sensors

Article

Semi-Supervised Domain Adaptation for Multi-Label Classification on Nonintrusive Load Monitoring

Cheong-Hwan Hur [1], Han-Eum Lee [1], Young-Joo Kim [2] and Sang-Gil Kang [1,*]

[1] Department of Computer Engineering, Inha University, Inha-ro 100, Nam-gu, Incheon 22212, Korea
[2] Electronics and Telecommunications Research Institute (ETRI), 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Korea
* Correspondence: sgkang@inha.ac.kr

Abstract: Nonintrusive load monitoring (NILM) is a technology that analyzes the load consumption and usage of an appliance from the total load. NILM is becoming increasingly important because residential and commercial power consumption account for about 60% of global energy consumption. Deep neural network-based NILM studies have increased rapidly as hardware computation costs have decreased. A significant amount of labeled data is required to train deep neural networks. However, installing smart meters on each appliance of all households for data collection requires the cost of geometric series. Therefore, it is urgent to detect whether the appliance is used from the total load without installing a separate smart meter. In other words, domain adaptation research, which can interpret the huge complexity of data and generalize information from various environments, has become a major challenge for NILM. In this research, we optimize domain adaptation by employing techniques such as robust knowledge distillation based on teacher–student structure, reduced complexity of feature distribution based on gkMMD, TCN-based feature extraction, and pseudo-labeling-based domain stabilization. In the experiments, we down-sample the UK-DALE and REDD datasets as in the real environment, and then verify the proposed model in various cases and discuss the results.

Keywords: nonintrusive load monitoring; transfer learning; domain adaptation; pseudo labeling; semi-supervised learning; appliance usage classification

check for updates

Citation: Hur, C.-H.; Lee, H.-E.; Kim, Y.-J.; Kang, S.-G. Semi-Supervised Domain Adaptation for Multi-Label Classification on Nonintrusive Load Monitoring. *Sensors* **2022**, *22*, 5838. https://doi.org/10.3390/s22155838

Academic Editors: Pak Kwong Chan and Alfio Dario Grasso

Received: 31 May 2022
Accepted: 3 August 2022
Published: 4 August 2022

1. Introduction

Understanding energy usage in buildings has been considered an important issue because residential and commercial power consumption account for about 60% of global energy consumption [1]. Optimized energy usage management has advantages for both suppliers and consumers of energy. From the supplier's point of view, planned consumption may be encouraged according to the frequency and pattern of use of home appliances. In addition, it is also easy for consumers to develop plans that can reduce costs through comprehensive information about device-specific operations [2]. The electricity usage profile is to install a submeter for each appliance and record instantaneous power readings, but in reality, applying this method to all devices is difficult to realize due to cost and difficulty in maintenance. Therefore, nonintrusive load monitoring (NILM) aims to disaggregate energy consumption by device. The NILM method that does not depend on submeters has shown significant efficiency in commercial and residential energy utilization and remains an important task [3].

NILM is inherently difficult because it analyzes information about the simultaneous switching or noise generation of multiple devices without attaching multiple submeters [4–6]. To solve the problem, many techniques such as dynamic time warping (DTW), matrix factorization, neuro-fuzzy modeling, and graph signal processing (GSP) have been proposed and supervised and unsupervised learning-based techniques have been studied [7–9]. Hart [10] first introduced unsupervised learning methods to decompose electrical loads through clustering. However, various techniques such as hidden Markov models (HMM) have been proposed for a while because clustering-based methods do not have training data and are difficult to predict accurate power loads. In recent years, the number of research studies on deep neural networks (DNNs) has increased rapidly with the advancement of high-end hardware devices, and the availability of data for supervised learning has increased [11]. Long short-term memory (LSTM), a representative supervised learning technology, considered NILM as a prediction problem based on time series data. Refs. [12,13] proposed a method for learning models by controlling data applied with various data sampling-based windows. Nolasco et al. [14] included multi-label procedures to increase the recognition rate for multi-loads by marking on loads at any given time and developed architectures based on convolutional neural networks (CNNs), resulting in an outstanding performance in signal detection and feature extraction. However, existing supervised learning methods for NILM still have two problems. First, there is a fundamental problem that assumes that the power usage data of real devices has a distribution similar to that of training data. It is impossible to ensure the same performance in actual situations because devices of the same type have different energy consumption depending on products and brands, noise form, intensity, physical environment, etc. [15]. To overcome this problem, training data containing all domain information must be acquired, but it is practically impossible since collecting the energy consumption of each device from different houses requires huge costs. Another problem is that, even assuming that neural network models are trained on all the data for different environments, extracting critical information is very difficult because of the vast amount of complex data [16–18]. Therefore, identifying suitable techniques that can handle the large complexity of data and generalize various domains of information is the main challenge in NILM.

To solve these problems, we consider domain adaptation [19,20]. Domain adaptation is one of the transfer learnings, which can adapt the trained model to the other domain dataset on the same task. This concept can easily be applied to the NILM system. Many researchers proposed domain adaptation systems to generalize various domain information [21,22]. Liu et al. [21] conducted a regression study to refine energy consumption by applying the most typical domain adaptation method to NILM. Since only the basic concept of domain adaptation has been applied to NILM, it has the potential to develop in various ways. Ref. [22] proposed a method that incorporates the mean teacher method into domain adaptation. Regression work is performed on the source and target domains using one model. However, this method did not show good performance in domain generalization due to its shallow model structure. To the best of our knowledge, there are no papers on classification tasks in domain adaptation studies for NILM. In this paper, we perform classification tasks for device usage detection in NILM by incorporating powerful feature information distillation based on the teacher–student structure and pseudo-labeling (PL) into domain adaptation.

The main contents of this paper are as follows:

1. We conduct the first classification study in the domain adaptation field for NILM;
2. We show performance improvements by incorporating robust feature information distillation techniques based on the teacher–student structure into domain adaptation;
3. The decision boundaries are refined through PL-based domain stabilization.

The remainder of this paper is organized as follows. Section 2 shows a brief review of related studies of NILM and domain adaptation. Section 3 introduces the proposed method. Section 4 presents the experimental setup, case study, and discussions. Finally, Section 5 concludes the paper.

2. Related Work

2.1. Nonintrusive Load Monitoring

Consider a building with m appliances and k operating power modes of each appliance for time $[1, \dots, T]$. Let $x_i = (x_i(1 \dots \dots, x_i(T))$ denote the energy consumption of the i-th device. The whole energy usage of the i-th device in sample time n can be formulated as follows:

$$x_i(n) = \left[U_1^i \ldots), \ldots, U_k^i(n)\right] \begin{bmatrix} \psi_1^i \\ \vdots \\ \psi_k^i \end{bmatrix} + \epsilon^i(n) \tag{1}$$

where ψ_k^i is the electricity consumption consumed in a particular operating mode, $\epsilon^i(n)$ denotes the measurement of background noises, and $U_k^i(n)$ is the operating On/Off $[0, 1]$ status of the i-th appliance in time n. The operating status assures the equality constraint $\sum_{j=1}^{k} U_j^i(n) = 1$ since all appliances operate in a single mode. At time n, the final energy aggregate of the house is expressed as follows:

$$x(n) = \sum_{i=1}^{m} \left[\ldots(n), \ldots, U_k^i(n)\right] \begin{bmatrix} \psi_1^i \\ \vdots \\ \psi_k^i \end{bmatrix} + \epsilon^i(n) \tag{2}$$

The goal of the NILM algorithm is to disaggregate the measured electricity usage x to generate appliance-specific energy consumption profiles [23,24]. Therefore, the final challenge is to reduce the difference between the actual measurements of the device and the disaggregated energy consumption [25].

Elafoudi et al. [7] detected the edge within the time window and used DTW to identify the unique load signature. Lin et al. [8] proposed a hybrid classification technique that combines fuzzy c-means and clustering piloting with neuro-fuzzy classification to distinguish devices that have similar load signatures. He et al. [9] handled the NILM as a single-channel blind source separation problem to perform low-complexity classification active power measurements. Based on this idea, they proposed the GSP-based NILM system to handle the large training overhead and the computational cost of the conventional graph-based method.

2.2. Domain Adaptation

Domain adaptation is an area of transfer learning [26]. In general transfer learning, a task or domain can be changed from source to target; however, in domain adaptation, the task sets the premise that only the domain is changed [19,27]. This aims to generalize the classification or regression model, which is trained on the source domain to be applied to the target domains with different distributions, since distribution disagreement between training and real data yields poor model performance. Ganin et al. [19] proposed a multi-task learning model with a class and domain classifier. The model was trained to only classify class labels, except for domain labels. For this, they introduced the gradient reversal layer (GRL) to the domain classifier. The GRL multiplies the negative constant and the gradient on the backward pass. Additionally, it makes the model remove the domain information in its feature extractor. With the advancement of deep neural networks (DNNs), the performance of domain adaptation has achieved outstanding performance in various fields [11,14,28–33]. In [34], domain adversarial training of neural networks (DANN), inspired by the generative adversarial network (GAN), laid the foundation for applying adversarial learning methodologies to domain adaptation and accomplished excellent performance. In addition, domain adaptation algorithms based on maximum mean discrepancy (MMD) between source and target were mainly studied [35–38]. In [39], Long et al. proposed a joint MMD to adjust the joint distribution. Deep domain confusion (DDC) [34] proposed a technique for using pre-trained networks by adding adaptive layers based on MMD.

Although domain adaptation is used in various fields as expressed above, the application of domain adaptation in NILM has not been researched a lot and requires advancement. In [40], Wan proposed a domain adaptation algorithm for optical character recognition (OCR), which was extended to apply it to the NILM field and produce prominent results.

Recently, Wang and Mao proposed applying a model-agnostic meta-learning (MAML)-based domain adaptation algorithm to NILM, inspired by the pre-trained model, which is heavily studied in the NLP field and outperformed the state-of-the-art deep learning-based methods. [41].

