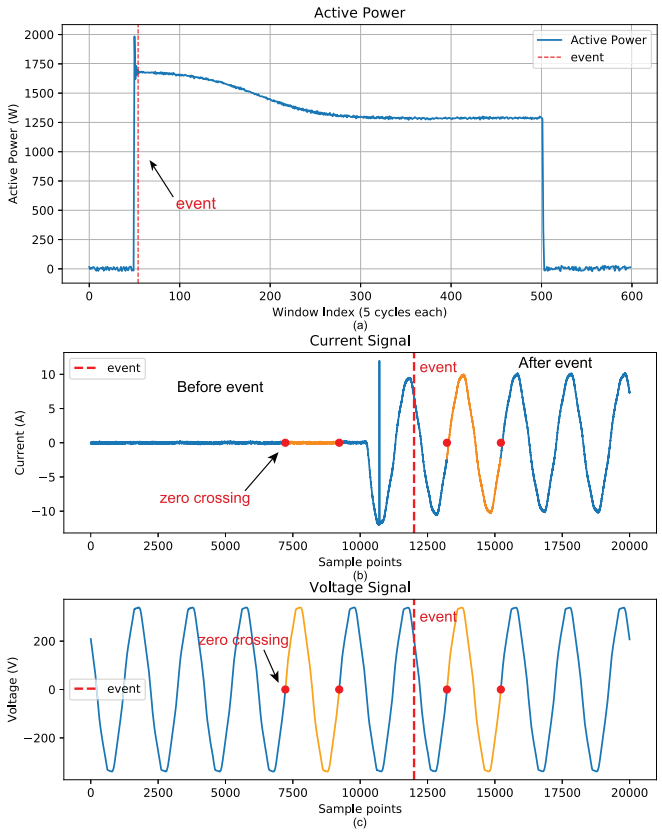

Fig. 2. Active power variation and segmented current and voltage waveform extraction for a fan in the COOLL dataset (index: 25). (a) Active power. (b) Current waveform. (c) Voltage waveform.

with red dashed lines to indicate the detected switching event, i.e., when $\Delta P > P_{\text{thresh}}$. The positive zero-crossing point of the ac voltage waveform is used as the reference to align the extracted signal cycles. Then, one complete cycle of steady-state current and voltage data is extracted before and after the event, corresponding to the active power waveform, as shown in Fig. 2(b) and (c). The samples of one-cycle-long voltage data records (one just before and the other just after each event) are averaged to mitigate the effect of noise. In contrast, the current samples before the event are subtracted from the current samples after the event to extract the actual consumption of the appliance that is switched on or off. In the following, the vectors of voltage and current data are then used to build feature sets for subsequent analysis.

Recent studies have shown that typical electrical signal features generally yield better performance [27]. Therefore, we adopted a variety of time- and frequency-domain signal features. It is well known that the real and reactive power profiles can easily be used to track appliance usage [5]. However, since the real and reactive power (P and Q) profiles may be insufficient for accurate NILM classification, especially in cases involving nonlinear loads, we also considered additional features of the current waveforms [29]. These include the root mean square (rms) value (I_{rms}), the amplitude of the odd current harmonics till the eleventh (i.e., $I_{\text{har1}}, I_{\text{har3}}, \ldots, I_{\text{har11}}$), and the total harmonic distortion (THD_I). Besides, a 16×16 binary V-I trajectory image will be used as the feature representation, which will be flattened into a 256-D vector to serve as the input to the model. The extracted features are used to train ML models, which can be deployed to monitor appliances in real-world scenarios. The overall process is illustrated in Fig. 1(c).

Since our method uses the RF model, feature importance can be estimated using the built-in impurity-based metric. However, a detailed feature engineering analysis is beyond the scope of this article. For further discussion, readers may refer to [30], [31], [32].

B. ISER Algorithm

When ML models are trained on specific household data, their performance often degrades in new environments due to domain shifts. To address these challenges, TL techniques have been developed to adapt pretrained models to new environments. Recently, two novel TL techniques for DTs were proposed by Segev et al. [33]: the structure expansion/reduction (SER) and structure transfer algorithms. Our methods build upon the SER algorithm.

Given a pretrained DT from the source domain and a small subset X^u from the target domain, the SER algorithm adapts the tree in two steps: *Expansion* and *Reduction*.

In the step of *expansion*, for each node v, the algorithm identifies X_v^u, the subset of X^u that reaches v. Each leaf node leaf node is then expanded by growing a binary subtree using the CART algorithm with X_v^u. The *BuildTree Function* in Algorithm 1 is used for this purpose.

The second step, called *Reduction*, involves pruning the tree structure by working bottom-up through the tree. Each internal node v is processed using the subset X_v^u from the target domain that reaches v. Two types of errors are computed for each node. The first one is the *leaf error*, which represents the empirical classification error if node v was pruned as a leaf node. It is defined as follows:

$$\text{leaf Err}\left(v, X_v^u\right) = \frac{1}{\left|X_v^u\right|} \sum_{i=1}^{\left|X_v^u\right|} \mathbf{1}_{\{y_i \neq y(v)\}} \tag{5}$$

where $|X_v^u|$ is the number of samples reaching v, y_i is the label of the i-sample of X_v^u, and $y(v)$ is the majority class at node v. $\mathbf{1}_{\{y_i \neq y(v)\}}$ is the indicator function that counts misclassified samples at node v. The indicator function $\mathbf{1}_{\{y_i \neq y(v)\}}$ returns 1 if the sample is misclassified and 0 otherwise.

The second type of error, the *subtree error*, represents the empirical error of the entire subtree rooted at node v. It is formally defined as follows:

$$\text{subtree Err}\left(v, X_v^u\right) = \frac{1}{\left|X_v^u\right|} \sum_{z \in \mathcal{L}(v)} \sum_{i=1}^{\left|X_z^u\right|} \mathbf{1}_{\{y_i \neq y(z)\}} \tag{6}$$

where $\mathcal{L}(v)$ denotes the set of all leaf nodes in the subtree rooted as v, X_z^u is the subset of samples that reach the leaf node z, and $y(z)$ is the majority class label at z. If the leaf error is smaller than the subtree error, the subtree is pruned into a leaf node. The decision value at each leaf of the DT is obtained using the target (empirical) distribution.

Algorithm 1 Pseudocode of the WTRF Algorithm

Input: X^u−target update dataset;
$\mathcal{T} = \{T_i\}_{i=1}^{N_t}$−a well-trained RF;
N_u−number of trees to be updated;
Output: Updated RF $\mathcal{T}'$;
$\mathcal{T}_{new} \leftarrow$Select N_u trees randomly from $\mathcal{T} = \{T_i\}_{i=1}^{N_t}$;
$\mathcal{T}_{old} \leftarrow \mathcal{T} - \mathcal{T}_{new}$;
for $i \leftarrow 1$ **to** N_u **do**
  $v \leftarrow$the root node in T_i^{new};
  $T_i^{new} \leftarrow$ **ISER**(v, X^u);
end
$\mathcal{T}' \leftarrow \mathcal{T}_{old} \cup \mathcal{T}_{new}$;
return $\mathcal{T}'$;
Function `ISER`$(v,\ X_v^u)$**:**
  Input: Node v;
  X_v^u:The subset that reaches node v;
  Output: Node v;
  `// Expansion`
  if *v is a leaf node* **then**
    $v \leftarrow$ **BuildTree**(X_v^u); `// CART Algorithm`
  end
  ISER$(v_l, X_{v_l}^u)$; `// Recursive left child`
  ISER$(v_r, X_{v_r}^u)$; `// Recursive right child`
  `// Reduction`
  if leafErr(v, X_v^u) < subtreeErr(v, X_v^u) **then**
    if $c_{\min} \notin y(X_v^s)$ **then**
      for $v_i \in \{v_l, v_r\}$ **do**
        deleteNode(v_i);
      end
      $y(v) \leftarrow \underset{c \in \mathcal{Y}}{\operatorname{argmax}} \left|\{\mathbf{x} \in X_v^u | y = c\}\right|$;
    end
  end
  return v;
return

While the pruning step generally enhances efficiency and avoids overfitting, it may lead to suboptimal decisions in specific scenarios. For instance, when the target domain has an insufficient number of samples for a particular class, pruning may incorrectly discard useful structures, resulting in unfavorable outcomes. To illustrate the pruning risk, we adopt the definition of leaf loss risk as described in [34].

