FedAR+: A Federated Learning Approach to Appliance Recognition with Mislabeled Data in Residential Environments

Ashish Gupta[1], Hari Prabhat Gupta[2], and Sajal K. Das[1]
ashish.gupta@mst.edu,hariprabhat.cse@iitbhu.ac.in,sdas@mst.edu
[1]Dept. of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409, USA
[2]Dept. of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, India

ABSTRACT

With the enhancement of people's living standards and the rapid evolution of cyber-physical systems, residential environments are becoming smart and well-connected, causing a significant raise in overall energy consumption. As household appliances are major energy consumers, their accurate recognition becomes crucial to avoid unattended usage and minimize peak-time load on the smart grids, thereby conserving energy and making smart environments more sustainable. Traditionally, an appliance recognition model is trained at a central server (service provider) by collecting electricity consumption data via smart plugs from the clients (consumers), causing a privacy breach. Besides that, the data are susceptible to noisy labels that may appear when an appliance gets connected to a non-designated smart plug. While addressing these issues jointly, we propose a novel federated learning approach to appliance recognition, called FedAR+, enabling decentralized model training across clients in a privacy-preserving way even with mislabeled training data. FedAR+ introduces an adaptive noise handling method, essentially a joint loss function incorporating weights and label distribution, to empower the appliance recognition model against noisy labels. By deploying smart plugs in an apartment complex, we collect a labeled dataset that, along with two existing datasets, are utilized to evaluate the performance of FedAR+. Experimental results show that our approach can effectively handle up to 30% concentration of noisy labels while outperforming the prior solutions by a large margin on accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; • **Hardware** → *Energy metering*.

KEYWORDS

Appliance recognition, federated learning, noisy labels, smart plug

ACM Reference Format:
Ashish Gupta[1], Hari Prabhat Gupta[2], and Sajal K. Das[1]. 2023. FedAR+: A Federated Learning Approach to Appliance Recognition with Mislabeled Data in Residential Environments. In *ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2022) (ICCPS '23), May 9–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3576841.3585921

ICCPS '23, May 9–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0036-1/23/05…$15.00
https://doi.org/10.1145/3576841.3585921

1 INTRODUCTION

Energy consumption in residential buildings is increasing rapidly with the growth of electrical household appliances. According to the United States Energy Information Administration (US EIA) [32], 22% of the total energy consumption in 2020 is accounted by residential buildings, requiring dedicated efforts to reduce the usage of electricity. A practical solution is encouraging consumers to use electric appliances efficiently, which involves recognizing the appliances uniquely based on their consumption patterns recorded via appliance-wise smart plugs [26, 29, 34]. By having information about currently running appliances, the consumers can minimize the electricity usage by restricting high power appliances (e.g., electric heater, air conditioner) during peak hours [4]. Moreover, the utility company (service provider) may also incentivize the consumers by offering direct monetary benefit through a dynamic pricing policy [4, 11] and indirect benefit through an appliance-wise breakage of consumption bill. Literature indicates that appliance recognition has been a building block in wide range of important cyber-physical applications such as load forecasting [2, 37], occupancy detection [13], and energy management in smart buildings [20, 24, 28]. However, the current appliance recognition approaches have disregarded the following two practical issues.

(i) *Privacy preservation of consumers' data* – As recognition model is essentially a machine learning model, it requires a large amount of labeled training data which, in general, collected from many consumers at a central server (service provider). Sharing of data brings in a privacy concern to the consumers as the data may be misused by adversaries via theft or burglary, and by detecting home occupancy [1, 13]. Hence, the consumers may be reluctant to upload the data and as a consequence, the existing approaches [5, 19, 29, 33, 34, 39] would fail to train the recognition model, indicating a need of a model that can be trained collaboratively at the consumer side without sharing any data.

(ii) *Mislabeled training data* – Some data samples may appear with wrong (noisy) labels when an appliance is mistakenly connected to a non-designated smart plug.[1] Moreover, a compromised consumer may also flip the labels in its local dataset with an intent to poison the model and to receive monetary benefits from a rival service provider. As prior studies [19, 29, 33, 34] do not incorporate any noisy-label handling mechanism, they can not withstand mislabeled training data. Slightly on a different track, learning with noisy labeled data has been a topic of great interest in computer

[1] Assuming that each smart plug, during deployment, is designated to a specific appliance to collect the labeled data automatically.

vision; however, the proposed solutions [6, 8, 16, 31, 35, 36] mainly relied upon visual features, thus their applicability to time series data (generated from smart plugs) is discouraged.

Although there exist some works [25, 28] on privacy-preserving appliance recognition using Federated Learning (FL), they do not consider the presence of noisy labels in training data. *In this paper, we address this important challenge by building an appliance recognition model in a collaborative manner using mislabeled training data while preserving consumers' privacy.* To the best of our knowledge, this is the first work to tackle the practical issues of privacy preservation and mislabeled training data **jointly** for appliance recognition in residential environments.

Contributions: Major contributions of this paper are given below:

- With a goal to train an appliance recognition model across distributed consumers using their local private data, we propose a novel federated learning approach, called FedAR+, in presence of a coordinating server (service provider). The server initializes the training by broadcasting the model (i.e., weights) to all the clients (consumers); each client re-trains the model using its local data and dispatches the updated model back to the server for aggregation. By repeating the above steps for some iterations, FedAR+ produces a generalized model without exposing the consumers' data.
- FedAR+ incorporates an innovative aggregation function to deal with the biasing problem caused due to non independent and identically distributed (non-iid) data across clients.
- We propose an adaptive noise handling method that strategically exploits a joint loss function, incorporating the weight parameters and label distributions, to enable the learning with mislabeled training data.
- Finally, we collect real data by deploying smart plugs in three houses in an apartment complex, to experimentally validate the performance of FedAR+. Moreover, to demonstrate its efficacy, we also employ two widely used datasets from the same domain. The overall results show that FedAR+ outperforms prior solutions by a large margin while achieving an accuracy of more than 86% even when the concentration of noisy labels in training data is as high as 30%.