3. Semi-Supervised Domain Adaptation for Multi-Label Classification on Non-Intrusive Load Monitoring

Various deep learning models are applied to the NILM field. However, the task of segmenting the use of different devices in many houses is still a relatively new concept. To solve this problem, we propose the semi-supervised domain adaptation for multi-label classification on non-intrusive load monitoring. The overall diagram is shown in Figure 1.

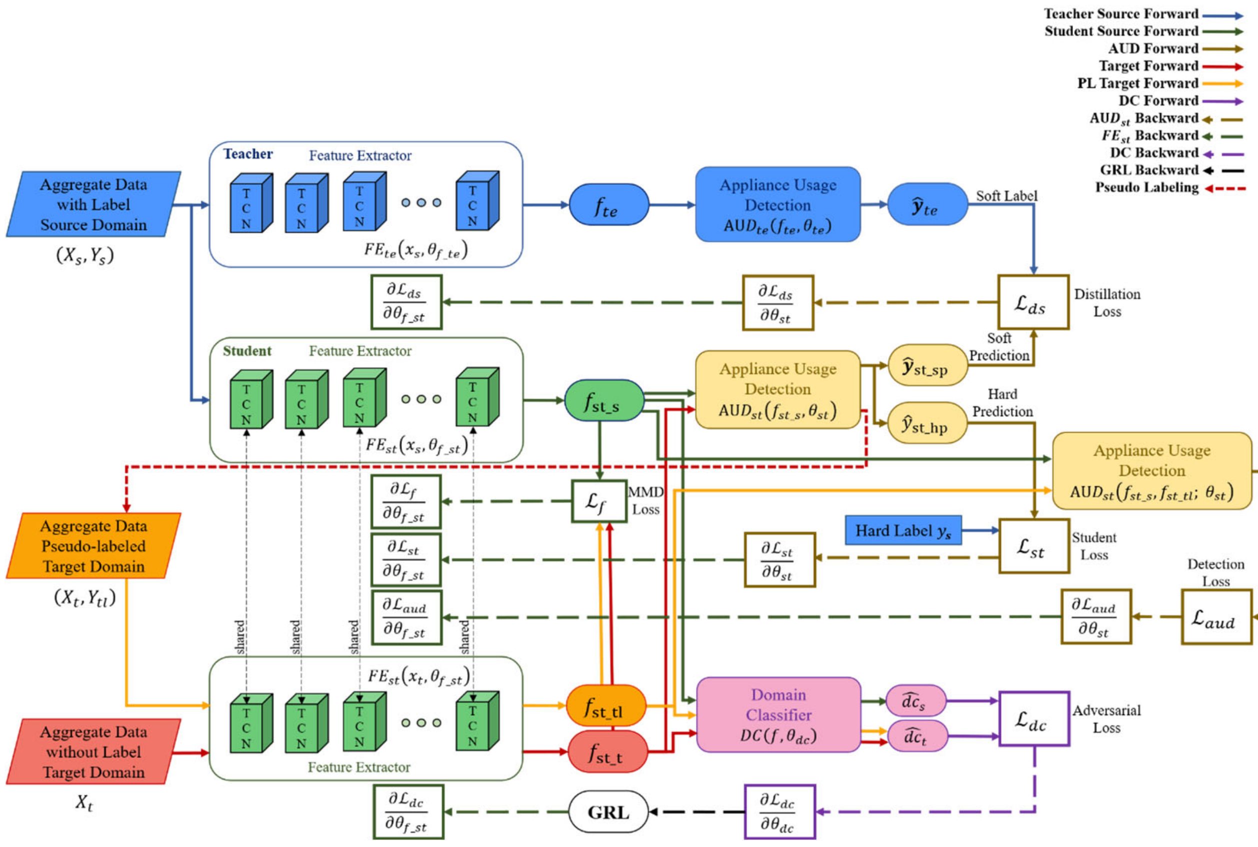
Teacher Source Forward
Student Source Forward
AUD Forward
Target Forward
PL Target Forward
DC Forward
AUD$_{st}$ Backward
FE$_{st}$ Backward
DC Backward
GRL Backward
Pseudo Labeling
Aggregate Data with Label Source Domain
(X_s, Y_s)
Teacher
Feature Extractor
TCN
$FE_{te}(x_s, \theta_{f\_te})$
f_{te}
Appliance Usage Detection $AUD_{te}(f_{te}, \theta_{te})$
$\hat{y}_{te}$
Soft Label
$\frac{\partial \mathcal{L}_{ds}}{\partial \theta_{f\_st}}$
$\frac{\partial \mathcal{L}_{ds}}{\partial \theta_{st}}$
$\mathcal{L}_{ds}$
Distillation Loss
Soft Prediction
Student
$FE_{st}(x_s, \theta_{f\_st})$
f_{st_s}
Appliance Usage Detection $AUD_{st}(f_{st\_s}, \theta_{st})$
$\hat{y}_{st\_sp}$
$\hat{y}_{st\_hp}$
Hard Prediction
Appliance Usage Detection $AUD_{st}(f_{st\_s}, f_{st\_tl}; \theta_{st})$
$\frac{\partial \mathcal{L}_{f}}{\partial \theta_{f\_st}}$
$\mathcal{L}_f$
MMD Loss
Aggregate Data Pseudo-labeled Target Domain
(X_t, Y_{tl})
$\frac{\partial \mathcal{L}_{st}}{\partial \theta_{f\_st}}$
$\frac{\partial \mathcal{L}_{st}}{\partial \theta_{st}}$
Hard Label y_s
$\mathcal{L}_{st}$
Student Loss
Detection Loss
$\frac{\partial \mathcal{L}_{aud}}{\partial \theta_{f\_st}}$
$\frac{\partial \mathcal{L}_{aud}}{\partial \theta_{st}}$
$\mathcal{L}_{aud}$
shared
$FE_{st}(x_t, \theta_{f\_st})$
f_{st_tl}
f_{st_t}
Domain Classifier $DC(f, \theta_{dc})$
$\hat{dc}_s$
$\hat{dc}_t$
$\mathcal{L}_{dc}$
Adversarial Loss
Aggregate Data without Label Target Domain
X_t
Feature Extractor
$\frac{\partial \mathcal{L}_{dc}}{\partial \theta_{f\_st}}$
GRL
$\frac{\partial \mathcal{L}_{dc}}{\partial \theta_{dc}}$

Figure 1. A detailed overall configuration diagram of the proposed semi-supervised domain adaptation for multi-label classification on nonintrusive load monitoring.

Several hypotheses are proposed in this work to apply semi-supervised domain adaptation to NILM. The first hypothesis is that the distribution of source and target domains is different. Most NILM systems are based on this hypothesis. We also use labels on the source for domain adaptation, not on the target. Second, even if the distribution of source domain data and target domain data is different in NILM, it is assumed that the same device has domain-independent common characteristics regardless of the domain. Because in motor devices, lagging current with slow current flow occurs, which results in a low power factor. Additionally, capacitor devices generate leading current with fast current flow, which results in a high-power factor. The power factor is the ratio of active power to apparent power regardless of the magnitude of power consumption. In other words, if two different houses use the same electronic devices (e.g., refrigerator, TV, etc.) from different manufacturers, it is assumed that there is a common usage pattern even if the power consumption is different.

The proposed method consists of three main steps, shown in Figure 2. In the knowledge distillation stage, high-level knowledge is distilled into the student network (SN) by a temporal convolutional network (TCN) [42]-based teacher network (TN) [43] trained using labeled source data. Domain-dependent features vary depending on the domain, and domain-independent features are constant regardless of the domain. In the next step, we perform a robust domain adaptation that allows us to extract only domain-independent features to adapt source and target data to neural networks regardless of domain. Appliance usage detection classifies devices from source domain data. Additionally, domain classifiers are trained with GRL to prevent classification for source and target domains. As a result, feature extractors can extract robust domain-independent features that enable device usage classification regardless of domain. In the domain stabilization step in Figure 2, we stabilize the domain through PL-based fine-tuning. First, domain-independent features of target data are extracted from the feature extractor and then pseudo-labeled based on the source domain label in appliance usage detection. Since all the target data cannot be pseudo-labeled, it is partially pseudo-labeled. Therefore, the target data consists of pseudo-labeled data and unlabeled data. Secondary domain adaptation is performed based on the enhanced target domain data and domain-independent features extracted through robust distillation. The network performance is stabilized and improved through the advantages of low-density separation between classes and entropy regularization. Details of each part of the proposed framework are in the subsections.

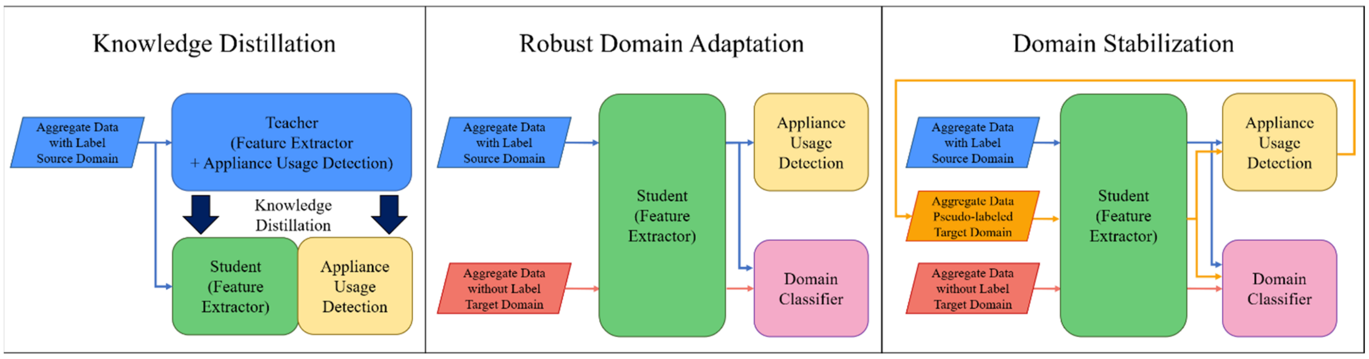
Knowledge Distillation
Aggregate Data with Label Source Domain
Teacher (Feature Extractor + Appliance Usage Detection)
Knowledge Distillation
Student (Feature Extractor)
Appliance Usage Detection
Robust Domain Adaptation
Aggregate Data with Label Source Domain
Aggregate Data without Label Target Domain
Student (Feature Extractor)
Appliance Usage Detection
Domain Classifier
Domain Stabilization
Aggregate Data with Label Source Domain
Aggregate Data Pseudo-labeled Target Domain
Aggregate Data without Label Target Domain
Student (Feature Extractor)
Appliance Usage Detection
Domain Classifier

Figure 2. Step-by-step flowchart of the proposed method.