Definition 1: A minority class leaf v, corresponding to class $c_{\min}$, is considered significant for the target domain if

$$p^T(y = c_{\min} | \mathbf{x} \in v) > p^T(y = c | \mathbf{x} \in v) \quad \forall c \neq c_{\min} \tag{7}$$

where $c_{\min}$ represents the minority class.

Definition 2: Let $n_t(c)$ denote the number of samples for class c available for updating in the target domain. The pruning risk of deleting a minority class leaf v is quantified as follows:

$$R_{n_t(c_{\min})}(v) = \left[p^T(x \notin v | y = c_{\min})\right]^{n_t(c_{\min})} \tag{8}$$

where $p^T(x \in v | y = c_{\min})$ represents the probability that a sample of class $c_{\min}$ does not belong to leaf v. This risk metric captures the likelihood of removing significant structures for the minority class.

When the sample size $n_t(c)$ is sufficient and the classes are balanced, the pruning risk $R_{n_t(c_{\min})}(v)$ approaches zero, and pruning becomes negligible. However, in our study, only one to three samples per appliance are available for model updates, which may result in a significant pruning risk, particularly for minority class leaves. As $n_t(c)$ decreases, the pruning risk for minority leaves increases, leading to a higher number of pruned leaves and potential degradation in model performance.

During the transfer process, pruning risk for the minority class can arise under two circumstances: 1) the target update data reaches v, causing the majority class at v to change and 2) the target update data does not reach v, leaving insufficient samples to preserve the minority class leaves. These scenarios increase the risk of incorrectly pruning minority class structures. To address class proportion shifts between domains, an improved SER (ISER) algorithm introduces a condition to preserve minority class leaves during the reduction phase. This condition ensures that minority class leaves are retained while allowing their controlled expansion. A pseudocode of this algorithm is presented in the *ISER Function* in Algorithm 1.

C. Weighted Transferable RF

Based on the ISER algorithm, a WTRF method is proposed to adapt tree-based models to the target domain. The WTRF algorithm selectively updates a subset of pretrained trees using the update dataset X^u while retraining the remaining trees to form a hybrid model. This approach integrates the adaptability of ISER with the robustness of the RF framework, ensuring efficient handling of domain shifts. The steps are given as follows.

Step 1: Let $\mathcal{T} = \{T_i\}_{i=1}^{N_t}$ denote a pretrained RF from the source domain, where T_i represents a DT and N_t is the number of trees in the forest.

Step 2: Randomly select N_u trees ($N_u < N_t$) from $\mathcal{T}$ to create $\mathcal{T}_{\text{new}} = \{T_i'\}_i^{N_u}$,, while $\mathcal{T}_{\text{old}} = \mathcal{T} \backslash \mathcal{T}_{\text{old}}$ contains the retained trees. Each tree in $\mathcal{T}_{\text{new}}$ is updated using the ISER algorithm and X^u. The selective update ensures that the method remains computationally efficient while adapting to the target domain.

Step 3: Once $\mathcal{T}_{\text{new}}$ is obtained, a new hybrid model can be constructed by

$$\mathcal{T}' \leftarrow \mathcal{T}_{\text{new}} \cup \mathcal{T}_{\text{old}}. \tag{9}$$

The WTRF algorithm is detailed in Algorithm 1. By updating only a subset of trees, this approach balances adaptation to the target domain with retaining source-domain knowledge. Once $\mathcal{T}'$ is obtained, it can be used to make a prediction of the target domain data. Since new trees $\mathcal{T}_{\text{new}}$ are likely better adapted to the target domain than old trees $\mathcal{T}_{\text{old}}$, give more weight on them, and the trees in $\mathcal{T}_{\text{old}}$, are weighted to have less voting power. Thus, the final prediction of the model can be expressed as follows:

$$\hat{\mathcal{T}}'(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \left(\sum_{T_i \in \mathcal{T}_{\text{new}}} w_{\text{new}}\, \hat{p}_c(T_i) + \sum_{T_i \in \mathcal{T}_{\text{old}}} w_{\text{old}}\, \hat{p}_c(T_i) \right) \tag{10}$$

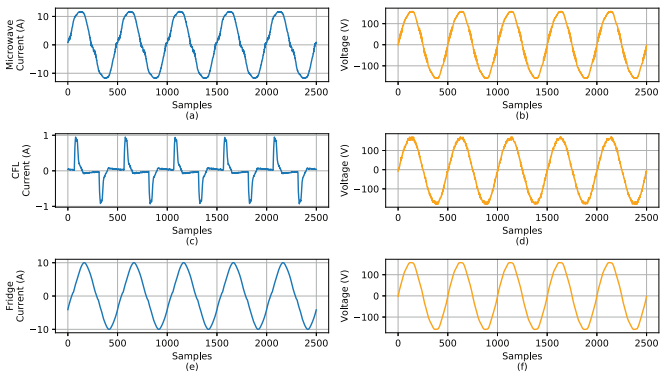

Fig. 3. Waveforms of current and voltage signals for a microwave, CFL, and fridge in the PLAID dataset.

where $\hat{p}_c(T_i)$ is the predicted probability of class c from tree T_i. $w_{\text{new}} \in [0, 1]$ and $w_{\text{old}} \in [0, 1]$ are the weights assigned to $\mathcal{T}_{\text{new}}$ and $\mathcal{T}_{\text{old}}$,, respectively. Note that $w_{\text{new}} > w_{\text{old}}$, and $w_{\text{new}} + w_{\text{old}} = 1$.

The combination of updated and retained trees leverages the adaptability of $\mathcal{T}_{\text{new}}$ while preserving the robustness of $\mathcal{T}_{\text{old}}$, ensuring a balance between computational efficiency and adaptability. This design allows the WTRF algorithm to address domain shifts effectively in NILM tasks. Note that the proposed method adopts a TL approach. Model updating is required only once when deploying to a new household or replacing monitored appliances. After the update, the model functions as a standard RF predictor, without further adaptation.

IV. Case Settings

This section presents the experimental setup used to evaluate the proposed method, including the datasets, performance indicators, comparison methods, and evaluation procedure.

A. Dataset Description

In this study, we use three publicly available datasets to evaluate the proposed method: PLAID [35], WHITED [36], and COOLL [37], following the original labeling specifications provided by each dataset. Each of these provides voltage and current measurements from various appliances.