The paper is organized as follows. Section 2 reviews the related work while Section 3 proposes our federated learning approach, FedAR+. Section 4 discusses the dataset preparation steps and elaborates the causes for the presence of noisy labels. Section 5 builds the underlying appliance recognition model with a noise handling method. Section 6 evaluates the performance of FedAR+ and compares with prior solutions. Finally, Section 7 concludes the paper.

2 RELATED WORK

This section discusses the notable and relevant existing works to position our proposed approach.

2.1 Appliance recognition

Many works exist on appliance recognition as it has been a building block to energy monitoring applications. For example, in [20], a lightweight appliance recognition model is developed for energy management in smart buildings. The authors in [5] attempted to identify a malfunctioning appliance and its operating states by leveraging electricity consumption patterns. While a line of works [19, 34, 39] involved in distinguishing the appliances from one to another, the work in [29] aimed to identify load profile as intermittent, continuous, or phantom, for energy management in smart home settings. Slightly different from above works, Codispoti *et al.* [3] presented a K-active neighbors based appliance recognition approach to learn from unlabeled data.

Although the aforementioned prior approaches achieve good performance using machine learning and deep learning algorithms, their performance heavily relies on the assumption that the training data are correctly labeled and do not contain any noisy labels. However, in practice, satisfying this assumption requires additional care from the consumers during data collection (via smart plugs [3, 5, 34, 39]), restricting their flexibility and thereby the consumers may be reluctant to adopt such solutions. Besides that the recognition model should not fully rely upon the consumers' actions rather it should be robust enough to leverage mislabeled training data.

2.2 Learning with noisy labels

Learning with mislabeled (noisy labeled) data has been a widely studied problem in computer vision and image processing because the manual labeling is time consuming and costly [17, 30]. The work in [8] presented an iterative learning approach to re-label the noisy-labeled training samples while in another work [31] the authors estimated correct labels against noisy ones during training by jointly optimizing the model parameters and intermediary corrected labels. In [35], a symmetric learning approach is proposed to simultaneously address the presence of noisy labels and overfitting problem of Deep Neural Networks (DNNs). A distillation process leveraging knowledge graph is introduced in [16] to learn with noisy labels. Recently, a meta-learning approach is developed in [36] to directly learn correct labels from the training data. However, as these approaches mostly work around visual features, they can not offer an accurate solution to mislabeled time series data.

2.3 Federated learning

In last few years, a new learning paradigm, Federated Learning (FL) [22] has received an unprecedented attention because it facilitates collaborative model training without compromising clients' privacy. Prior works illustrate the effectiveness of FL in real-world applications such as next word prediction [9], keyword spotting [14], and visual object detection [18]. However, FL is yet to be explored for the appliance recognition models that are otherwise trained at the central server by collecting data from multiple clients (consumers) and revealing the client's privacy. FL offers an effective solution to this problem by keeping the data locally with the clients while allowing participation in collaborative training. Recently, a few studies [25, 40] have also attempted to apply FL in smart energy management to enable load forecasting and load disaggregation at consumer side. In another work [28], the authors presented an FL approach to identify office plug load, however they do not consider the presence of noisy labels in the training data, which we aim to address in this work. Besides all, the application of FL to appliance recognition needs to be investigated from robustness perspective in the presence of noisy labels.

3 FedAR+ APPROACH

This section presents an overall setup of our FL approach, FedAR+, with multiple clients[2] and a common remote server, as depicted in Figure 1. In appliance recognition scenario, the consumer acts as a client and the service provider works as a remote server. A client may have many appliance-specific smart plugs, each connected to a designated appliance to measure the appliance's electricity consumption and transfer that data to a local in-house computing device. To initialize training, the server dispatches an appliance recognition model to all the clients. Each client retrains the model using its local data and sends the weight updates to the server for aggregation. Next, the aggregated (or global) model is sent back to the clients. By repeating these steps for a certain number of global rounds, the model eventually converges to an optimal solution.

In FedAR+, we build a deep learning model for appliance recognition which requires a large amount of data for training. At the beginning of the deployment, the clients may not have sufficient data and thereby the model would need first few rounds to get stabilized. To avoid flow disruption, we discuss dataset preparation steps using the power consumption data (generated from smart plugs) in Section 4; and in subsequent Section 5, we present appliance recognition model with noise handing method. We formulate an aggregate function to alleviate the bias that might be introduced by the clients having substantially larger dataset than the others.

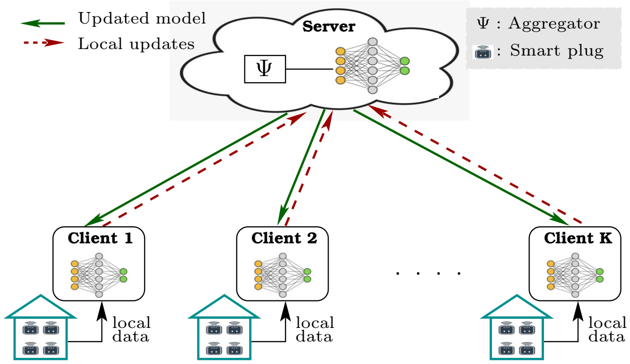


Figure 1: Overview of FedAR+. At each client, a noise handling method is incorporated to enable the model learn with mislabeled data.

Let K denote the number of clients collaborating in the learning to build the recognition model. Let $\mathcal{D}^j = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}\}$ be the local dataset with j^{th} client, which is collected over a fixed period of time, where $1 \leq j \leq K$. At each update round, the objective of the remote server is to learn optimal weight parameters $\boldsymbol{\theta}$ by minimizing an empirical loss function as

$$\underset{\boldsymbol{\theta}}{\text{argmin}} \quad \left\{\mathcal{F}(\boldsymbol{\theta}) = \Psi\left(\{f^j(\boldsymbol{\theta})\}_{1\leq j\leq K}\right)\right\}, \tag{1}$$

where $\Psi(\cdot)$ is an aggregate function and $f^j(\cdot)$ is the local objective function used by the j^{th} client. We also propose a noise handling method, in Section 5.2, to facilitate learning with mislabeled training data at the client.