3.1. Network Architecture

The goal of this section is to build a semi-supervised domain adaptation model that can estimate the target domain label Y_t using labeled source data $(X_s,\ Y_s)$ and target data X_t. As shown in Figure 2, the model includes three parts: knowledge distillation, robust domain adaptation, and domain stabilization. Details of the network structure are as follows:

(1) Knowledge distillation: knowledge is distilled using a TCN feature extraction-based teacher–student network to receive robust domain-independent features of source data. TCN is an extended time-series data modeling structure in CNN. It provides better performance than typical time-series deep learning models such as LSTM because it has a much longer and more effective memory without a gate mechanism. The TCN consists of several residual blocks, and this block consists of a dilated casual convolution operation O. For input $x \in \mathbb{R}^n$ and filter $ft\ : \{0,1\ldots,k-1\} \rightarrow \mathbb{R}$, O at point s is defined by Equation (3).

$$O(s) = (x *_d ft)(s) = \sum_{i=0}^{k-1} ft(i)\cdot x_{s-d\cdot i} \tag{3}$$

where d is the dilated factor, $*_d$ is $*_d$-dilated convolution, k is the filter size, and $s - d\cdot i$ is the past value. However, as the network depth increases, performance decreases rapidly due to overfitting. However, as the network depth increases, performance decreases quickly due to overfitting. Resnet's key concept, namely, residual mapping, can solve this problem. The TCN residual block includes two layers of dilated casual convolution based on the ReLU activation function, weighted normalization, and dropout. The 1×1 convolution layer on the TCN ensures that the input and output are the same size. The output of the transformation T of the time series data in the TCN dual block is added to the identity mapping of the input x and expressed as follows:

$$R(s) = T(x, \theta) + x \tag{4}$$

where θ means the set of parameters of the network. It has already been demonstrated that this concept of residual block improves network performance by learning modifications to identity mapping rather than overall transformations. Based on this, it is possible to build a deep TCN network by stacking multiple TCN residual blocks. Assuming that x_I is the input of the I-th block, the network forward propagation from the I-th block to the $I + n$-th block can then be formulated as follows:

$$x_{I+n} = x_I + \sum_{i=I}^{I+n-1} T(x_i, \theta_i) \tag{5}$$

where, x_I is I-th block, θ_i is the parameter set of the i-th block. Therefore, the feature extractor $FE_{te}(x_s,\ \theta_{f_{te}})$ of the TN is defined as follows:

$$FE_{te}(x_s,\ \theta_{f\_te}) = x_s + \sum_{i=0}^{k-1} T(x_{s\_i}, \theta_{f\_te\_i}) \tag{6}$$

where the number of layers is k, x_s is source data, $\theta_{f_{te}}$ is the parameter set of TN, x_{s_i} is i^{th} source data, $\theta_{f_{te_i}}$ is the parameter set of i^{th} block in the TN. Additionally, the feature extractor $FE_{st}(x_s,\ \theta_{f_{st}})$ of SN can be defined as follows:

$$FE_{st}(x_s,\ \theta_{f\_st}) = x_s + \sum_{i=0}^{l-1} T(x_{s\_i}, \theta_{f\_st\_i}) \tag{7}$$

where l is the number of layers, $\theta_{f_{st}}$ is the parameter set of SN, $\theta_{f_{st_i}}$ is the parameter set of i^{th} block in SN. Based on f_{te} extracted from Equation (6), the TN must extract soft label information for transferring knowledge to the SN through appliance usage detection, which consists of a fully connected layer. The output $\hat{y}_{te}$ of the TN is defined as follows:

$$\hat{y}_{te} = Softmax_{with\ T}(AUD_{te}(f_{te},\ \theta_{te})_i) = \frac{\frac{e^{AUD_{te}(f_{te},\ \theta_{te})_i}}{T}}{\sum_{j=1}^{K} \frac{e^{AUD_{te}(f_{te},\ \theta_{te})_j}}{T}} \tag{8}$$

where te refers to the TN, $\hat{y}_{te}$ is a predicted classification label of x_s in the TN, T is a temperature parameter, $Softmax_{with\ T}$ *is* a $Softmax$ function with a temperature parameter. θ_{te} is the parameter set of AUD_{te}, $AUD_{te}(f_{te},\ \theta_{te})_i$ is the elements of output vector of AUD_{te}, i refers to i^{th} element, K is the number of elements of the output vector. Maximize the benefits of soft label values for knowledge distillation by using temperature parameters to prevent information loss in $Softmax$ output. The estimated soft label $\hat{y}_{te}$ is compared to the soft prediction $\hat{y}_{st_{sp}}$ of SN and is used as a distillation loss in network training. $\hat{y}_{st_{sp}}$ is obtained as follows:

$$\hat{y}_{st\_sp} = Softmax_{with\ T}(AUD_{st}(f_{st\_s},\ \theta_{st})_i) = \frac{\frac{e^{AUD_{st}(f_{st\_s},\ \theta_{st})_i}}{T}}{\sum_{j=1}^{K} \frac{e^{AUD_{st}(f_{st\_s},\ \theta_{st})_j}}{T}} \tag{9}$$

where st refers to SN, $\hat{y}_{st_{sp}}$ is a predicted classification label of x_s in the SN and a soft prediction value of SN, θ_{st} is the parameter set of AUD_{st}, and $AUD_{st}(f_{st_s},\ \theta_{st})_i$

is the i-th element of the output vector of AUD_{st}. The classification performance of SN should be evaluated along with knowledge distillation. The performance can be evaluated by comparing the hard prediction $\hat{y}_{st_{hp}}$ of SN with the ground truth y_s of the source domain data. $\hat{y}_{st_{hp}}$ is obtained as follows:

$$\hat{y}_{st\_hp} = Softmax(AUD_{st}(f_{st\_s},\ \theta_{st})_i) = \frac{e^{AUD_{st}(f_{st\_s},\ \theta_{st})_i}}{\sum_{j=1}^{K} e^{AUD_{st}(f_{st\_s},\ \theta_{st})_j}} \quad (10)$$

where $\hat{y}_{st_{hp}}$ is a predicted classification label of x_s in the SN and is used as a hard prediction value of SN. In Equation (10), the temperature parameter is not used.

(2) Robust domain adaptation: domain adaptation is performed with robust features extracted with knowledge distillation to obtain domain-independent features. Domain adaptation consists of the following three stages: feature extractor, domain classifier, and appliance usage detection. First, a feature extractor FE_{st} of SN is used. A feature extractor $FE_{st}\left(x_s,\ \theta_{f_{st}}\right)$ of the source data and an $FE_{st}\left(x_t,\ \theta_{f_{st}}\right)$ of the target data share a parameter set. Models learned with only source data are difficult to express with target data. To adapt the target domain data representation to FE_{st}, the model learns the feature distribution difference between the two domains using MMD and minimizes it. The MMD distance is obtained as follows:

$$MMD(X_s, X_t) = \| \frac{1}{n_s} \sum_{i=1}^{n_s} \varphi\left(x_s^i\right) - \frac{1}{n_t} \sum_{j=1}^{n_t} \varphi\left(x_t^j\right) \|_{\mathcal{H}} \quad (11)$$

where φ is a feature space mapping function that turns the original feature space into the reproducing kernel Hilbert space H. Further descriptions of the kernel are given in the following subsection. The domain classifier $DC(f,\ \theta_{dc})$ learns by setting the ground truth values of the source domain data and the target domain data $dc_s = 0$ and $dc_t = 1$, respectively, to separate the domain-independent features from the feature extractor. $DC(f,\ \theta_{dc})$ has an output $\hat{dc}_s$ for source domain data and an output $\hat{dc}_t$ for target domain data. The two outputs are defined as follows:

$$\hat{dc}_s = Softmax(DC(f_{st\_s},\ \theta_{dc})) \quad (12)$$

$$\hat{dc}_t = Softmax(DC(f_{st\_t},\ \theta_{dc})) \quad (13)$$

where f_{st_s} is the source domain feature, f_{st_t} is the target domain feature and θ_{dc} is the parameter set of DC. $\hat{dc}_s$ and $\hat{dc}_t$ values between 0 and 1. DC can obtain domain-independent features from FE_{st} by learning that the source and target domains cannot be classified. Appliance usage detection uses the AUD_{st} of SN. AUD_{st} verifies classification performance using source data as input domain-independent features. The prediction of device usage detection can be obtained using Equation (10). In network inference, prediction of the target domain may be obtained using Equation (14).

$$\hat{y}_t = Softmax(AUD_{st}(f_{st\_t},\ \theta_{st})) \quad (14)$$

where $\hat{y}_t$ is the prediction of target data. Detection performance for target domain data is evaluated by comparing $\hat{y}_t$ with ground truth y_t of target domain data.