1) The PLAID dataset contains 1793 voltage and current waveforms sampled at 30 kHz, representing 11 appliance types collected from over 50 U.S. households. Fig. 3 shows waveforms of current and voltage signals for a microwave, compact fluorescent lamp (CFL), and fridge in the PLAID dataset.
2) The WHITED dataset is a public dataset comprising submetered voltage and current measurements sampled at 44 kHz for 46 appliance types. Each appliance type includes data from one to nine different brands, with ten start-up events per brand, resulting in 1100 total measurements.
3) The COOLL dataset contains 840 voltage and current measurements for 42 controllable appliances, sampled at 100 kHz. For each appliance, it includes 20 turn-on transient signals with varied delays relative to the mains voltage zero-crossing. These appliances have 12 different types, each with multiple instances.

B. Performance Indicators

To evaluate the performance of our method, we use widely recognized metrics in load recognition: accuracy (Acc), precision (Pre), recall (Rec), and $F1$-score. Accuracy measures the percentage of correctly classified results over the total samples and is defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{11}$$

where TP, TN, FP, and FN denote the number of true positives, true negatives, false positives, and false negatives, respectively.

The $F1$-score is a useful metric, particularly for imbalanced data, and is calculated separately for each appliance type. For the ith appliance type, the $F1$-score is defined as follows:

$$F1_i = \frac{2 \cdot \text{Pre}_i \cdot \text{Rec}_i}{\text{Pre}_i + \text{Rec}_i} \tag{12}$$

where

$$\text{Pre}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \quad \text{and } \text{Rec}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{13}$$

are the precision and recall in classifying the items of class i. We use the macroaverage $F1$-score for overall evaluation in multiclass classification, calculated as $F_{\text{macro}} = (1/C) \sum_{i=1}^{C} F1_i$.

C. Comparison Methods

To verify the effectiveness of the proposed approach, the following groups of methods are carried out for comparison.

1) *Source-Only:* The source-only approach refers to training only on source data and testing on target data without any TL techniques. We select four state-of-the-art (SOTA) methods that have demonstrated superior performance in previous studies: RF [24], TFCNN [6],[2] AWRG [38],[3] and LILA [39].[4] These methods are chosen for their proven effectiveness and availability of open-source code, which makes them easy to reproduce.
2) *Feature-Based:*Four commonly used UDA methods are adopted for evaluation: DANN [12],[5] ADNN [13],[6] MK-MMD [14],[7] and Deep CORAL [15].[8] This group of methods utilizes both labeled source data and unlabeled target data, classifying them as UDA methods. This categorization enables us to evaluate the effect of UDA methods on load identification. We select a CNN model using *V-I* trajectories as the baseline [7].

[2]https://github.com/zhz-yan/HSV-VI-NILM
[3]https://github.com/sambaiga/AWRGNILM
[4]https://github.com/Deep-fishing/ALILS
[5]https://github.com/fungtion/DANN
[6]https://github.com/corenel/pytorch-adda
[7]https://github.com/thuml/DAN
[8]https://github.com/SSARCandy/DeepCORAL

3) *Instance-Based:* Instance-based methods aim to reweight samples in the source domain to address marginal distribution differences and improve alignment with the target domain. In this study, we adopt two commonly used approaches: KMM [40] and TrAdaBoost [41].[9] KMM is an unsupervised method that reweights samples by aligning the means of source and target distributions. TrAdaBoost is a supervised method that uses 10% of the target data to reweight the source data in this study.
4) *Fine-Tuning:*In this study, the models in the source-only group are first pretrained on source data. Then, we fine-tune these models using a small subset (10%) of the target data. Following prior works [1], [6], only the last FC layer is fine-tuned to adapt to the target domain for CNN-based models, while the CNN layers remain untuned. For the RF, we retrain the model using 10% of the target data.

D. Evaluation Procedure

In this study, we evaluate the proposed method in scenarios involving both intradomain and interdomain shifts. Intradomain shifts are assessed in *Case* 1 to validate the method's suitability for online applications, while interdomain shifts are tested in *Cases* 2 and 3 from two perspectives: household and dataset.

Case 1: For intradomain shifts, one house is randomly selected as the source domain, and one appliance is replaced with a device of a different brand or model (of the same type). During online monitoring, the proposed method detects domain shifts caused by appliance replacement and updates the model accordingly. The test is repeated 20 times, with a different house and appliance randomly selected in each iteration.

Case 2: From the household perspective, the training and testing sets are separated by household. For the WHITED and COOLL datasets, we perform LoHoCV. For the PLAID dataset, where some houses have very few samples, we randomly select 20 houses for testing and use the remaining houses for training. Each test is repeated 20 times to compute the average (Avg.) and standard deviation (STD).

Case 3: From the dataset perspective, models trained on the PLAID dataset are reused to classify the same 11 types of appliances in the WHITED dataset.

Note that the annotation of measurement households is only available in the PLAID dataset. For the WHITED and COOLL datasets, households are artificially created by randomly assigning each appliance of each type to a household [7]. The total number of households is set to 9 in the WHITED dataset and 8 in the COOLL dataset, corresponding to the maximum number of appliances per type. Consequently, appliance types with only one sample per type are excluded.

In this study, k_i^T and k_i^S represent the number of samples per appliance in the target and source domains, respectively. For example, $k_i^T = 1$ indicates that one sample per appliance is used to update the model in the target domain.

[9]https://github.com/surajiyer/Transfer-learning-with-TrAdaBoost

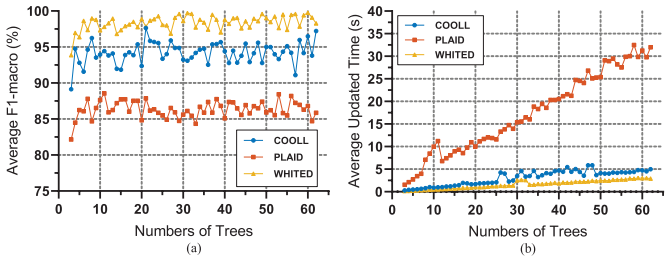

Fig. 4. Impact of N_t on (a) avg. $F1$-macro and (b) update time ($k_i^T = 1$).

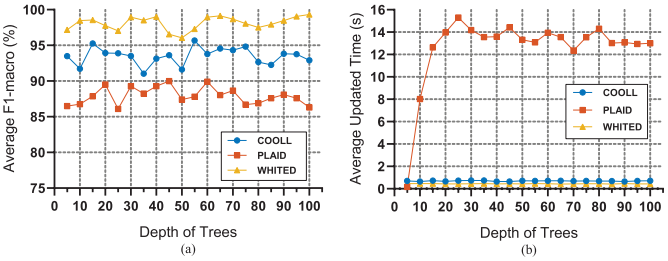

Fig. 5. Impact of $h_{\max}$ on (a) avg. $F1$-macro and (b) update time ($k_i^T = 1$).