3.1 Local model update at client

FedAR+ uses second-order method to perform local updates at the client. Particularly, we adopt canonical Newton's method of the form $-\nabla^2(f^j)^{-1}\nabla f^j$ [23] as it improves the convergence rate and reduces the accumulation of errors. Given the weight parameters $\boldsymbol{\theta}_{[t]}$ of the global model at update round t, the client j first computes the local gradient as

$$g^j_{[t]} = \nabla f^j(\boldsymbol{\theta}_{[t]}). \tag{2}$$

The client next computes the second-order gradient (Hessian matrix) at $\boldsymbol{\theta}_{[t]}$ as follows

$$h^j_{[t]} = \nabla^2 f^j(\boldsymbol{\theta}_{[t]}). \tag{3}$$

Now, the local model at the client j is updated as

$$\boldsymbol{\theta}^j_{[t+1]} = \boldsymbol{\theta}^j_{[t]} - \eta(h^j_{[t]})^{-1}g^j_{[t]} \tag{4}$$

and η is the learning rate. Finally, the local updates are sent back to the server for aggregation.

3.2 Global model update at server

The problem of aggregation at the server becomes quite simple if we assume that all the clients have independent identically distributed (iid) data, and it can be easily solved by using FedAvg [22] as

$$\Psi\left(\{f^j(\boldsymbol{\theta})\}_{1\leq j\leq K}\right) \overset{\text{def}}{=} \sum_{j=1}^{K} \frac{N^j}{N^{total}} \boldsymbol{\theta}^j_{[t+1]}, \tag{5}$$

where N^j is the size of $\mathcal{D}^j$ and $N^{total} = N^1 + N^2 + \cdots + N^h$. However, this assumption is unrealistic for applying FL to appliance recognition as the clients may have different number of appliances (essentially non-iid data). With FedAvg, the model may be biased towards the clients who have substantially larger dataset than others. To deal with this situation, we introduce an aggregation function

$$\Psi\left(\{f^j(\boldsymbol{\theta})\}_{1\leq j\leq K}\right) \overset{\text{def}}{=} \sum_{j=1}^{K} \frac{1}{K} \cdot \boldsymbol{\theta}^j_{[t+1]}. \tag{6}$$

With the new aggregation function, each client would receive an unbiased model regardless of number of appliances the client possesses. Algorithm 1 summarizes the major steps of FedAR+ with K clients for T number of global rounds.

Algorithm 1: FedAR+

```
Initialization:
The server builds and broadcasts a recognition model to all $K$ clients.
for $t \leftarrow 1$ to $T$ do
    Local model update at round $t$:
    for each client $j \in \{1, 2, \cdots, K\}$ do
        /* $\boldsymbol{\theta}^j_{[t]}$ is the weight parameters of local model */
        Obtain $\boldsymbol{\theta}^j_{[t+1]}$ using Eq. 4. // the underlying loss function is formulated in Section 5.2.
        Dispatch $\boldsymbol{\theta}^j_{[t+1]}$ to the server.
    Global model update using aggregation at round $t$:
    $\boldsymbol{\theta}_{[t+1]} = \sum_{j=1}^{K} \frac{1}{K} \cdot \boldsymbol{\theta}^j_{[t+1]}$, using Eq. 6.
    Broadcast $\boldsymbol{\theta}_{[t+1]}$ to the clients
```

[2] A client refers to a low-end computing device (e.g., personal computer) installed at consumer's house to collect data from smart plugs.

- *Model convergence:* The global recognition model (at the server) advances as the training progresses and it is said to be converged when stops improving. We study the model convergence, under the standard assumptions on the function $\mathcal{F}(\cdot)$ [22], in terms of the optimality gap $\delta = \mathcal{F}(\boldsymbol{\theta}_{[T]}) - \mathcal{F}(\boldsymbol{\theta}^*)$, where T denotes the maximum number of rounds and $\boldsymbol{\theta}^*$ denotes the weights of the optimal model. Ideally, $\delta \approx 0$ for a sufficiently large T. Assuming K clients, FedAR+ can achieve $O(\frac{1}{\sqrt{KT}})$ convergence for our DNN model (i.e., non-convex optimization problem). Our experimental results, reported in Section 6.4.1, show that the global model converges in $T = 30$ rounds (with 50 local iterations on each clients at each round) with 10 clients even when there exist 30% noisy labels.

4 DATASET PREPARATION

In this section, we discuss data collection and preprocessing for creating a labeled dataset. We utilize the power consumption data for recognizing appliances such as refrigerator, electric kettle, television, etc. The data are collected by connecting the appliance to power socket through a designated smart plug that provides a sequence of time stamped readings at a preset sampling rate.

4.1 Data collection

We collect power consumption data from three different households (within an apartment complex) for six common household appliances: refrigerator, microwave oven, television, washing machine, air conditioner, and mixer grinder. Each appliance is connected to a designated smart plug that transmits the readings to a in-house data collector (e.g., personal computer) at 1Hz. As we collected the data for a period of one month from each house, we got total 18 time series (i.e., six time series from each of the three households).

Definition 1 (**Time Series of Consumption**). *It is a temporal sequence of data points collected over a period of time. Let $X = \{x_1, x_2, \cdots, x_n\}$ denote the Time Series of power Consumption (TSC) readings from a designated smart plug, where n is the total number of data points collected during the entire experiment; and x_i denotes a reading taken at time t_i, where $1 \leq i \leq n$ and $t_{i-1} < t_i$.*

We construct the dataset using TSCs of different appliances. As the appliance can change its state from ON to OFF or vise-versa several times, each TSC (denoted by X) includes readings corresponding to both the states. We first separate out only the subsequences (of X) that correspond to ON states occurred at distinct time steps along X. Then for each separated subsequence, an appliance footprint is computed and stored as an instance of the respective appliance.

4.2 Data Preprocessing

Let us first discuss the terminologies for a better illustration of data preprocessing.