(3) Domain stabilization: The target domain data is pseudo-labeled with AUD_{st} to enhance the data, thereby stabilizing the domain and improving the performance of the network. First, the feature $f_{\mathrm{st_t}}$ of the target domain data x_t is input to the AUD_{st}. If $Softmax(AUD_{st}(f_{st_t},\ \theta_{st}))$ is obtained through Equation (14), PL is generated as a prediction value having the highest probability among $Softmax$ values. However, if the probability is lower than the threshold, the data is not pseudo-labeled. The threshold is obtained experimentally. Domain stabilization consists of three steps, such as feature extraction and domain classifier. Appliance usage detection uses

the following three types of data: source data (X_s, Y_s), pseudo-labeled target data (X_t, Y_{tl}), and unlabeled target data X_t. For feature extraction, $f_{\mathrm{st_s}}$, $f_{\mathrm{st_{tl}}}$ and $f_{\mathrm{st_t}}$ are output through FE_{st}. DC has no change in the domain, $f_{\mathrm{st_s}}$, $f_{\mathrm{st_{tl}}}$ and $f_{\mathrm{st_t}}$ are classified as inputs, as in Equations (12) and (13). The appliance usage detection performs $\mathrm{AU}D_{st}\left(f_{st_s}, f_{st_{tl}};\ \theta_{st}\right)$.

3.2. Network Losses

We carefully design network losses to obtain domain-independent features from feature distributions. We divide the network loss into the following four stages: knowledge distillation loss, feature distribution difference loss, domain classification loss, and appliance usage detection loss.

(1) Knowledge distillation loss: As shown in Figure 1, the knowledge distillation phase loss is the sum of the distillation loss $\mathcal{L}_{ds}$ and student loss $\mathcal{L}_{ds}$. $\mathcal{L}_{ds}$ is to include the difference in the classification results of the TN and the SN in the loss. $\mathcal{L}_{ds}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{ds} &= 2\alpha T^2 \mathcal{L}_{ce}\left(\frac{\frac{e^{AUD_{te}(f_{te},\ \theta_{te})_i}}{T}}{\sum_{j=1}^{K} \frac{e^{AUD_{te}(f_{te},\ \theta_{te})_j}}{T}}, \frac{\frac{e^{AUD_{st}(f_{st_s},\ \theta_{st})_i}}{T}}{\sum_{j=1}^{K} \frac{e^{AUD_{st}(f_{st_s},\ \theta_{st})_j}}{T}}\right) \\ &= 2\alpha T^2 \mathcal{L}_{ce}(Softmax_{with\ T}(AUD_{te}(f_{te},\ \theta_{te})_i), Softmax_{with\ T}(AUD_{st}(f_{st_s},\ \theta_{st})_i)) \\ &= 2\alpha T^2 \mathcal{L}_{ce}\left(\hat{y}_{te}, \hat{y}_{st\_sp}\right)\end{aligned} \tag{15}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss and α is the learning rate. The cross-entropy loss is calculated about teacher and student output. If the classification results of the teacher and the student are the same and distillation is good, $\mathcal{L}_{ds}$ takes a small value. Additionally, $\mathcal{L}_{st}$ means the cross-entropy loss of the classification of SN. $\mathcal{L}_{st}$ is defined as follows:

$$\begin{aligned}\mathcal{L}_{st} &= (1-\alpha)\mathcal{L}_{ce}\left(\frac{e^{AUD_{st}(f_{st_s},\ \theta_{st})_i}}{\sum_{j=1}^{K} e^{AUD_{st}(f_{st_s},\ \theta_{st})_j}}, y_s\right) = (1-\alpha)\mathcal{L}_{ce}(Softmax(AUD_{st}(f_{st_s},\ \theta_{st})_i), y_s) \\ &= (1-\alpha)\mathcal{L}_{ce}\left(\hat{y}_{st\_hp}, y_s\right)\end{aligned} \tag{16}$$

Even in a network with relatively fewer parameters than in the TN, $\mathcal{L}_{st}$ is also reduced when $\mathcal{L}_{ds}$ is smaller, so it shows good feature extraction and classification performance.

(2) Feature distribution difference loss: As shown in Figure 1, the feature distribution difference loss is MMD Loss [44] $\mathcal{L}_f$. $\mathcal{L}_f$ estimates the difference between the feature distribution of the source domain data X_s and the feature distribution of the target domain data X_t through MMD. $\mathcal{L}_f$ is generally defined as follows:

$$\begin{aligned}\mathcal{L}_f(f_{st\_s}, f_{st\_t}) &= MMD^2(f_{st\_s}, f_{st\_t}) = \|\mathbb{E}_{X_s \sim f_{st\_s}}\varphi(X_s) - \mathbb{E}_{X_t \sim f_{st\_t}}\varphi(X_t)\|_{\mathcal{H}}^2 \\ &= <\mathbb{E}_{X_s \sim f_{st_s}}\varphi(X_s), \mathbb{E}_{X'_s \sim f_{st_s}}\varphi(X'_s)>_{\mathcal{H}} + <\mathbb{E}_{X_t \sim f_{st_t}}\varphi(X_t), \mathbb{E}_{X'_t \sim f_{st_t}}\varphi(X'_t)>_{\mathcal{H}} \\ &\quad -2<\mathbb{E}_{X_s \sim f_{st\_s}}\varphi(X_s), \mathbb{E}_{X_t \sim f_{st\_t}}\varphi(X_t)>_{\mathcal{H}}\end{aligned} \tag{17}$$

For the mapping function φ of Equation (17), we use kernel tricks because computational resources are required too much to obtain all the moments. We utilize the Gaussian kernel as shown in Equation (18).

$$gk(x, y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \tag{18}$$

where gk is the Gaussian kernel. In the Equation (18), Taylor's development of the exponential develops as in Equation (19).

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots \tag{19}$$

Since Equation (19) contains all the moments for x, we use the Gaussian kernel. G$k(x, y)$ is derived as Equation (20).

$$gk(x, y) = <\varphi(x), \varphi(y)>_{\mathcal{H}} \tag{20}$$

When Equation (15) is organized using Equation (20), $\mathcal{L}_f$ is re-formulated as shown in Equation (21).

$$\begin{aligned}\mathcal{L}_f(f_{st\_s}, f_{st\_t}) = &<\mathbb{E}_{X_s \sim f_{st\_s}}\varphi(X_s), \mathbb{E}_{X'_s \sim f_{st\_s}}\varphi(X'_s)>_{\mathcal{H}} + <\mathbb{E}_{X_t \sim f_{st\_t}}\varphi(X_t), \mathbb{E}_{X'_t \sim f_{st\_t}}\varphi(X'_t)>_{\mathcal{H}} \\ &-2<\mathbb{E}_{X_s \sim f_{st_s}}\varphi(X_s), \mathbb{E}_{X_t \sim f_{st_t}}\varphi(X_t)>_{\mathcal{H}} \\ &= \mathbb{E}_{X_s X'_s \sim f_{st_s}} gk(X_s, X'_s) + \mathbb{E}_{X_t X'_t \sim f_{st_t}} gk(X_t, X'_t) - 2\mathbb{E}_{X_s \sim f_{st\_s}, X_t \sim f_{st\_t}} gk(X_s, X_t)\end{aligned} \tag{21}$$

(3) Domain classification loss: As shown in Figure 1, the domain classification loss $\mathcal{L}_{dc}$ is related to FE_{st} and DC. $DC(f,\ \theta_{dc})$ is modeled so that the source domain and the target domain cannot be distinguished. To minimize the distribution difference between $f_{\mathrm{st_s}}$ and $f_{\mathrm{st_t}}$, the loss of $DC(f,\ \theta_{dc})$ should be maximized. Using $\hat{dc}_s$ and $\hat{dc}_t$ of $DC(f,\ \theta_{dc})$, cross-entropy loss as a binary classifier-based $\mathcal{L}_{dc}$ can be obtained as Equation (22).

$$\mathcal{L}_{dc}(x_s, x_t; \theta_{f\_st}, \theta_{dc}) = -\sum_{i=1}^{sn}\left[\log(1 - \hat{dc}_s^i) + \log(\hat{dc}_t^i)\right] \tag{22}$$

where, sn is the sample number of mini-batch.

(4) Appliance usage detection loss: as shown in Figure 1, the appliance usage detection loss uses $\mathcal{L}_{st}$ in the domain adaptation phase and $\mathcal{L}_{aud}$ in the robust domain adaptation phase. Since both losses are applied to the same AUD_{st}, the same loss equation is formularized as in Equations (23) and (24).

$$\mathcal{L}_{st} = \mathcal{L}_{ce}(Softmax(AUD_{st}(f_{st\_s},\ \theta_{st})_i), y_s) \tag{23}$$

$$\mathcal{L}_{aud} = \mathcal{L}_{ce}(Softmax(AUD_{st}(f_{st\_s},\ \theta_{st})_i), y_s) + \mathcal{L}_{ce}\big(Softmax\big(AUD_{st}(f_{st\_tl},\ \theta_{st})_i\big), y_t\big) \tag{24}$$

Each neural network is learned by differentiating loss with corresponding weights, as shown in the dotted line in Figure 1.

3.3. Training Strategy

According to the network loss discussed above, the final optimization objective can be expressed as follows:

$$(\theta^*_{f\_st}, \theta^*_{st}, \theta^*_{dc}) = \mathrm{argmin}[\mathcal{L}_{aud} + \mathcal{L}_{dc} + \mathcal{L}_f] \tag{25}$$

Assuming that θ_{f_te}, θ_{te} are pre-learned high-performance networks, they do not perform additional learning to reduce network loss. When we learn $\mathcal{L}_{dc}$ of Equation (22), we apply the gradient reversal layer (GRL) to learn in a direction that fails to classify domains. The pseudo-code of the proposed model is summarized in Algorithm 1.