TABLE I

$F1$-MACRO VALUE RESULTS UNDER DIFFERENT k_i^S AND k_i^T CONFIGURATIONS IN CASE 1

k_i^S	k_i^T	F-macro			$F1_{new}$		
		PLAID	WHITED	COOLL	PLAID	WHITED	COOLL
0	0	89.8±6.8	87.9±11.9	76.8±18.3	36.2±45.2	51.2±48.5	43.0±45.6
0	1	87.6±8.8	81.7±14.4	67.8±15.9	30.0±45.8	35.0±47.7	15.0±35.7
0	3	88.6±8.6	89.1±12.5	66.7±16.8	35.0±47.7	55.0±49.7	19.9±39.7
1	1	94.7±5.2	99.3±2.3	90.6±10.3	88.0±20.4	100±0.0	86.6±22.6
1	3	97.2±4.8	99.2±2.1	92.2±11.2	92.4±20.9	99.7±1.1	88.6±21.4
3	1	97.9±3.1	99.0±2.4	93.1±8.5	96.9±6.6	99.1±3.8	96.6±4.7
3	3	98.3±2.9	99.1±2.3	91.5±7.6	95.9±9.3	100.0±0.0	94.0±7.0

$F1_{new}$ denotes the F1-macro score computed exclusively for the newly introduced appliances.

V. RESULTS AND DISCUSSION

A. Parameter Studies

Experimental results of Cases 1–3 are shown in Tables I–III respectively. Test results of average training, updated and testing times, and memory footprint of the proposed methods are provided in Table IV. Results obtained on the Raspberry Pi 4 are listed in Table Table V. We first investigate the impact of various parameters on the performance of the WTRF method. Let N_t denote the number of trees, $h_{\max}$ denote the parameter controlling the maximum depth of trees, and p_{new} represent the proportion of new trees in the total forest, i.e., $p_{\text{new}} = N_u/N_t$. Here, we evaluate the impact of N_t, $h_{\max}$, w_{new}, and p_{new} on model accuracy and update time. The evaluation is performed using LoHoCV, with each test repeated ten times. For model updates, one sample is selected from each appliance in the target domain, i.e., $k_i^T = 1$.

1) Effect of N_t: To investigate the impact of N_t on prediction performance, we varied N_t from 3 to 62 with a step size of 1. The depth of each tree was not constrained, and we fixed $w_n = 1.0$ and $p_{\text{new}} = 1.0$. Fig. 4(a) illustrates that $F1$-macro improves significantly as N_t increases, particularly when N_t is below 10. Beyond this threshold, the performance stabilizes due to sufficient model capacity. However, as shown in Fig. 4(b), excessive trees increase model complexity, resulting

TABLE II
EXPERIMENTAL RESULTS IN CASE 2

Methods	PLAID				WHITED				COOLL			
	Accuracy	Precision	Recall	F1-macro	Accuracy	Precision	Recall	F1-macro	Accuracy	Precision	Recall	F1-macro
Classic RF (Source)	74.9±6.2	71.7±5.4	69.8±5.9	68.4±6.2	50.6±14.5	38.3±17.2	42.8±14.8	38.4±15.0	53.7±17.8	43.0±19.0	43.2±15.8	39.9±17.1
TFCNN (Source)	75.6±2.8	72.0±4.5	70.9±3.9	69.9±4.1	55.3±14.2	42.1±14.0	45.8±12.8	42.0±13.7	51.3±14.9	40.1±16.3	40.7±15.6	37.5±15.0
AWRG (Source)	77.6±4.7	77.1±4.3	75.4±5.2	74.1±4.8	54.8±20.2	40.8±19.6	45.1±18.5	40.6±19.1	27.5±15.2	20.6±13.4	22.2±13.2	17.2±11.0
LILA (Source)	80.4±4.7	80.3±4.4	80.4±3.6	78.8±3.6	55.7±16.4	42.8±16.3	42.5±14.3	40.8±14.7	37.3±19.5	27.1±15.7	28.1±14.5	25.4±15.0
DANN	62.0±20.3	47.5±24.5	49.4±23.9	46.8±24.8	51.4±18.7	40.6±18.5	42.1±16.2	39.1±17.0	34.6±24.3	23.6±24.2	27.7±18.7	22.0±20.6
ADDN	60.6±5.8	57.2±4.9	59.6±4.3	56.0±4.6	32.8±15.0	23.1±12.5	25.4±12.8	22.5±11.8	17.2±13.2	11.8±11.5	13.6±10.7	11.1±10.0
MK-MMD	53.2±29.3	44.7±26.1	37.2±27.8	38.6±27.4	43.8±12.8	32.2±15.9	34.8±13.6	30.4±13.1	31.0±13.8	23.7±12.9	20.0±10.4	19.1±10.3
Deep CORAL	47.5±27.9	36.8±25.9	36.6±26.1	34.1±26.1	54.4±16.5	42.3±19.5	42.4±16.5	39.9±18.0	23.5±10.0	20.3±9.1	14.2±6.4	14.5±6.2
TrAdaBoost (10%)	79.0±3.4	76.0±4.0	75.2±4.8	74.5±4.5	89.6±7.1	85.6±11.0	82.0±11.9	81.9±11.9	83.3±9.6	72.0±11.7	65.7±13.8	66.8±13.0
KMM	71.8±4.7	67.4±4.1	69.5±3.7	66.4±4.0	47.7±18.8	39.8±17.5	35.7±17.9	35.9±16.7	54.1±16.0	40.5±13.8	42.4±14.6	38.7±13.4
Retra. RF (10%)	74.7±2.7	74.0±6.2	68.2±5.5	68.6±5.5	99.5±0.8	99.5±0.7	99.5±0.8	99.5±0.8	93.5±5.6	94.6±4.3	93.5±5.6	93.2±6.1
AWRG (F-T)	82.5±4.1	82.7±4.4	78.8±4.9	79.4±4.9	85.1±14.6	84.5±15.5	84.7±15.2	82.8±16.7	60.1±15.7	57.6±21.0	60.1±15.7	53.3±18.7
TFCNN (F-T)	78.4±3.4	75.6±3.7	75.3±2.9	74.0±3.2	92.0±7.8	92.4±8.0	91.5±7.9	90.5±9.0	89.6±8.1	86.7±9.8	85.0±11.3	85.0±10.9
LILA (F-T)	65.7±8.5	65.5±8.0	60.6±8.1	59.6±8.0	72.3±17.6	74.8±19.8	72.3±17.6	69.6±19.1	49.3±14.8	49.0±18.9	49.2±14.8	45.9±16.8
Proposed (1-shot)	87.7±5.2	87.9±4.6	87.2±4.5	86.1±0.3	98.5±3.8	98.3±5.4	98.5±3.8	98.2±4.9	89.0±7.6	91.7±5.1	89.0±7.6	88.0±8.8
Proposed (3-shot)	92.9±3.4	92.3±4.0	93.6±3.0	92.3±3.7	99.6±0.7	99.7±0.6	99.6±0.7	99.6±0.7	95.0±5.2	96.0±4.2	95.0±4.5	94.8±5.5