- *Switch point:* For a TSC $X = \{x_1, x_2, \cdots, x_n\}$, a time instance t is said to be a switch point if the following conditions hold: (i) The difference $\delta(t) = |X(t) - X(t-1)| > \phi_1$, a predefined threshold, where $X(t)$ and $X(t-1)$ denote the power consumption readings at time t and $t-1$, respectively; and (ii) The rate of change in power readings $\delta_r(t) = \delta(t)/X(t) > \phi_2$, another threshold. For setting an appropriate value for ϕ_1 and ϕ_2, we visualized several time series for different appliances including both low power (e.g., television) and high power (e.g., air conditioner). We observed that with $\phi_1 = 30$ watts and $\phi_2 = 0.2$ (i.e., 20%) jointly, the switch points can be detected correctly for most commonly available appliances. Further, as the thresholds are set empirically, their values are subject to change according to the appliance' operating environment (such as brand and power rating standards of different countries). With small thresholds, we may get frequent false positive; on the contrary, some ON states may get lost with large thresholds.
- *Steady point:* A time point t along the time series X is said to be steady if $\delta_r(t) < \phi_2$.
- *Steady period:* Given a time series X, a steady period is a subsequence $X_{t:m} = \{x_{t+1}, x_{t+2}, \cdots, x_{t+m}\}$ if all of its time points are steady. Here, t and m respectively denote a switch point and the length of the steady period, where $t < (n - m)$.

Definition 2 (**Appliance Footprint**). *For a given steady period $X_{t:m}$, corresponding to the ON state of the appliance, we define the appliance footprint as:*

$$X_{af} = \{X_{t:m}(i) - X_{t:m}(i-1) \mid 1 \leq i \leq m\}, \tag{7}$$

where $X_{t:m}(i)$ denotes the i^{th} data point of steady period.

We compute single-order differences between the consecutive data points to capture subtle fluctuations, revealing better identifiable patterns than those with higher-order statistics. Moreover, the single order difference automatically scales down the values to a smaller range, eliminating the need of normalization. The appliance footprint essentially represents the power consumption pattern of the appliance when it is active. For a given TSC $X = \{x_1, x_2, \cdots, x_n\}$, the extraction of footprints includes following three steps:

1. Identify switch points in X under thresholds ϕ_1 and ϕ_2.
2. For each identified switch point t, follow two sub-steps: (a) search for a steady period of length m after t. Let $X_{t:m} = \{x_{t+1}, x_{t+2}, \cdots, x_{t+m}\}$ be a steady period obtained after the switch point t. (b) if $X(t) - X(t+m) < 0$, then the steady period $X_{t:m}$ corresponds to ON state of the appliance; otherwise OFF state.
3. The steady periods corresponding to ON states, are used to obtain appliance footprints; each of which along with its label (name of the appliance) is stored as an instance.

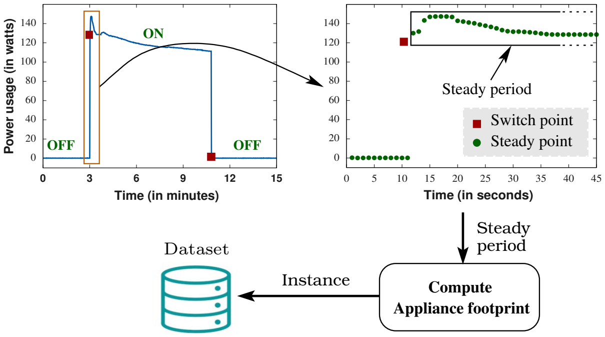


Figure 2: Illustrating switch points, steady points, and steady period, in a TSC of a refrigerator for a window of 15 minutes.

Figure 2 illustrates a TSC of a refrigerator with identified switch points, steady points, and steady period. The obtained steady periods are used to compute the appliance footprints. Upon obtaining the footprints by processing TSCs of all the appliances, we perform padding on shorter instances to make all the instances of equal length and store them in the dataset.

4.3 Presence of noisy labels

During local data collection at the client, some TSCs may get associated with noisy (wrong) labels due to following reasons:

- *From deployment perspective:* An appliance (say A_1) mistakenly got connected to a non-designated smart plug that was marked to connect with some other appliance (say A_2). Consequently, the generated TSC receives a noisy label A_2, creating several mislabeled instances (wrong appliance footprints) in the training dataset. It is true that such noisy labels may be avoided *at the cost of additional care from the consumers*, however it is not preferable rather the model should be robust against mislabeled training data.
- *From security perspective:* A malicious consumer (or compromised client) may attempt to inject a wrong label intentionally, to gain some incentive from a rival service provider. As a consequence, the client's local model would generate corrupted local updates, which eventually would diminish the performance of global model.

Definition 3 (**Noisy label**). *Let $\mathcal{D} = \{(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})\}$ be a dataset where $\mathcal{X}^i \in \boldsymbol{\mathcal{X}}$ is the i^{th} instance associated with a class label $\mathcal{Y}^i \in$ $\mathrm{Y} = \{y_1, y_2, \cdots, y_C\}$, the set of all C classes (appliances). The label $\mathcal{Y}^i$ is said to be noisy if either of the following holds: (i) $\mathcal{Y}^i$ is mislabeled as other class label, i.e., $\mathcal{Y}^i \in \{\mathrm{Y} - y_c\}$, where y_c denotes the correct class label of $\mathcal{X}^i$, or (ii) $\mathcal{Y}^i$ is an arbitrary class label, i.e., $\mathcal{Y}^i \notin \mathrm{Y}$.*

5 APPLIANCE RECOGNITION MODEL

This section presents a deep neural network (DNN) for appliance recognition that trains collaboratively on locally collected data. The choice of DNN is inspired by its success at recognition tasks with a rich set of learnable features. The network (model) learns from a training dataset and predicts the class label of a new instance. Additionally, we propose an adaptive noise handling method to enable the model learning with mislabeled data at the clients.

5.1 Base model overview

We build a DNN with three convolutional layers (connected sequentially) followed by a flatten and a Fully Connected (FC) layer, as shown in Figure 3. Let $\mathcal{D} = \{\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}\}$ be a local training dataset available with a client. The model takes a training dataset $\mathcal{D}$ and yields a set of class probabilities using a *softmax* function. The convolutional layers are all one-dimensional, each consisting of 128 filters of size 1×1 with input shape $(1, m)$, where m denotes the number of data points in each instance (i.e., appliance footprint). Considering there exist total C labels in $\mathcal{D}$, we use C neurons at the FC layer. Finally, a *softmax* function is applied on the output of FC layer to get the class probabilities.