Algorithm 1: Parameter optimization procedure of the proposed method.

```
Input: The source domain data $(x_s, y_s)$, The target domain data $(x_t)$ with M total samples, respectively.
Output: The optimized parameters $(\theta^*_{f_{st}}, \theta^*_{st}, \theta^*_{dc})$
# Knowledge Distillation Phase
for m = 0 to epochs do
for $n$ to minibatch do
    #Foward propagation
    Teacher: $f_{te} \leftarrow FE_{te}(x_s,\ \theta_{f\_te})$, $\hat{y}_{te} \leftarrow AUD_{te}(f_{te},\ \theta_{te})$
    Student: $f_{\mathrm{st\_s}} \leftarrow FE_{st}(x_s,\ \theta_{f\_st})$, $\hat{y}_{\mathrm{st\_sp}} \leftarrow \mathrm{AUD}_{st}(f_{st\_s},\ \theta_{st})$, $\hat{y}_{st\_hp} \leftarrow \mathrm{AUD}_{st}(f_{st\_s},\ \theta_{st})$
    $\mathcal{L}_{ds} \leftarrow (\hat{y}_{te}, \hat{y}_{\mathrm{st\_sp}}) = 2\alpha T^2 \mathcal{L}_{ce}(\hat{y}_{te}, \hat{y}_{st\_sp})$, $\mathcal{L}_{st} \leftarrow (\hat{y}_{st\_hp}, y_s) = (1-\alpha)\mathcal{L}_{ce}(\hat{y}_{st\_hp}, y_s)$
    $\mathcal{L} \leftarrow \mathcal{L}_{ds} + \mathcal{L}_{st}$
    #Back propagation
    $\theta_{f\_st},\ \theta_{st} \leftarrow \mathrm{Adam}(\nabla_\theta \mathcal{L},\ \theta_{f\_st},\ \theta_{st})$
end for
end for
# Domain Adaptation Phase
for m = 0 to epochs do
for $n$ to minibatch do
    #Foward propagation
    Source: $f_{\mathrm{st\_s}} \leftarrow FE_{st}(x_s,\ \theta_{f\_st})$, $\hat{dc}_s \leftarrow DC(f_{\mathrm{st\_s}},\ \theta_{dc})$, $\hat{y}_{st\_hp} \leftarrow AUD_{st}(f_{st\_s},\ \theta_{st})$
    Target: $f_{\mathrm{st\_t}} \leftarrow FE_{st}(x_t,\ \theta_{f\_st})$, $\hat{dc}_t \leftarrow DC(f_{\mathrm{st\_t}},\ \theta_{dc})$
    $\mathcal{L}_f \leftarrow (f_{st\_s}, f_{st\_t}) = \mathbb{E}_{X_s X'_s \sim f_{st_s}} gk(X_s, X'_s) + \mathbb{E}_{X_t X'_t \sim f_{st_t}} gk(X_t, X'_t) - 2\mathbb{E}_{X_s \sim f_{st_s}, X_t \sim f_{st_t}} gk(X_s, X_t)$,
    $\mathcal{L}_{dc} \leftarrow (x_s, x_t; \theta_{f\_st}, \theta_{dc}) = -\sum_{i=1}^{sn} \left[\log(1-\hat{dc}^i_s) + \log(\hat{dc}^i_t)\right]$,
    $\mathcal{L}_{st} \leftarrow (f_{st\_s},\ \theta_{st}) = \mathcal{L}_{ce}(Softmax(AUD_{st}(f_{st\_s},\ \theta_{st})), y_s)$
    $\mathcal{L} \leftarrow \mathcal{L}_f + \mathcal{L}_{dc} + \mathcal{L}_{st}$
    #Back propagation
    $\theta_{f\_st}, \theta_{st},\ \theta_{dc} \leftarrow \mathrm{Adam}(\nabla_\theta \mathcal{L},\ \theta_{f_{st}},\ \theta_{st},\ \theta_{dc})$
end for
end for
# Robust Domain Adaptation Phase
#Pseudo labeling
$f_{\mathrm{st\_t}} \leftarrow FE_{st}(x_t,\ \theta_{f\_st})$, $y_{tl} \leftarrow AUD_{st}(f_{\mathrm{st\_t}},\ \theta_{st})$
for m = 0 to epochs do
for $n$ to minibatch do
    #Foward propagation
    Source: $f_{\mathrm{st\_s}} \leftarrow FE_{st}(x_s,\ \theta_{f\_st})$, $\hat{dc}_s \leftarrow DC(f_{\mathrm{st\_s}},\ \theta_{dc})$, $\hat{y}_{st\_hp} \leftarrow AUD_{st}(f_{st\_s},\ \theta_{st})$
    Target: $f_{\mathrm{st\_t}} \leftarrow FE_{st}(x_t,\ \theta_{f\_st})$, $\hat{dc}_t \leftarrow DC(f_{\mathrm{st\_t}},\ \theta_{dc})$
    Pseudo Target: $f_{\mathrm{st\_tl}} \leftarrow FE_{st}(x_t,\ \theta_{f\_st})$, $\hat{y}_{st\_tl} \leftarrow AUD_{st}(f_{\mathrm{st\_tl}},\ \theta_{st})$
    $\mathcal{L}_f \leftarrow (f_{st\_s}, f_{st\_t}) = \mathbb{E}_{X_s X'_s \sim f_{st_s}} gk(X_s, X'_s) + \mathbb{E}_{X_t X'_t \sim f_{st_t}} gk(X_t, X'_t) - 2\mathbb{E}_{X_s \sim f_{st_s}, X_t \sim f_{st_t}} gk(X_s, X_t)$,
    $\mathcal{L}_{dc} \leftarrow (x_s, x_t; \theta_{f\_st}, \theta_{dc}) = -\sum_{i=1}^{sn} [\log(1-\hat{dc}^i_s) + \log(\hat{dc}^i_t)]$,
    $\mathcal{L}_{aud} \leftarrow (f_{st_s}, f_{st_{tl}};\ \theta_{st})$
          $= \mathcal{L}_{ce}(Softmax(AUD_{st}(f_{st\_s},\ \theta_{st})_i), y_s) + \mathcal{L}_{ce}(Softmax(AUD_{st}(f_{st\_tl},\ \theta_{st})_i), y_{tl})$
    $\mathcal{L} \leftarrow \mathcal{L}_f + \mathcal{L}_{dc} + \mathcal{L}_{aud}$
    #Back propagation
    $\theta_{f\_st}, \theta_{st},\ \theta_{dc} \leftarrow \mathrm{Adam}(\nabla_\theta \mathcal{L},\ \theta_{f_{st}},\ \theta_{st},\ \theta_{dc})$
end for
end for
    $(\theta^*_{f_{st}}, \theta^*_{st}, \theta^*_{dc})$
```

4. Experiments

4.1. Data Preparation

4.1.1. Dataset

Two publicly available NILM datasets, UK-DALE [45] and REDD [46], were used for performance evaluation. UK-DALE collected smart meter data from five UK buildings, with sampling resolution and corresponding device-level consumption of 1 s and 6 s, respectively, for the total home consumption. The data set was recorded for 39–600 days. REDD was collected from six actual buildings in the United States. The measurement period is between 3 and 19 days, consisting of appliance-level energy consumption data sampled every 3 s and total measurements sampled every 1 s.

This article analyzes the use of the following five representative house appliances: dishwasher (DW), refrigerator (FG), kettle (KT), microwave (MV), and washing machine (WM). Since REDD does not have kettle data, NILM uses four house appliances, excluding kettles. The selected electronic products exhibit various power patterns and power levels. FG generally consumes constant and low power; however, other power consumption is very high power. DW and WM have very complex power usage patterns and power strengths. MV and KT have very monotonous power usage patterns. These five home appliances are generally designated as representative research targets because they account for most of the power consumption in the building.

In UK-DALE, House 1 uses data collected for 74 days from 1 January 2013 to 15 March 2013, and House 2 uses data collected for 74 days from 1 June 2013 to 13 August 2013. In REDD, House 1 and House 3 use data collected over 39 days from 17 April 2011 to 25 May 2011.

4.1.2. Data Preprocessing

Each power consumption of the two datasets is downsampled to 1 min and then preprocessed for missing values using linear interpolation. Each house appliance is classified as ON (1) if the power consumption (for 15 min) is greater than the experimentally set threshold and is classified as OFF (0) if it is less than the threshold. Figures 3 and 4 show the power usage of each home appliance in UK-DALE and REDD, respectively, and the thresholds for determining the ON event accordingly. The threshold was experimentally determined to be sure to include all ON states. However, since the FG is continuously operating, the threshold was determined based on the state in which the motor was running. Table 1 shows the exact threshold value of each home appliance and the number of ON events determined accordingly. The split ratio of training, validation, and test data are 6:2:2. The sliding window is used for around 15 min based on the ON event. A sliding window W with a stride length l_s runs the sequence forward to obtain an input sample $x = (x_1, x_2, \dots, x_W)$. For each i^{th} window, the network has $y^i = (y^i_{DW}, y^i_{FG}, y^i_{KT}, y^i_{MV}, y^i_{WM})$ as output power.

4.2. Experimental Setup

4.2.1. Implementation Configuration

To obtain an input sample, W is set to 15, and l_s is set to 15 so that data is non-overlapped. In the TN, there are 3.2 times more parameters in the feature extractor and 1.6 times more parameters in the fully connected layer compared to the SN. The epochs in the robust domain adaptation and the domain stabilization phases are not set separately because the early stopping parameter automatically controls learning. The basic structure of SN is cited in [20]. The TN is experimentally determined to have a structure approximately twice as large as the SN. The mini-batch size is set to the maximum value applicable in the experimental environment. The decaying learning rate is used to determine the optimal value by repeatedly reducing it by one-third. The parameters of the proposed model are listed in Table 2.