TABLE III
EXPERIMENTAL RESULTS IN CASE 3

Methods	Accuracy	Precision	Recall	F1-macro
RF (Source)	9.3±3.1	7.8±2.2	5.7±3.3	5.7±1.7
TFCNN (Source)	11.5±1.7	10.0±5.3	8.3±2.1	6.8±2.2
AWRG (Source)	25.9±7.1	17.0±4.5	31.3±3.9	19.1±3.6
LILA (Source)	13.4±29	11.1±2.0	19.1±4.3	10.4±2.6
DANN	5.4±1.4	3.5±2.5	6.4±2.6	0.0±0.0
ADDN	5.1±3.6	3.4±2.6	5.0±2.8	0.0±0.0
MK-MMD	8.1±2.4	7.4±3.4	6.7±2.4	0.1±0.0
Deep CORAL	10.1±2.1	8.0±1.9	7.8±1.7	0.1±0.0
TrAdaBoost (10%)	83.0±3.7	78.1±4.6	80.4±4.5	77.2±5.0
KMM	10.4±2.3	7.1±1.7	6.4±1.7	0.1±0.0
Retra. RF (10%)	92.3±2.3	91.2±6.0	87.6±5.6	88.0±5.9
AWRG (F-T)	66.7±6.9	69.0±9.8	66.1±8.7	62.8±9.2
TFCNN (F-T)	66.9±3.7	60.3±7.1	58.5±7.0	55.8±6.4
LILA (F-T)	62.0±12.3	60.0±14.3	56.9±11.2	55.1±12.8
Proposed (1-shot)	97.7±1.1	98.1±1.4	96.8±2.5	97.0±2.6
Proposed (3-shot)	98.1±1.6	98.1±1.6	97.4±1.7	97.6±1.7

TABLE IV
TEST OF AVERAGE TRAINING, UPDATED AND TESTING TIMES, AND MEMORY FOOTPRINT OF THE PROPOSED METHODS

	PLAID	WHITED	COOLL
Avg. training time (s)	0.049±0.011	0.015±0.003	0.017±0.001
Avg. update time (s)	12.107±2.012	0.499±0.081	0.742±0.101
Avg. testing time (s)	0.001±0.001	0.001±0.000	0.001±0.001
Memory footprint (KB)	34.6±100.3	123.1±178.7	131±240.2
Model size (KB)	299.8±8.4	108.9±7.3	103.7±4.1
F1-macro (%)	94.4±10.4	98.2±2.1	95.9±4.6
Accuracy (%)	94.8±10.2	98.3±2.0	95.9±4.5

TABLE V
RESULTS OBTAINED ON THE RASPBERRY PI 4

	PLAID	WHITED	COOLL
Update time (s)	66.529±5.926	3.071±0.337	5.389±0.994
Testing time (s)	0.003±0.000	0.003±0.000	0.005±0.001
Avg. update samples	4.7±2.3	7.6±2.2	9.8±2.3
Avg. n_{node}	226.9±7.0	75.2±3.7	86.1±5.5
Avg. h_{max}	14.3±0.5	9.4±0.5	11.7±0.7
F1-macro (%)	94.6±10.1	98.5±3.2	93.9±5.9
Accuracy (%)	94.9±9.9	98.68±3.0	93.9±6.0

in higher update times. To balance update time and accuracy, we selected $N_t = 11$ for subsequent experiments.

2) Effect of $h_{\max}$: We analyzed the impact of $h_{\max}$ on model performance by varying $h_{\max}$ from 5 to 100 in steps of 5. As shown in Fig. 5, the results indicate that increasing $h_{\max}$ beyond 20 provides minimal improvements. This is because the limited feature set and small number of update samples limit deeper splits. Thus, we set $h_{\max} = 20$ for the remaining experiments, as moderate tree depths are sufficient to achieve promising results.

Fig. 5 shows that the $F1$-macro on the PLAID dataset improves significantly when $h_{\max} < 15$ but exhibits diminishing returns beyond that point. Similarly, the update time increases with deeper trees and begins to fluctuate when $h_{\max} > 20$, as shown in Fig. 5(b). These trends are mainly attributed to differences in the complexity of pretrained trees. The PLAID dataset contains significantly more appliance instances than WHITED and COOLL, which leads to deeper pretrained DTs with a higher number of nodes. As a result, the update process becomes more time-consuming since more nodes may be affected during model adaptation and require modification. As shown in Table V, the pretrained decision trees are generally deeper and have more nodes (e.g., $h_{\max} = 14.3 \pm 0.5$ and $n_{\text{node}} = 226.9 \pm 7.0$). When $h_{\max}$ is set too low (e.g., <10), these deep trees are pruned prematurely, leading to suboptimal learning. As $h_{\max}$ increases, the model can better leverage the tree structure and improve its performance. However, once the trees reach their effective depth (e.g., $h_{\max} > 20$), further increases in depth have little effect, as no additional splits occur during model updating.

3) Effect of p_{New} and w_{New}: The impact of p_{new} and w_{new} was studied by varying p_{new} from 0.4 to 0.9 and w_{new} from 0.6 to 0.9, both with a step size of 0.1. We conducted two sets of tests on the COOLL and WHITED datasets, respectively, as illustrated in Fig. 6. The results reveal that increasing p_{new} initially improves $F1$-macro scores, but performance plateaus when $p_{\text{new}} > 0.7$. While larger p_{new} values improve adaptation, they also increase update times due to the computational cost of incorporating more trees. Through trial and error, we determined $p_{\text{new}} = 0.7$ and $w_{\text{new}} = 0.8$ in subsequent experiments to ensure efficient updates without compromising performance.

B. Results on Case 1

Table I presents the experimental results of Case 1. When no samples from either the old or new appliances are used to update the model ($k_i^T = k_i^S = 0$), the model struggles to

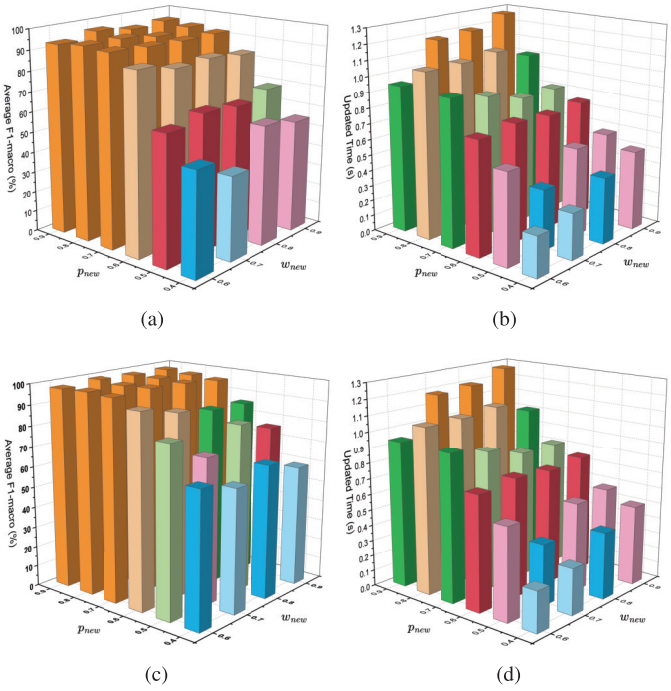

Fig. 6. Effect of p_{new} and w_{new} on model performance and efficiency. (a) Avg. $F1$-macro in COOLL. (b) Update time in COOLL. (c) Avg. $F1$-macro in WHITED. (d) Update time in WHITED.

classify new appliances, with $F1$new values consistently below 50%. This highlights the model's limited ability to generalize to unseen appliances without additional training data. When updates rely solely on new appliance samples ($k_i^S = 0$ and $k_i^T = 1$ or 3), $F1_{\text{new}}$ improves slightly, but $F1$-macro values for old appliances decline significantly. This indicates catastrophic forgetting, where the model loses previously learned information due to the absence of old appliance data.