Now, we present mathematical formulation of our base model (i.e., excluding the noise handling method). Given the dataset $\mathcal{D}$, the recognition model mainly attempts to learn a mapping $\mathcal{H} : \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{Y}}$, which usually expressed as

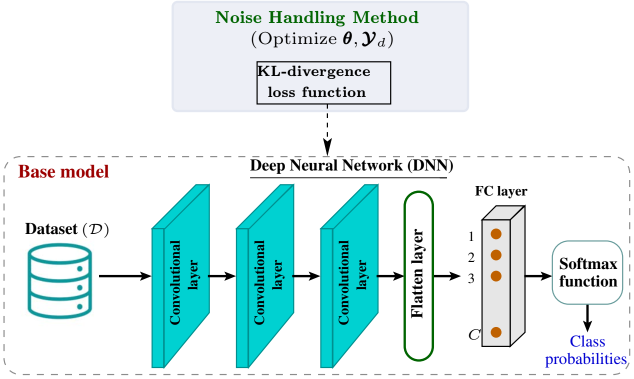


Figure 3: An overview of the appliance recognition model in FedAR+.

$$\mathcal{H}(\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}) = \sigma_{\mathrm{Y}}(\boldsymbol{\theta}\boldsymbol{\mathcal{X}}), \tag{8}$$

where $\sigma(\cdot)$ is a *softmax* function and $\mathrm{Y} = \{y_1, y_2, \cdots, y_C\}$ is a set of different class labels in $\mathcal{D}$. The function $\sigma(\cdot)$ can transform a vector into probability distribution over its elements. For a vector $z \in \mathbb{R}^C$, the *softmax* function is:

$$\sigma_c(z) = \frac{\mathrm{e}^{z_c}}{\sum_{c=1}^{C} \mathrm{e}^{z_c}} = p(c|z) \quad \forall c \in \{1, 2, \cdots, C\}.$$

Rewriting Eq. 8,

$$\mathcal{H}(\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}) = p(\mathrm{Y}|\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}). \tag{9}$$

To this end, the appliance recognition problem with base model, at the client, can be observed as a local optimization problem

$$f(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \mathcal{L}\big(\boldsymbol{\mathcal{Y}}, \mathcal{H}(\boldsymbol{\mathcal{X}}, \boldsymbol{\theta})\big) \right\}, \tag{10}$$

where $\mathcal{L}(\cdot)$ is an underlying empirical loss function. The base model employs cross-entropy loss function as widely used in DNNs for solving recognition problems [7]. The cross-entropy loss function for $\mathcal{D}$ with N instances, is given as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mathcal{Y}}, \mathcal{H}(\boldsymbol{\mathcal{X}}, \boldsymbol{\theta})) &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \mathbb{I}(\mathcal{Y}^i, y_c) \log \mathcal{H}(\mathcal{X}^i, \boldsymbol{\theta}), \\ &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \mathbb{I}(\mathcal{Y}^i, y_c) \log p(\mathcal{Y}^i = y_c | \mathcal{X}^i, \boldsymbol{\theta}), \end{aligned} \tag{11}$$

where $\mathbb{I}(\mathcal{Y}^i, y_c) = 1$ if $\mathcal{Y}^i$ is y_c, and 0 otherwise.

5.2 Noise handling method

To enable the base model learning from mislabeled training data, this section proposes a noise handling method that learns correct labels by iteratively updating the label distributions probabilistically, which is significantly different from the existing approaches [6, 15, 38] where constant distributions were used in all the iterations. In our method, the model optimizes label distributions along with weight parameters during training, and therefore the local objective function (i.e., Eq. 10) can be written as

$$f(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta},\boldsymbol{\mathcal{Y}}_d}{\operatorname{argmin}} \left\{\mathcal{L}\big(\boldsymbol{\mathcal{Y}}_d, \mathcal{H}(\boldsymbol{\mathcal{X}},\boldsymbol{\theta})\big)\right\}, \tag{12}$$

where $\boldsymbol{\mathcal{Y}}_d$ denotes the label distributions among C classes for all N instances of the dataset $\mathcal{D}$. To solve Eq. 12, we introduce a noise handling method consisting of three steps explained below.

5.2.1 *Learn weight parameters $\boldsymbol{\theta}$.*

First, we train the base model with cross-entropy loss function (Eq. 11) on the training dataset $\mathcal{D}$. By optimizing the loss function, the model learns the weight parameters $\boldsymbol{\theta}$. Due to noisy labels, the learned weights may be far from optimality; nevertheless, they can certainly be used for the initial estimation of the label distributions over the training data.

5.2.2 *Estimate label distributions $\boldsymbol{\mathcal{Y}}_d$:*

Given the trained model, the label distributions $\boldsymbol{\mathcal{Y}}_d$ can be estimated for all instances of $\mathcal{D}$ through validation as $\boldsymbol{\mathcal{Y}}_d = \mathcal{H}(\boldsymbol{\mathcal{X}},\boldsymbol{\theta})$. For each instance, the model provides probability distribution of labels using learned $\boldsymbol{\theta}$. The label with highest probability is assigned to the instance. In general, if the assigned label is the same as true, then its probability should differ substantially from that of other labels. However, this statement holds only if the training dataset does not contain any noisy label. Hence, our method utilizes the distribution instead of only the highest probable label while computing the loss.