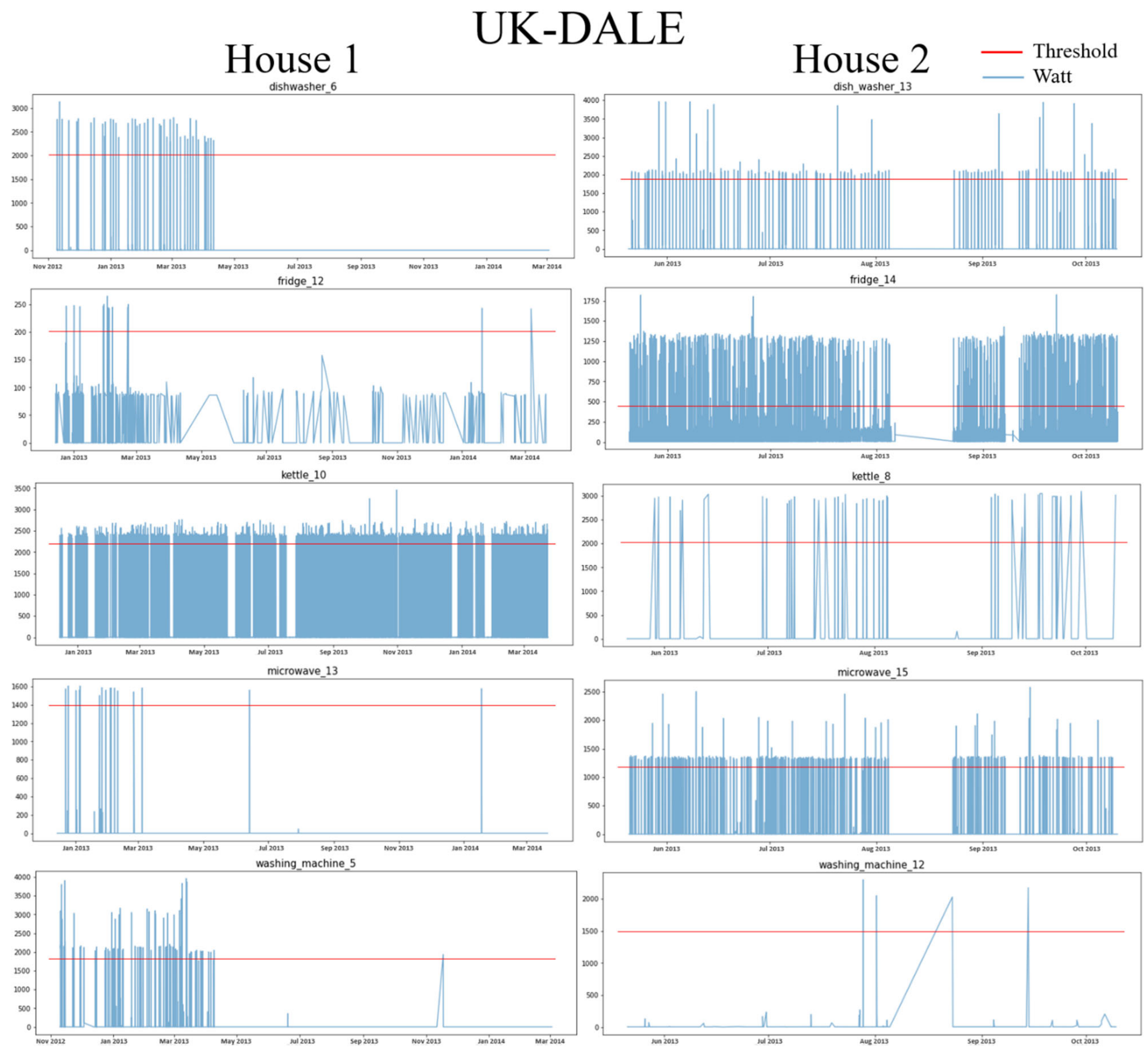

Figure 3. Power Usage and ON Thresholds for House 1 and House 2 in the UK-DALE dataset.

All experimental models were modified and executed in Python 3.6 [47] and the Pytorch framework [48], and learning and inferencing used the NVIDIA RTX 2070 SUPER.

4.2.2. Ablation Study Methods

Our model consists of the following four main techniques: TCN, gkMMD, teacher–student (TS) structure, and PL. We introduce an ablation study on five methods to investigate how individual components influence performance improvements in the proposed model.

1. Baseline: Typical domain adaptation method with BiLSTM-based feature extractors;
2. TCN-DA: Domain adaptation method with TCN-based feature extractor;
3. gkMMD-DA: Domain adaptation method with Gaussian kernel trick-based MMD Loss in baseline;
4. TS-DA: A domain adaptation method for extracting features based on the robust knowledge distillation of the teacher–student structure. The feature extractor of SN used BiLSTM, such as the baseline, and the feature extractor of TN used BiLSTM, which is four times the size of the student;
5. PL-DA: How to perform domain optimization with pseudo labeling on baseline method

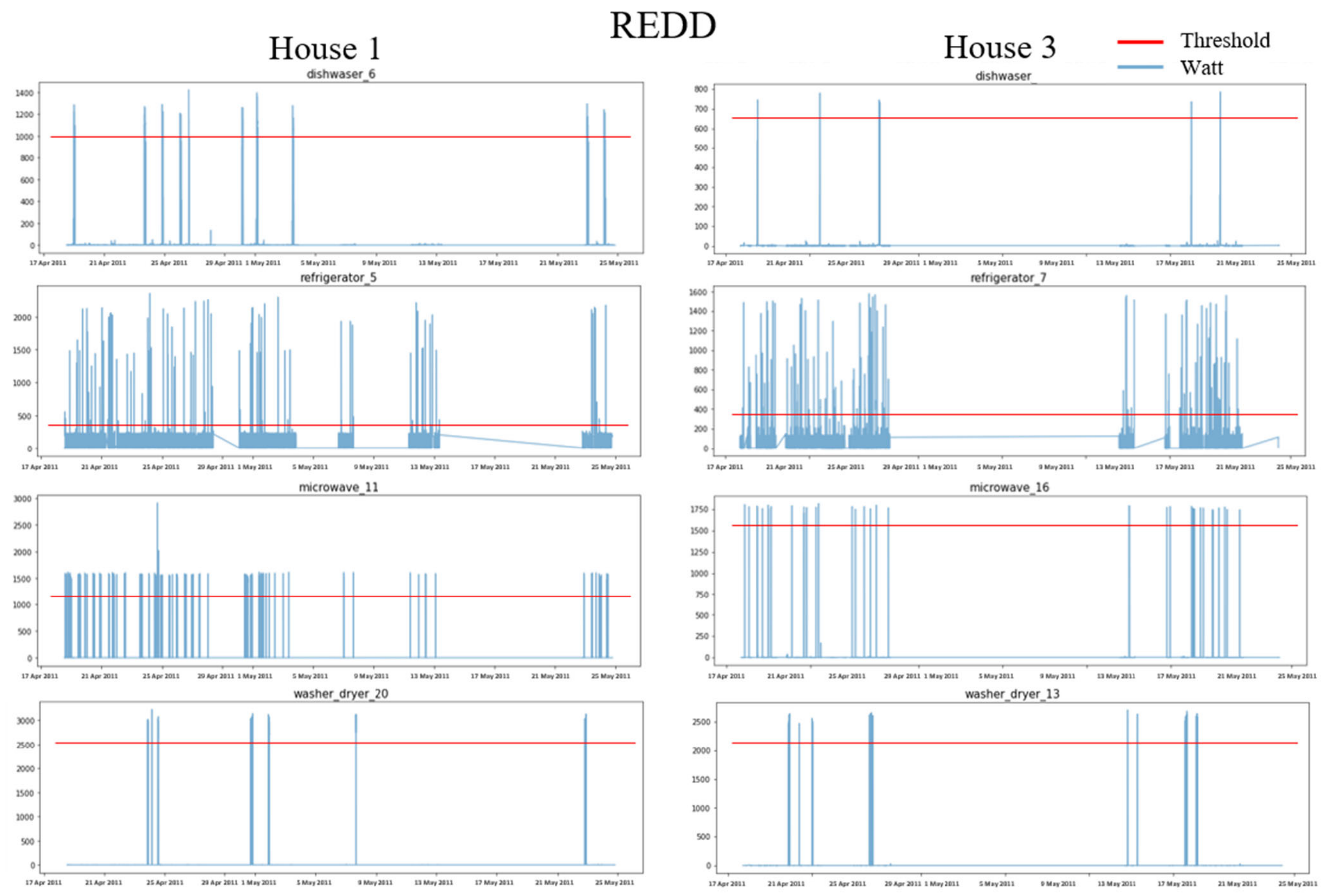

Figure 4. Power usage and ON thresholds for House 1 and House 3 in the REDD dataset.

Table 1. ON threshold and the number of ON events in UK-DALE and REDD datasets.

	UK-DALE				**REDD**			
	House 1		**House 2**		**House 1**		**House 3**	
Appliance	**Threshold**	**The Number of ON Event**	**Threshold**	**The Number of ON Event**	**Threshold**	**The Number of ON Event**	**Threshold**	**The Number of ON Event**
DW	2000	4431	1800	3236	1000	6712	650	2934
FG	250	2441	400	5291	400	2944	350	3344
KT	2200	4495	2000	1694	-	-	-	-
MV	1400	1242	1200	4218	1200	4809	1600	1327
WM	1800	4980	1500	1524	2500	4796	2200	5764

4.2.3. Evaluation Metrics

Performance evaluation uses the F1-score, a general metric. The F1-score is derived as shown in Equation (26).

$$F1(TP, FP, FN) = \frac{2}{\frac{1}{\frac{TP}{TP+FP}} + \frac{1}{\frac{TP}{TP+FN}}} = \frac{2TP}{2TP + FP + FN} \tag{26}$$

where TP is true positive, FP is false positive, and FN is false negative.

To the best of our knowledge, there is no low sampling-based classification study in the domain adaptation field for NILM. Therefore, we did not conduct a one-on-one comparison with other studies.

Table 2. Training parameters.