As k_i^T and k_i^S increase, the model's performance improves significantly across all datasets. Both $F1$-macro and $F1_{\text{new}}$ achieve their highest values when $k_i^T = k_i^S = 3$, indicating the importance of including samples from both new and old appliances during updates. This finding suggests that in the online deployment of NILM models, when and only when individual appliances are replaced, the retraining process should include samples from unchanged appliances to prevent catastrophic forgetting while adapting to the new ones. Nevertheless, even providing only one to three samples per device is sufficient to effectively adjust model parameters and achieve excellent performance.

C. Results on Case 2

Table II presents the experiment results of Case 2. Models trained solely on source domain data perform poorly, particularly on COOLL, where the highest macro $F1$-score among source-only models is just 39.8% ± 17.1%. These results underscore the limitations of source-only models in handling domain shifts effectively.

While UDA techniques offer promising scenarios without labeled target data, they may underperform under substantial domain shifts. For instance, ADDN achieves a macro $F1$-score of 56.0% ± 4.6% on PLAID, which is lower than some source-only models. This indicates that UDA methods may struggle with severe domain shifts, occasionally leading to negative transfer.

In contrast, incorporating a small amount of labeled target data significantly improves performance. Retrained RF achieves macro $F1$-scores of 93.2% ± 6.1% on COOLL, and TrAdaBoost achieves 74.5% ± 4.5% on PLAID using just 10% of target data. F-T DL-based models also yield better results in most cases, such as AWRG (F-T) achieving 82.2% ± 16.7% on WHITED. However, F-T can sometimes result in negative transfer, as observed with LILA (F-T), where its macro $F1$-score drops from 78.8% ± 3.6% to 59.6% ± 8.0%. These findings emphasize that introducing labeled target data is far more effective for adapting to domain shifts.

The proposed methods consistently outperform both source-only and UDA approaches. With just one sample per appliance, the proposed (one-shot) achieves an $F1$-macro of 86.1% ± 0.3% on PLAID, 98.2% ± 4.9% on WHITED, and 88.0% ± 8.8% on COOLL. Increasing to three samples per load, the proposed (three-shot) further improves performance to 92.3% ± 3.7%, 99.6% ± 0.7%, and 94.8% ± 5.5%, respectively. These results demonstrate the effectiveness of the proposed methods in handling intradomain shifts, achieving high performance even with minimal target data while mitigating the risks of negative transfer.

D. Results on Case 3

In Case 3, models trained on the PLAID are tested on the WHITED, where significant interdomain shifts present challenges for most methods. Source-only models perform poorly, with macro $F1$-scores ranging from 5.7% ± 1.7% to 19.1% ± 3.6%. This highlights the difficulty of directly applying models across datasets with distinct distributions. Similarly, UDA methods fail to bridge the domain gap effectively, with macro $F1$-scores close to 0%, indicating severe negative transfer. These results underscore the impact of substantial feature differences between datasets, a limitation that UDA methods fail to address effectively.

Incorporating 10% labeled target domain data significantly improves performance. TrAdaBoost and Retrained RF achieve macro $F1$-scores of 77.2% ± 5.0% and 88.0% ± 5.9%, respectively, demonstrating the benefit of labeled target data. F-T neural models provide moderate improvements but show inconsistencies. For example, AWRG (F-T) achieves a macro $F1$-score of 62.8% ± 9.9%, but LILA (F-T) suffers from negative transfer, dropping to 55.1% ± 12.8%.

The proposed method consistently outperforms all other approaches, achieving 97.0% ± 2.6% (one-shot) and 97.6% ± 1.7% (three-shot). This demonstrates its ability to efficiently leverage minimal labeled data to adapt to new domains, overcoming interdomain shifts while maintaining robust performance. These results underscore the necessity of incorporating labeled target data to mitigate domain discrepancies and achieve superior generalization.

E. Computational Efficiency Analysis

The proposed technique updates a pretrained RF. Let n_{node} be the average number of nodes per tree. During the update

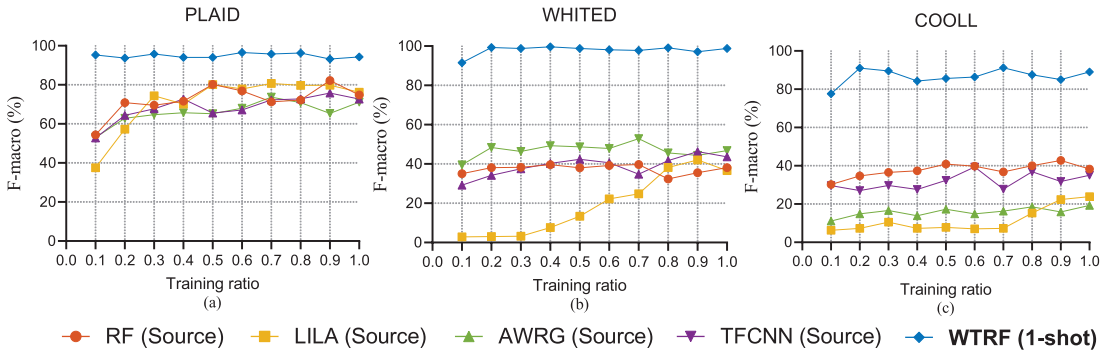

Fig. 7. Impact of training ratios on $F1$-Macro Scores in (a) PLAID, (b) WHITED, and (c) COOLL.

phase, the algorithm evaluates whether each node requires modification based on error metrics (*leafErr* and *subtreeErr*) using n_u samples, taking $O(n_u)$ time per node. If modification is required, the algorithm either reassigns the class label or rebuilds subtrees. Rebuilding involves constructing a DT using the CART algorithm, which has a complexity of $O(Mn_u \log_2(n_u))$ for each rebuilt node. For a single tree with n_{node}, the worst case complexity of the update phase is $O(n_{\text{node}} n_u + n_{\text{rebuild}} M n_u \log_2(n_u))$, where n_{rebuild} is the number of nodes requiring subtree rebuilding. When extending to N_u trees, the overall computational complexity becomes $O(N_u \cdot (n_{\text{node}} n_u + n_{\text{rebuild}} M n_u \log_2(n_u)))$.

To analyze computational efficiency, Table IV presents the average training time, update time, testing time, memory footprint, and model size of the proposed method under LoHoCV with one-shot updating. The proposed method achieves fast update times on WHITED (0.499 ± 0.081 s) and COOLL (0.742 ± 0.101 s), while PLAID exhibits a higher update time (12.107 ± 2.012 s) due to its larger node numbers.

The method maintains a low memory footprint across all datasets, with values of 34.6 ± 100.3 kB (PLAID), 123.1 ± 178.7 kB (WHITED), and 131.1 ± 240.2 kB (COOLL). In addition, its compact model size, particularly for WHITED (108.9 ± 7.3 kB) and COOLL (103.7 ± 4.1 kB), ensures compatibility with resource-constrained devices. Compared to DL-based methods, the proposed technique requires significantly less memory and computation while maintaining ease of integration into RF-based methods [24], [27]. These results highlight its suitability for deployment in edge computing environments.

F. Edge Computing Implementation

The deployment of NILM systems on embedded devices is crucial to enable real-time, low-latency, and resource-efficient energy monitoring for practical applications. The proposed technique was implemented and tested on a Raspberry Pi 4 board to evaluate its performance in an edge computing environment. We selected Raspberry Pi due to its wide adoption in previous NILM studies [42], [43], [44], strong community support,[10] and cost-effectiveness, despite the availability of more powerful alternatives such as the NVIDIA Jetson Nano and Odroid XU4. Using LoHoCV and updating the model with one sample per appliance, the method achieves high $F1$-macro scores of 94.6% $\pm$ 10.1% (PLAID), 98.5% $\pm$ 3.2% (WHITED), and 93.9% $\pm$ 5.9% (COOLL), as shown in Table V.