5.2.3 *Optimize $\boldsymbol{\theta}$ and $\boldsymbol{\mathcal{Y}}_d$:*

This step aims to optimize $\boldsymbol{\mathcal{Y}}_d$ using Kullback-Leibler (KL) divergence [21] and subsequently fine-tuning the parameters $\boldsymbol{\theta}$ with optimized version of $\boldsymbol{\mathcal{Y}}_d$. Thus, the loss function of the base model, defined Eq. 11, can be replaced by

$$\mathcal{L}\big(\boldsymbol{\mathcal{Y}}_d, \mathcal{H}(\boldsymbol{\mathcal{X}},\boldsymbol{\theta})\big) = \frac{1}{N}\sum_{i=1}^{N} KL(\mathcal{Y}_d^i \parallel \mathcal{H}(\mathcal{X}^i,\boldsymbol{\theta})), \tag{13}$$

$$\text{where} \quad KL(\mathcal{Y}_d^i \parallel \mathcal{H}(\mathcal{X}^i,\boldsymbol{\theta})) = \sum_{c=1}^{C} \mathcal{Y}_d^{i,c} \log\left(\frac{\mathcal{Y}_d^{i,c}}{\mathcal{H}_c(\mathcal{X}^i,\boldsymbol{\theta})}\right).$$

Let us first compute the gradient of $\mathcal{L}(\cdot)$ for all i and c as

$$\frac{d\mathcal{L}\big(\boldsymbol{\mathcal{Y}}_d, \mathcal{H}(\boldsymbol{\mathcal{X}},\boldsymbol{\theta})\big)}{d\mathcal{Y}_d^{i,c}} = 1 + \sum_{c=1}^{C} \log\left(\frac{\mathcal{Y}_d^{i,c}}{\mathcal{H}_c(\mathcal{X}^i,\boldsymbol{\theta})}\right), \tag{14}$$

and then update $\boldsymbol{\mathcal{Y}}_d$ by

$$\boldsymbol{\mathcal{Y}}_d = \boldsymbol{\mathcal{Y}}_d - \eta\,\frac{d\mathcal{L}\big(\boldsymbol{\mathcal{Y}}_d, \mathcal{H}(\boldsymbol{\mathcal{X}},\boldsymbol{\theta})\big)}{d\boldsymbol{\mathcal{Y}}_d}, \tag{15}$$

where η is the learning rate. Once $\boldsymbol{\mathcal{Y}}_d$ stabilizes, the model stops learning. With learned $\boldsymbol{\mathcal{Y}}_d$, the weight parameters $\boldsymbol{\theta}$ are then fine tuned for a fixed number of local iterations usually preset by the server before initializing the training. Finally, the updated weights $\boldsymbol{\theta}$ are collected by the server from all the clients for aggregation. Note that the proposed noise handling method is *adaptive* as it automatically adapts to mislabeled training data without requiring any additional mechanism to correct the labels beforehand.

5.3 Recognition

Given the trained model and a testing TSC (generated from the smart-plug), we first need to extract appliance footprints using the preprocessing steps (discussed in Section 4.2) to prepare input to the model. Let $\boldsymbol{\theta}^*$ be the optimized model, obtained after T global rounds, dispatched from the server to all the clients to predict the class label (or recognize an appliance) using the footprint. Figure 4 illustrates the recognition process using DNN-based model.

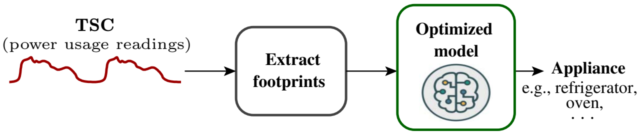


Figure 4: Appliance recognition using the optimized recognition model $\boldsymbol{\theta}^*$.

Let $\mathcal{X}' \notin \mathcal{D}$ be a testing instance (an appliance footprint) for which the class label is to be predicted. The client utilizes the model with $\boldsymbol{\theta}^*$ to first compute the posterior class probabilities and then assigns the highest probable class label to $\mathcal{X}'$ as expressed below

$$\mathcal{Y}' = \underset{y_c}{\operatorname{argmax}}\ \{\sigma_{y_c}(\boldsymbol{\theta}^*\mathcal{X}')\}. \tag{16}$$

Time complexity of our appliance recognition model mainly depends on the operations at the convolutional layers. As the model comprises three layers with identical configurations, the time complexity is $O\big(m(\sum_{i=1}^{3} p_{i-1}\cdot s_i^2\cdot f_i\cdot o_i^2)\big)$ [10], where m is the length of the instance (i.e., appliance footprint), p_{i-1} is the number of input channels, s_i and f_i are respectively the number and size of filters, and o_i is the spatial size of output feature vector at i^{th} layer. As the model is trained across different clients, which can be done in parallel, FedAR+ would impose some communication overheads due to aggregation after each FL round.

6 PERFORMANCE EVALUATION

We evaluate the performance of FedAR+ on a dataset collected from three different households, and also on two existing datasets, namely UK-DALE [12] and Tracebase [27], using accuracy, precision, recall, and F_1-score. To demonstrate the superiority of our proposed approach, we compare it with three existing ones and report the results using the considered metrics and execution time.

6.1 Datasets

6.1.1 *Collected dataset.*

Data collection is done by deploying appliance-specific smart plugs in three different houses for a period of one month. The power consumption data are collected at a sampling rate 1Hz. After preprocessing (discussed in Section 4.2), we get a dataset of 840 instances for six different appliances including refrigerator, microwave oven, television, washing machine, air conditioner, and mixer grinder. Each instance corresponds to a footprint of a particular appliance. We call this dataset as *appliance footprints in residential buildings* (Res-AF).

6.1.2 *UK-DALE dataset [12].*

It contains power consumption data of various household appliances from five houses for a period of one year. The data are recorded at a sampling rate 1/6 Hz. For the experiments, we selected five appliances including refrigerator, washing machine, kettle, dishwasher, and boiler.

6.1.3 *Tracebase dataset [27].*

It contains more than 1000 power consumption traces collected from 15 different houses. One trace corresponds to a time series of readings taken for one particular appliance over a window of 24 hours. The readings are reported at an average sampling rate of 1 Hz. In [27], a Measurement and Actuation Unit (MAU) is developed to collect the power consumption traces of the appliances. MAU is installed between wall mounted power outlet and power plug of the appliance. For experimental evaluation, we selected five appliances having sufficient number of traces (instances) in the dataset. The five selected appliances are refrigerator, microwave oven, kettle, television, and dishwasher.