Parameter Description	**Value**
Number of TCN blocks	8 (TN)
	5 (SN)
Number of filters in each TCN block	128 (TN)
	64 (SN)
Filter size	3
Number of fully connected layers	5 (TN)
	3 (SN)
	2 (Domain Classifier)
Dilation factor	2^i for block i
Activation function	ReLU
Dropout probability	0.1
Number of maximum epochs	200
Number of minimum early stopping epochs	4
Mini-batch size	512
Learning rate	3×10^{-3}

4.3. Case Studies and Discussions

In this section, we conduct an experiment assuming two cases. In the first case, a house was designated as a source domain and a house was designated as a target domain within the same dataset. The second case was experimented with by specifying a source domain and a target domain between different datasets. Tables 3–5 show the F1 scores of domain adaptations for six segmentation methods. The 'Improvement' row shows how much the proposed method has improved. In addition, experiments on ablation studies are included, indicating how much each method affects overall performance.

4.3.1. Domain Adaptation within the Same Dataset

In this subsection, experiments are carried out on the first case described above. In Table 3, U^1 denotes House1, U^2 denotes House2, R^1 denotes House1 of REDD, and R^3 denotes House3 of REDD. There is no result for the appliances since REDD does not have a kettle, and DW is not used in R^3.

Based on the baseline, TCN-DA was the method that had the most influence on performance except for our method, showing an average performance improvement of 3.38%. Next, TS-DA showed a performance improvement of 2.45%. In the case of gkMMD-DA, there was a bit of performance improvement or slightly reduced performance. Table 4 shows F1 score for TCN and gkMMD. gkMMD generally helps improve the performance when used with networks with residual blocks. PL-DA showed an average performance stabilization of 0.51% because it learns models in the direction of stabilizing the domain by finetuning the network. Our method showed a significant performance improvement of 6.03% on average compared to the baseline.

4.3.2. Domain Adaptation between Different Datasets

In this subsection, experiments are performed on the second case described above. In Table 5, UK-DALE $\rightarrow$ REDD is an experiment using UK-DALE as a source domain and REDD as a target domain, and REDD $\rightarrow$ UK-DALE is an experiment using REDD as a source domain and UK-DALE as a target domain.

Table 3. F1 score comparison of domain adaptation within the same dataset.

Appliance	**Method**	**UK-DALE**		**REDD**	
		($U^1 \to U^2$)	**($U^2 \to U^1$)**	**($R^1 \to R^3$)**	**($R^3 \to R^1$)**
DW	Baseline	0.781	0.805	–	–
	TCN-DA	0.832	0.827	–	–
	gkMMD-DA	0.778	0.793	–	–
	TS-DA	0.812	0.826	–	–
	PL-DA	0.787	0.811	–	–
	Ours	0.822	0.832	–	–
	Improvement	5.25%	3.35%	–	–
FG	Baseline	0.833	0.834	0.817	0.818
	TCN-DA	0.842	0.841	0.829	0.840
	gkMMD-DA	0.837	0.836	0.819	0.819
	TS-DA	0.850	0.853	0.824	0.827
	PL-DA	0.834	0.845	0.818	0.819
	Ours	0.875	0.872	0.843	0.852
	Improvement	5.04%	4.56%	3.18%	4.16%
KT	Baseline	0.761	0.832	–	–
	TCN-DA	0.811	0.839	–	–
	gkMMD-DA	0.753	0.820	–	–
	TS-DA	0.807	0.835	–	–
	PL-DA	0.770	0.833	–	–
	Ours	0.817	0.868	–	–
	Improvement	7.36%	4.33%	–	–
MV	Baseline	0.742	0.791	0.793	0.790
	TCN-DA	0.751	0.798	0.806	0.721
	gkMMD-DA	0.746	0.795	0.797	0.774
	TS-DA	0.753	0.803	0.804	0.798
	PL-DA	0.744	0.796	0.794	0.793
	Ours	0.774	0.812	0.814	0.818
	Improvement	4.31%	2.65%	2.65%	3.54%
WM	Baseline	0.615	0.611	0.841	0.782
	TCN-DA	0.725	0.708	0.844	0.799
	gkMMD-DA	0.623	0.625	0.842	0.786
	TS-DA	0.668	0.653	0.832	0.783
	PL-DA	0.623	0.615	0.843	0.783
	Ours	0.736	0.713	0.870	0.832
	Improvement	19.67%	16.69%	3.45%	6.39%

Table 4. F1 score comparison of TCN + gkMMD domain adaptation within the same dataset.

Appliance	UK-DALE		REDD	
	$(U^1 \rightarrow U^2)$	$(U^2 \rightarrow U^1)$	$(R^1 \rightarrow R^3)$	$(R^3 \rightarrow R^1)$
DW	0.823	0.828	–	–
FG	0.857	0.854	0.834	0.847
KT	0.813	0.841	–	–
MV	0.762	0.805	0.809	0.764
WM	0.730	0.709	0.852	0.815

Table 5. F1 score comparison of domain adaptation between different datasets.

Appliance	Method	UK-DALE →REDD	REDD → UK-DALE
DW	Baseline	0.741	0.712
	TCN-DA	0.779	0.737
	gkMMD-DA	0.736	0.713
	TS-DA	0.770	0.745
	PL-DA	0.747	0.714
	Ours	0.778	0.747
	Improvement	4.99%	4.92%
FG	Baseline	0.786	0.764
	TCN-DA	0.794	0.787
	gkMMD-DA	0.787	0.769
	TS-DA	0.800	0.772
	PL-DA	0.787	0.770
	Ours	0.821	0.797
	Improvement	4.45%	4.32%
MV	Baseline	0.719	0.739
	TCN-DA	0.726	0.716
	gkMMD-DA	0.719	0.746
	TS-DA	0.729	0.749
	PL-DA	0.717	0.743
	Ours	0.742	0.763
	Improvement	3.2%	3.25%
WM	Baseline	0.563	0.758
	TCN-DA	0.669	0.773
	gkMMD-DA	0.573	0.766
	TS-DA	0.610	0.758
	PL-DA	0.568	0.763
	Ours	0.672	0.769
	Improvement	19.36%	1.45%

In the second case experiment, the average performance is improved by 5.74% even though the degree of domain characteristic change is greater than that of the first case experiment. Although the domains are different, the same type of appliance has almost the same pattern as the power usage, so the domain adaptation is well performed. Therefore, we have confirmed the possibility that in the field of NILM, we do not have to learn new neural networks even if each household and living area are different. Our method shows better results compared to the baseline.

Experiments show that domain adaptations within the same dataset perform well when the proposed method is used, and performance improvements can also be seen for domain adaptations between different datasets. It is a very significant result that our method without individual model learning for all households achieves a performance improvement of 5–6% through only one learning. There are several main reasons for improving accuracy. (1) Rich domain independent feature information is extracted by learning through teacher–student-based knowledge distillation. (2) By using TCN residual blocks and gkMMD together can effectively reduce the distribution mismatch between the two domains. (3) PL can stabilize the network's decision boundaries.

4.3.3. Discussions

The proposed model can automatically track the use of individual appliances under full load. We look at a series of our method-based applications for elderly households living alone and public electricity management institutions.

In the case of elderly households living alone, the risk of dying alone is generally very high. This risk situation is one of the critical problems to be solved at the government level. By analyzing device usage patterns, it is possible to develop a household risk detection system through abnormality detection in the household. Efficient energy management is an essential issue in public electricity management institutions. It is possible to develop an energy management system that adjusts the power generation ratio by identifying and managing energy-inefficient customers using home appliance usage patterns and power usage.

There are several limitations to the proposed method. (1) Domain adaptation is difficult to apply if house appliances of source and target data are different. (2) The difference in power usage between households is so large that the data imbalance is severe. (3) Although performance is improved by reducing distribution differences over the source and target features, there is no clear academic basis for extracting domain-independent features by reducing distribution differences. It is generally on an experimental basis. In future work, we aim to address the second limitation, which is the data imbalance. Data imbalance is the most fundamental problem in neural network training. Future work is planned in the direction of GAN-based sampling methods to resolve data imbalance or networks that perform high-quality learning despite data imbalance.

5. Conclusions

We developed a novel methodology that combines robust knowledge transfer and network stabilization for NILM to improve previous tasks and perform generalization across domains. The proposed method improves the detection performance of device usage for unlabeled target domain data by using a network trained only on the labeled source data. Teacher–student-based knowledge distillation is adopted to transfer quality features from the source domain. PL is utilized for domain stabilization through low-density separation between classes and entropy regularization effects. gkMMD is employed to reduce distribution differences between domain-independent features. Based on various techniques, we improve the performance of the proposed domain adaptation method by considering the distribution of robust domain-independent features.

To prove the proposed method, we used UK-DALE data and REDD as data. For data preprocessing, data such as training, verification, and testing were constructed by experimentally setting thresholds for distinguishing ON events in each appliance. Five methods of ablation study were performed for the performance test. Within the same dataset, domain adaptation improved the F1 score of the proposed method over the baseline by an average of 6.04%. Domain adaptation on different datasets improved the F1 score of the proposed method over the baseline by an average of 5.74%. While performance has not improved significantly for problems with much larger domain feature changes, maintaining existing performance alone is a great achievement.