The update time varies significantly across datasets. PLAID requires 66.5 ± 6.0 s due to its larger number of appliances on the source domain, leading to a more complex tree structure with greater depth ($h_{\max} = 14.3 \pm 0.5$) and more nodes ($n_{\text{node}} = 226.9 \pm 7.0$). In contrast, WHITED and COOLL, with simpler tree structures, achieve much faster update times of 3.1 ± 0.3 s and 5.4 ± 1.0 s, respectively. Despite requiring only 4.7 ± 2.3 update samples per house on PLAID, the complexity of the tree increases the update time. Once updated, the model achieves extremely low prediction times (e.g., 3 ms per house for PLAID), demonstrating its suitability for real-time applications on resource-constrained devices.

G. Effect of Training Set Size

Practical applications of NILM require load recognition models to generalize effectively to unseen environments, including new appliances and households. It is widely assumed that increasing the training data improves generalization performance on new houses. To evaluate this, we adopt LoHoCV and vary the training set ratio from 0.1 to 1.0 in increments of 0.1, repeating each test 20 times. As shown in Fig. 7, increasing the training data slightly improves DL-based methods, but their performance remains poor, with $F1$-macro scores below 60% on WHITED and COOLL, even with the full training set. This highlights the limitations of existing models in adapting to new environments due to their sensitivity to domain shifts.

In contrast, the proposed WTRF method achieves consistently high $F1$-macro scores across all datasets, performing near-optimally on PLAID and WHITED with a training ratio as low as 0.1. These findings underline the effectiveness of WTRF in leveraging limited labeled data to achieve robust performance, even under significant domain shifts.

VI. Conclusion

This article proposes a WTRF method for load recognition in NILM, addressing performance degradation caused by domain shifts. The proposed method requires only a small number of labeled samples from the new environment for

[10] https://forums.raspberrypi.com/

model updates (e.g., one to three samples per appliance). When using just one sample per appliance, the method achieves an $F1$-macro of 97.0% $\pm$ 1.7% when transferring from the PLAID dataset to the WHITED dataset, significantly outperforming source-only models, fine-tuned models, and domain adaptation methods. Moreover, the model exhibits low computational overhead, requiring less than 300 kB in total memory.

On a Raspberry Pi 4, it achieves $F1$-macro scores of 94.6% $\pm$ 10.1% (PLAID), 98.5% $\pm$ 3.2% (WHITED), and 93.9% $\pm$ 5.9% (COOLL) under LoHoCV. The average update times are 66.5 ± 6.0 s (PLAID), 3.1 ± 0.3 s, and 5.4 ± 1.0 s, highlighting the method's suitability for resource-constrained edge computing environments.

Limitations and Future Works: This study assumes that the target and source domains share the same label space, which may not hold in real-world scenarios. In addition, the algorithm's complexity increases with the number of trees and nodes in the source model, resulting in longer update times for complex models. Nevertheless, this limitation can be mitigated by optimizing tree depth and the number of trees in future studies. Future work will focus on extending the method to accommodate differing label spaces and exploring ways to leverage unlabeled target data, enhancing adaptability to real-world conditions.

REFERENCES

[1] M. D'Incecco, S. Squartini, and M. Zhong, "Transfer learning for non-intrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1419–1429, Mar. 2020.

[2] D. Zheng, X. Ma, Y. Wang, Y. Wang, and H. Luo, "Non-intrusive load monitoring based on the graph least squares reconstruction method," *IEEE Trans. Power Del.*, vol. 37, no. 4, pp. 2562–2570, Aug. 2022.

[3] K. Carrie Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? The case of electricity," *Energy Policy*, vol. 52, pp. 213–234, Jan. 2013.

[4] W. Chen, Q. Gong, G. Geng, and Q. Jiang, "Cloud-based non-intrusive leakage current detection for residential appliances," *IEEE Trans. Power Del.*, vol. 35, no. 4, pp. 1977–1986, Aug. 2020.

[5] G. W. Hart, "Nonintrusive appliance load monitoring," *Proc. IEEE*, vol. 80, no. 12, pp. 1870–1891, Jun. 1992.

[6] Y. Liu, X. Wang, and W. You, "Non-intrusive load monitoring by voltage-current trajectory enabled transfer learning," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5609–5619, Sep. 2019.

[7] L. De Baets, J. Ruyssinck, C. Develder, T. Dhaene, and D. Deschrijver, "Appliance classification using VI trajectories and convolutional neural networks," *Energy Build.*, vol. 158, pp. 32–36, Jan. 2018.

[8] D. Murray, L. Stankovic, V. Stankovic, S. Lulic, and S. Sladojevic, "Transferability of neural network approaches for low-rate energy disaggregation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Aug. 2019, pp. 8330–8334.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[10] L. Wang, S. Mao, and R. M. Nelms, "Transformer for nonintrusive load monitoring: Complexity reduction and transferability," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18987–18997, Oct. 2022.

[11] D. Li et al., "Transfer learning for multi-objective non-intrusive load monitoring in smart building," *Appl. Energy*, vol. 329, Jan. 2023, Art. no. 120223.

[12] Y. Ganin et al., "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1–35, 2016.

[13] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7167–7176.

[14] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, Jul. 2015, pp. 97–105.

[15] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Proc. Computer Vision (ECCV) Workshops*, Amsterdam, The Netherlands. Cham, Switzerland: Springer, 2016, pp. 443–450.

[16] J. Lin, J. Ma, J. Zhu, and H. Liang, "Deep domain adaptation for non-intrusive load monitoring based on a knowledge transfer learning network," *IEEE Trans. Smart Grid*, vol. 13, no. 1, pp. 280–292, Jan. 2022.

[17] Y. Liu, L. Zhong, J. Qiu, J. Lu, and W. Wang, "Unsupervised domain adaptation for nonintrusive load monitoring via adversarial and joint adaptation network," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 266–277, Jan. 2022.

[18] P. Hao, L. Zhu, Z. Yan, Y. Huang, Y. Lei, and H. Wen, "Synthetic-to-real domain adaptation for nonintrusive load monitoring via reconstruction-based transfer learning," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–13, 2024.

[19] F. Zhong, Z. Shan, G. Si, A. Liu, G. Zhao, and B. Li, "Source-free domain adaptation with self-supervised learning for nonintrusive load monitoring," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–13, 2024.

[20] S. Chen, B. Zhao, M. Zhong, W. Luan, and Y. Yu, "Nonintrusive load monitoring based on self-supervised learning," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[21] G. Tanoni, L. Stankovic, V. Stankovic, S. Squartini, and E. Principi, "Knowledge distillation for scalable nonintrusive load monitoring," *IEEE Trans. Ind. Informat.*, vol. 20, no. 3, pp. 4710–4721, Mar. 2024.

[22] 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.

[23] Q. Luo, T. Yu, C. Lan, Y. Huang, Z. Wang, and Z. Pan, "A generalizable method for practical non-intrusive load monitoring via metric-based meta-learning," *IEEE Trans. Smart Grid*, vol. 15, no. 1, pp. 1103–1115, Jan. 2023.