• **Preprocessing:** In the existing datasets, the readings are collected for a continuous period and thus the resulting time series include both ON and OFF states of the appliances. We therefore preprocess these datasets to prepare them for training and testing the proposed recognition model. By following the preprocessing steps described in Section 4.2, we obtained $1,860$ and 930 instances (i.e., appliance footprints) in UK-DALE and Tracebase datasets, respectively.

6.2 Experimental setup

Prior to conducting experiments, we split each dataset into two parts: training with 80% and testing with 20% instances. The training data are further split into 10 non-iid chunks using Dirichlet distribution with parameter $\alpha = 0.9$ and 20% overlapping. These chunks are then provided to $K = 10$ clients. It is to note that each client may have different number of instances per class due to non-iid data. Further, to produce noisy labels in the training datasets, we flip the labels of a fixed percentage (indicated in the respective results) of instances across all class labels.

We simulated the FedAR+ algorithm with a server and 10 clients in Python programming language through Tensorflow libraries. In the implementation, we chose the following parameters: optimizer = 'sgd', activation = 'relu' with each convolutional layer, and learning rate $\eta = 0.1$. The number of FL global rounds are set based on the results obtained after rigorous experiments (see Section 6.4.1). Since our recognition model consists of only three convolutional layers, each with 128 filters, it does not overfit with small training datasets.

6.3 Performance metrics

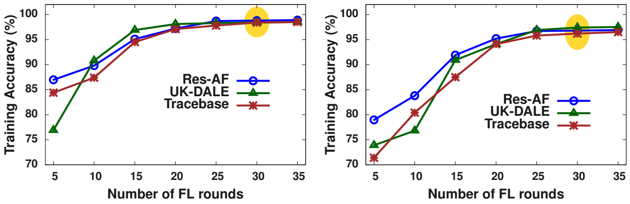
The following metrics evaluate the performance of FedAR+.

- *Precision* (P): It is the ratio of the number of correctly classified instances of an appliance x to the total number of instances classified as x. Precision indicates a quality aspect of the appliance recognition model.
- *Recall* (R): It is the ratio of the number of correctly classified instances of an appliance x to the total number of instances actually belonging to x. Recall measures the completeness and relevance of the recognition model.
- *F_1 score:* It is a harmonic mean of precision and recall, and is computed as $\frac{2\times P\times R}{P+R}$.
- *Accuracy:* It is the percentage of correctly classified instances of the testing dataset.

6.4 Experimental results

Through experiments, we seek answers to the following six questions: (1) How does the accuracy of the model improve over rounds in FedAR+? (2) How does the concentration of noisy labels influence the testing performance? (3) What is the appliance-wise performance of the model? (4) How does FedAR+ scale to the number of clients? (5) How efficiently does FedAR+ outperform the prior approaches? The results are presented below.

6.4.1 Training accuracy over FL rounds. At first, we analyze the training accuracy of the recognition model over 35 rounds with different concentrations (from 5% to 30%) of noisy labels in the training dataset. We set the local epochs to 50 at the clients in all the experiments. Figure 5 demonstrates the results for 5% and 30% cases. The results clearly indicate that the model is able to achieve more than 92% of training accuracy at 30^{th} round, even when 30% training instances are mislabeled. As initial model is far from the optimality in first few rounds, it shows low accuracy for all datasets. The accuracy increases rapidly up to 15^{th} rounds and starts stabilizing afterwards. As no change is observed in the accuracy between 30^{th} and 35^{th} rounds, we report all the subsequent results with 30 FL rounds and 50 local epochs at the clients.

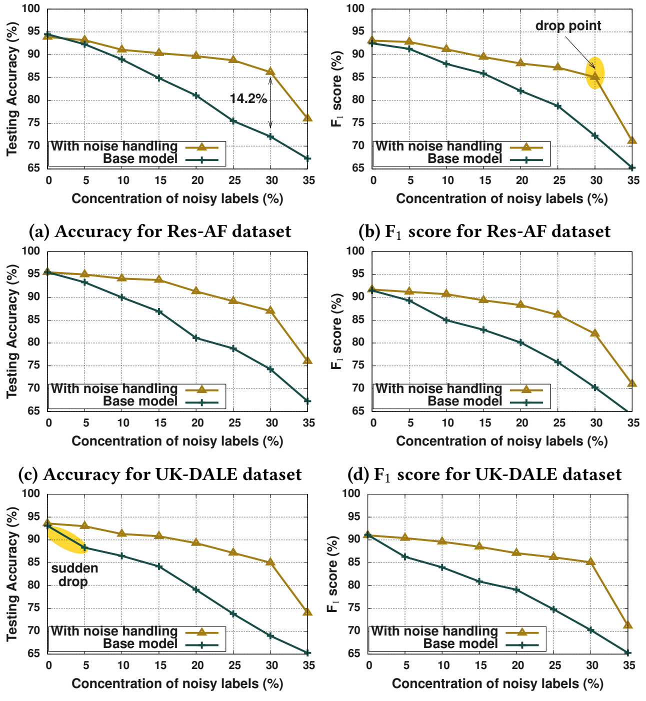


(a) With 5% noisy labels **(b) With 30% noisy labels**

Figure 5: Training accuracy of the proposed appliance recognition model with the noise handling method in FedAR+.

6.4.2 Testing performance with varying concentration of noisy labels. In Figure 6, we report the impact of noisy labels with varying concentrations on the performance of the model. As the concentration of noisy labels increases, the accuracy and F_1 score decrease, which seems bit obvious but, such a drop is substantial for the base model compared to the one with noise handling method. For instance, in case of Tracebase dataset, with just 5% noisy labels, the base model immediately loses more than 4% accuracy. On the flip side, for Res-AF dataset with 30% noisy labels, the model gains 14.2% on accuracy by utilizing the proposed noise handling method, indicating the effectiveness of the proposed approach. For all the datasets, FedAR+ achieves an accuracy of more than 84% and F_1 score of above 81% up to 30% of noisy labels; however, the performance drops sharply afterwards, signaling the noise handling upper limit of our approach. With higher concentrations, the reason for such a drop is the increase in confusion while differentiating between correct and noisy labels. Similar observations can be made from F_1 score, shown in parts (b), (d) and (f) of Figure 6.