Author Contributions: Formal analysis, C.-H.H.; Methodology, C.-H.H.; Writing—original draft, C.-H.H.; Writing—review & editing, C.-H.H. and S.-G.K.; Data curation, H.-E.L.; Validation, H.-E.L., Y.-J.K. and S.-G.K.; Visualization, H.-E.L.; Funding acquisition, Y.-J.K.; Supervision, S.-G.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government. [22ZS1300, Research on High Performance Computing Technology to overcome limitations of AI processing].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: REDD and UK-DALE datasets can be found at http://redd.csail.mit.edu/ (accessed on 30 May 2022) and https://ukerc.rl.ac.uk/DC (accessed on 30 May 2022), respectively.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gherheș, V.; Fărcașiu, M.A. Sustainable Behavior among Romanian Students: A Perspective on Electricity Consumption in Households. *Sustainability* **2021**, *13*, 9357. [CrossRef]
2. Somchai, B.; Boonyang, P. Non-intrusive appliances load monitoring (nilm) for energy conservation in household with low sampling rate. *Procedia Comput. Sci.* **2016**, *86*, 172–175.
3. Nur Farahin, E.; Md Pauzi, A.; Yusri, H.M. RETRACTED: A review disaggregation method in Non-intrusive Appliance Load Monitoring. *Renew. Sustain. Energy Rev.* **2016**, *66*, 163–173.
4. Shikha, S.; Angshul, M. Deep sparse coding for non–intrusive load monitoring. *IEEE Trans. Smart Grid* **2017**, *9*, 4669–4678.
5. Cominola, A.; Giuliani, M.; Piga, D.; Castelletti, A.; Rizzoli, A.E. A hybrid signature-based iterative disaggregation algorithm for non-intrusive load monitoring. *Appl. Energy* **2017**, *185*, 331–344. [CrossRef]
6. Shi, X.; Ming, H.; Shakkottai, S.; Xie, L.; Yao, J. Nonintrusive load monitoring in residential households with low-resolution data. *Appl. Energy* **2019**, *252*, 113283. [CrossRef]
7. Georgia, E.; Lina, S.; Vladimir, S. Power Disaggregation of Domestic Smart Meter Readings Using Dynamic Time warping. In Proceedings of the 2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP), Athens, Greece, 21–23 May 2014; IEEE: Manhattan, NY, USA, 2014; pp. 36–39.
8. Yu-Hsiu, L.; Men-Shen, T. Non-intrusive load monitoring by novel neuro-fuzzy classification considering uncertainties. *IEEE Trans. Smart Grid* **2014**, *5*, 2376–2384.
9. Kanghang, H.; He, K.; Stankovic, L.; Liao, J.; Stankovic, V. Non-intrusive load disaggregation using graph signal processing. *IEEE Trans. Smart Grid* **2016**, *9*, 1739–1747.
10. Hart, G.W. Nonintrusive appliance load monitoring. *Proc. IEEE* **1992**, *80*, 1870–1891. [CrossRef]
11. Yang, Y.; Zhong, J.; Li, W.; Gulliver, T.A.; Li, S. Semisupervised multilabel deep learning based nonintrusive load monitoring in smart grids. *IEEE Trans. Ind. Inform.* **2019**, *16*, 6892–6902. [CrossRef]
12. Sagar, V.; Shikha, S.; Angshul, M. Multi-label LSTM autoencoder for non-intrusive appliance load monitoring. *Electr. Power Syst. Res.* **2021**, *199*, 107414.
13. Hyeontaek, H.; Sanggil, K. Nonintrusive Load Monitoring using a LSTM with Feedback Structure. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–11.
14. Da Silva Nolasco, L.; Lazzaretti, A.E.; Mulinari, B.M. DeepDFML-NILM: A New CNN-Based Architecture for Detection, Feature Extraction and Multi-Label Classification in NILM Signals. *IEEE Sens. J.* **2021**, *22*, 501–509. [CrossRef]
15. Christoforos, N.; Dimitris, V. On time series representations for multi-label NILM. *Neural Comput. Appl.* **2020**, *32*, 17275–17290.
16. Patrick, H.; Calatroni, A.; Rumsch, A.; Paice, A. Review on deep neural networks applied to low-frequency nilm. *Energies* **2021**, *14*, 2390.
17. Kong, W.; Dong, Z.Y.; Hill, D.J.; Luo, F.; Xu, Y. Improving nonintrusive load monitoring efficiency via a hybrid programing method. *IEEE Trans. Ind. Inform.* **2016**, *12*, 2148–2157. [CrossRef]
18. Basu, K.; Debusschere, V.; Douzal-Chouakria, A.; Bacha, S. Time series distance-based methods for non-intrusive load monitoring in residential buildings. *Energy Build.* **2015**, *96*, 109–117. [CrossRef]
19. Yaroslav, G.; Lempitsky, V. Unsupervised Domain Adaptation by Backpropagation. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1180–1189.
20. Long, M.; Zhu, H.; Wang, J.; Jordan, M.I. Unsupervised domain adaptation with residual transfer networks. *Adv. Neural Inf. Processing Syst.* **2016**, *29*, 136–144.

21. Liu, Y.; Zhong, L.; Qiu, J.; Lu, J.; Wang, W. Unsupervised domain adaptation for nonintrusive load monitoring via adversarial and joint adaptation network. *IEEE Trans. Ind. Inform.* **2021**, *18*, 266–277. [CrossRef]
22. Lin, J.; Ma, J.; Zhu, J.; Liang, H. Deep Domain Adaptation for Non-Intrusive Load Monitoring Based on a Knowledge Transfer Learning Network. *IEEE Trans. Smart Grid* **2021**, *13*, 280–292. [CrossRef]
23. Suzuki, K.; Inagaki, S.; Suzuki, T.; Nakamura, H.; Ito, K. Nonintrusive Appliance Load Monitoring Based on Integer Programming. In Proceedings of the 2008 SICE Annual Conference, Tokyo, Japan, 20–22 August 2008; IEEE: Manhattan, NY, USA, 2008; pp. 2742–2747.
24. Michael, B.; Jürgen, V. Nonintrusive appliance load monitoring based on an optical sensor. In Proceedings of the 2003 IEEE Bologna Power Tech Conference Proceedings, Bologna, Italy, 23–26 June 2003; IEEE: Manhattan, NY, USA, 2003; Volume 4, p. 8.
25. Arend, B.J.; Xiaohua, X.; Jiangfeng, Z. Active Power Residential Non-Intrusive Appliance Load Monitoring System. In Proceedings of the AFRICON 2009, Nairobi, Kenya, 23–25 September 2009; IEEE: Manhattan, NY, USA, 2009; pp. 1–6.
26. Pan, S.J.; Tsang, I.W.; Kwok, J.T.; Yang, Q. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Netw.* **2010**, *22*, 199–210. [CrossRef] [PubMed]
27. Mei, W.; Weihong, D. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135–153.
28. Isobe, T.; Jia, X.; Chen, S.; He, J.; Shi, Y.; Liu, J.; Lu, H.; Wang, S. Multi-Target Domain Adaptation with Collaborative Consistency Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8187–8196.
29. Yuang, L.; Wei, Z.; Jun, W. Source-Free Domain Adaptation for Semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 1215–1224.
30. Guoqiang, W.; Lan, C.; Zeng, W.; Chen, Z. Metaalign: Coordinating Domain Alignment and Classification for Unsupervised Domain Adaptation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16643–16653.
31. Zechen, B.; Wang, Z.; Wang, J.; Hu, D.; Ding, E. Unsupervised Multi-Source Domain Adaptation for Person Re-Identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 12914–12923.
32. Jingjing, L.; Jing, M.; Su, H.; Lu, K.; Zhu, L.; Shen, H.T. Faster domain adaptation networks. *IEEE Trans. Knowl. Data Eng.* **2021**, 1. [CrossRef]
33. Dongdong, W.; Han, T.; Chu, F.; Zuo, M.J. Weighted domain adaptation networks for machinery fault diagnosis. *Mech. Syst. Signal Processing* **2021**, *158*, 107744.
34. Tzeng, E.; Hoffman, J.; Zhang, N.; Saenko, K.; Darrell, T. Deep domain confusion: Maximizing for domain invariance. *arXiv* **2014**, arXiv:1412.3474.
35. Hao, W.; Wang, W.; Zhang, C.; Xu, F. Cross-Domain Metric Learning Based on Information Theory. In Proceedings of the AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014.
36. Juntao, H.; Hongsheng, Q. Unsupervised Domain Adaptation with Multi-kernel MMD. In Proceedings of the 2021 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; IEEE: Manhattan, NY, USA, 2021; pp. 8576–8581.
37. Zhang, W.; Zhang, X.; Lan, L.; Luo, Z. Maximum mean and covariance discrepancy for unsupervised domain adaptation. *Neural Processing Lett.* **2020**, *51*, 347–366. [CrossRef]
38. Wen, Z.; Wu, W. Discriminative Joint Probability Maximum Mean Discrepancy (DJP-MMD) for Domain Adaptation. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; IEEE: Manhattan, NY, USA, 2020; pp. 1–8.
39. Mingsheng, L.; Zhu, H.; Wang, J.; Jordan, M.I. Deep Transfer Learning with Joint Adaptation Networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 2208–2217.
40. Wan, N.; Zhang, C.; Chen, Q.; Li, H.; Liu, X.; Wei, X. MDDA: A Multi-scene Recognition Model with Multi-dimensional Domain Adaptation. In Proceedings of the 2021 IEEE 4th International Conference on Electronics Technology (ICET), Chengdu, China, 7–10 May 2021; IEEE: Manhattan, NY, USA, 2021; pp. 1245–1250.
41. Wang, L.; Mao, S.; Wilamowski, B.M.; Nelms, R.M. Pre-trained models for non-intrusive appliance load monitoring. *IEEE Trans. Green Commun. Netw.* **2021**, *6*, 56–68. [CrossRef]
42. Shaojie, B.; Zico, K.J.; Koltun, V.K. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
43. Geoffrey, H.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
44. Xin, Y.; Chaofeng, H.; Lifeng, S. Two-Stream Federated Learning: Reduce the Communication Costs. In Proceedings of the 2018 IEEE Visual Communications and Image Processing (VCIP), Taichung, Taiwan, 9–12 December 2018; IEEE: Manhattan, NY, USA, 2018; pp. 1–4.
45. Kelly, J.K.; Knottenbelt, W. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Sci. Data* **2015**, *2*, 150007. [CrossRef]
46. Zico, K.J.; Johnson, M.J. Redd: A public data set for energy disaggregation research. In Proceedings of the Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA, USA, 21 August 2011; pp. 59–62.

47. Linge, S.; Langtangen, H.P. *Programming for Computations-Python: A Gentle Introduction to Numerical Simulations with Python 3.6*; Springer Nature: Cham, Switzerland, 2020.
48. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Processing Syst.* **2019**, *32*, 8024–8035.