[24] J. Gao, E. C. Kara, S. Giri, and M. Berges, "A feasibility study of automated plug-load identification from high-frequency measurements," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2015, pp. 220–224. Accessed: Nov. 27, 2022.

[25] Y. Han, Q. Gao, H. Chen, and Q. Zhao, "A few-shot learning method for nonintrusive load monitoring with V–I trajectory features," *IEEE Sensors J.*, vol. 24, no. 7, pp. 11867–11877, Apr. 2024.

[26] D. Ding, J. Li, H. Wang, and K. Wang, "Load recognition with few-shot transfer learning based on meta-learning and relational network in non-intrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 15, no. 5, pp. 4861–4876, Sep. 2024.

[27] E. Tabanelli, D. Brunelli, A. Acquaviva, and L. Benini, "Trimming feature extraction and inference for MCU-based edge NILM: A systematic approach," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 943–952, Feb. 2022.

[28] Z. Yan, D. Brunelli, D. Macii, M. Nardello, H. Wen, and D. Petri, "Appliance identification in NILM through image or signal features: A performance comparison," in *Proc. AEIT Int. Annu. Conf. (AEIT)*, Sep. 2024, pp. 1–6.

[29] J. Liang, S. K. K. Ng, G. Kendall, and J. W. M. Cheng, "Load signature study—Part I: Basic concept, structure, and methodology," *IEEE Trans. Power Del.*, vol. 25, no. 2, pp. 551–560, Apr. 2010.

[30] M. Kahl, T. Kriechbaumer, A. U. Haq, and H.-A. Jacobsen, "Appliance classification across multiple high frequency energy datasets," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Oct. 2017, pp. 147–152.

[31] M. Kahl, A. Ul Haq, T. Kriechbaumer, and H.-A. Jacobsen, "A comprehensive feature study for appliance recognition on high frequency energy data," in *Proc. 8th Int. Conf. Future Energy Syst.* New York, NY, USA: ACM, May 2017, pp. 121–131.

[32] N. Sadeghianpourhamami, J. Ruyssinck, D. Deschrijver, T. Dhaene, and C. Develder, "Comprehensive feature selection for appliance classification in NILM," *Energy Buildings*, vol. 151, pp. 98–106, Sep. 2017.

[33] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv, "Learn on source, refine on target: A model transfer learning framework with random forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1811–1824, Sep. 2017.

[34] L. Minvielle, M. Atiq, S. Peignier, and M. Mougeot, "Transfer learning on decision tree with class imbalance," in *Proc. IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2019, pp. 1003–1010.

[35] J. Gao, S. Giri, E. C. Kara, and M. Bergés, "Plaid: A public dataset of high-resoultion electrical appliance measurements for load identification research: Demo abstract," in *Proc. 1st ACM Conf. Embedded Syst. Energy-Efficient Buildings*, Jun. 2014, pp. 198–199.

[36] M. Kahl, A. U. Haq, T. Kriechbaumer, and H.-A. Jacobsen, "WHITED-a worldwide household and industry transient energy data set," in *Proc. 3rd Int. Workshop Non-Intrusive Load Monitor.*, 2016, pp. 1–4.
[37] T. Picon et al., "COOLL: Controlled on/off loads library, a public dataset of high-sampled electrical signals for appliance identification," 2016, *arXiv:1611.05803*.
[38] A. Faustine, L. Pereira, and C. Klemenjak, "Adaptive weighted recurrence graphs for appliance recognition in non-intrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 12, no. 1, pp. 398–406, Jan. 2021.
[39] Y. Zhang, H. Wu, Q. Ma, Q. Yang, and Y. Wang, "A learnable image-based load signature construction approach in NILM for appliances identification," *IEEE Trans. Smart Grid*, vol. 14, no. 5, pp. 3841–3849, Jan. 2023.
[40] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, Dec. 2007, pp. 601–608.
[41] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. 24th Int. Conf. Mach. Learn.*, Jun. 2007, pp. 193–200.
[42] E. Viciana, F. M. Arrabal-Campos, A. Alcayde, R. Baños, and F. G. Montoya, "All-in-one three-phase smart meter and power quality analyzer with extended IoT capabilities," *Measurement*, vol. 206, Jan. 2023, Art. no. 112309.
[43] Z. Wu, C. Wang, L. Xiong, R. Li, T. Wu, and H. Zhang, "A smart socket for real-time nonintrusive load monitoring," *IEEE Trans. Ind. Electron.*, vol. 70, no. 10, pp. 10618–10627, Oct. 2023.
[44] S. Kotsilitis, E. C. Marcoulaki, and E. Kalligeros, "A versatile, low-cost monitoring device suitable for non-intrusive load monitoring research purposes," *Meas., Sensors*, vol. 32, Apr. 2024, Art. no. 101081.

Zhongzong Yan is currently pursuing the Ph.D. degree with the College of Electrical and Information Engineering, Hunan University, Changsha, China.

Since October 2023, he has been a Visiting Ph.D. Student with the Department of Industrial Engineering, University of Trento, Trento, Italy. His research interests include machine learning, deep learning (DL), optimization, and control for power systems.

Pengfei Hao received the B.S. degree in electrical engineering from East China Jiaotong University, Nanchang, China, in 2022. She is currently pursuing the Ph.D. degree in electrical engineering with Hunan University, Changsha, Hunan, China.

Her current research interests include machine learning, transfer learning, and nonintrusive load monitoring.

Matteo Nardello (Member, IEEE) received the M.S. degree in electronics and telecommunications engineering and the Ph.D. degree in materials, mechatronics, and systems engineering and from the University of Trento, Trento, Italy, in 2016 and 2020, respectively.

He is currently a Research Fellow with the Department of Industrial Engineering, University of Trento. His research interests include the investigation of machine learning techniques applied to resource-constrained embedded platforms (TinyML), with a special focus on autonomous smart IoT devices, the evaluation of printed electronic solutions and applications, and the study of new architecture for indoor localization services. His other research interests encompass modeling and hardware-software codesign of original solutions to reduce power requirements of distributed wireless sensor networks.

Davide Brunelli (Senior Member, IEEE) received the M.S. (cum laude) and Ph.D. degrees in electrical engineering from the University of Bologna, Bologna, Italy, in 2002 and 2007, respectively.

He is currently an Associate Professor of electronics with the Department of Industrial Engineering, University of Trento, Trento, Italy. He has published more than 300 research papers in international conferences and journals on ultralow-power embedded systems, energy harvesting, and power management of VLSI circuits. He holds several patents and is annually ranked among the top 2% scientists according to the "Stanford World Ranking of Scientists" from 2020. His current research interests include new techniques of energy scavenging for IoT and embedded systems, the optimization of low-power and low-cost consumer electronics, and the interaction and design issues in embedded personal and wearable devices.

Dr. Brunelli is a member of several TPC conferences on the Internet of Things (IoT). He is an Associate Editor of IEEE TRANSACTIONS.

He Wen (Senior Member, IEEE) was born in Hunan, China, in 1982. He received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Hunan University, Changsha, Hunan, in 2004, 2007, and 2009, respectively.

He is currently a Full Professor with the College of Electrical and Information Engineering, Hunan University. His research interests include electrical contact reliability, wireless communications, power system harmonic measurement and analysis, power quality, and digital signal processing.

Dr. Wen is an Associate Editor of IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT and a Member of the Editorial Board of *Fluctuation and Noise Letters*.