6.4.3 Appliance-wise performance of the recognition model. Next, we analyze the appliance-wise performance results of the model with noise handling method for Res-AF and Tracebase datasets in Tables 1 and 2, respectively. The results are reported using precision, recall, and F_1 score evaluation metrics for 5% and 30% concentration of noisy labels. The results indicate that the appliances "television" and "refrigerator" were identified with more than 90% of recall even when the concentration of noisy labels is 30%, witnessing the level of robustness of FedAR+. We also observed that the precision values are marginally ($1 \sim 4$ approx.) differ from the recall ones, indicating the ability of FedAR+ to manage the good balance between the relevance and completeness of the appliance recognition model.



(a) Accuracy for Res-AF dataset (b) F_1 score for Res-AF dataset

(c) Accuracy for UK-DALE dataset (d) F_1 score for UK-DALE dataset

(e) Accuracy for Tracebase dataset (f) F_1 score for Tracebase dataset

Figure 6: Performance results of FedAR+ using the recognition model with and without noise handling method.

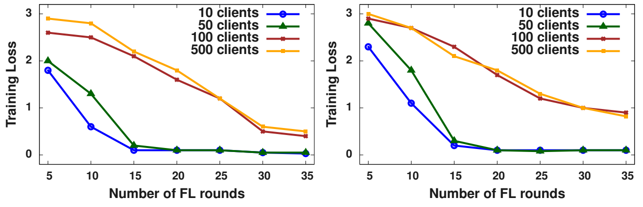
Table 1: Appliance-wise performance of the model with noisy handling method for *Res-AF* dataset using precision (P), recall (R), and F_1 score (F).

	5% Noisy labels			30% Noisy labels		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Refrigerator	90.4	94.0	90.8	90.2	89.8	90.0
Microwave oven	92.8	92.7	93.5	85.3	85.5	85.4
Television	93.1	94.4	94.7	90.5	90.8	90.6
Washing machine	89.0	91.4	89.1	87.2	80.4	87.8
Air conditioner	89.3	92.7	90.7	80.1	82.4	81.2
Mixer grinder	91.3	90.0	90.5	83.8	85.2	84.5
Average	90.8	92.5	91.5	85.8	87.3	86.5

Table 2: Appliance-wise performance of the model with noise handling method for *Tracebase* dataset using precision (P), recall (R), and F_1 score (F).

	5% Noisy labels			30% Noisy labels		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Refrigerator	91.1	94.2	92.6	86.3	91.5	88.6
Microwave oven	88.4	91.2	89.7	81.2	89.1	84.9
Kettle	86.7	88.2	87.4	80.3	85.2	82.6
Television	88.1	93.2	90.5	85.3	92.1	88.5
Dishwasher	87.2	90.1	88.6	81.1	87.9	84.3
Average	88.3	91.4	89.8	82.8	89.2	85.7

6.4.4 Scalability analysis. The scalability of our FedAR+ algorithm can be measured in terms of the number of clients it can support without affecting the performance of the model. We investigate the scalability by increasing the number of clients. Figure 7 shows the training loss of the global model over FL rounds for different number of clients. The model converges (i.e., loss stabilizes) after 20 rounds when only 50 clients exist, however it needs 10 more rounds with 100 to 500 clients because of higher diversity with more number of clients. It indicates that by increasing the number of FL rounds, our approach can be easily scaled to large number of clients without affecting the convergence.



(a) For Res-AF dataset (b) For Tracebase dataset

Figure 7: Training loss of the model in FedAR+ with 0% noisy labels.

Furthermore, we conducted some experiments with 0% and 30% noisy labels and reported the accuracy (obtained by the model trained for 30 rounds) in Table 3 for both Res-AF and Tracebase datasets. The results demonstrate the scalability of FedAR+ to 500 clients with a marginal drop in accuracy that cab be easily recovered by training the model for more FL rounds.

Table 3: Accuracy results for the model trained over 30 rounds.

No. of Clients	Res-AF dataset		Tracebase dataset	
	0% Noisy labels	30% Noisy labels	0% Noisy labels	30% Noisy labels
10	93.9	88.1	93.5	87.3
50	94.1	87.9	92.1	86.2
100	91.8	85.2	91.2	84.8
500	92.5	84.7	91.1	83.3

6.5 Comparison with existing approaches

We compare FedAR+ approach with three state-of-the-art solutions including two best performing plug-load identification models from [28] and Household Appliance Recognition through Bayes classification (HARB) [39]. The work [28] leverages FL to train four different deep learning models; we pick two best performers, long-short term memory (LSTM) and convolutional neural network (CNN), named as LSTM-AR and CNN-AR for the convenience. Similar to the proposed approach, LSTM-AR and CNN-AR models are also trained across 10 clients (possessing non-iid data) over 30 FL rounds with aggregation at every 50 local epochs. On the other hand, HARB [39] follows a central learning paradigm. As the existing solutions do not incorporate any noise handling method, we consider two variants of the proposed approach: 1) FedAR: without noise handling and 2) FedAR+: with noise handling, to make fair comparison. Table 4 shows the comparison results using precision, recall, F_1 score, and accuracy. We make following observations:

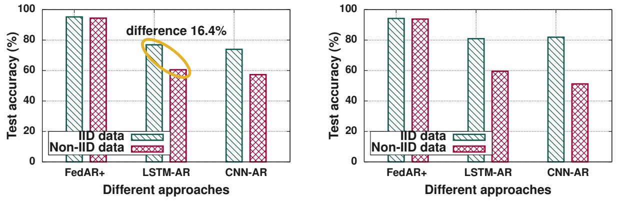
- Both FedAR and FedAR+ gain over the existing solutions on all the evaluation metrics by a large margin (approximately $15\% \sim 27\%$) even in the presence of 30% noisy labels.
- Prior FL models, LSTM-AR and CNN-AR, perform much worse than HARB, indicating their inability to learn from non-iid data. On the contrary, our FL approach FedAR outperforms HARB with a substantial margin of more than 5% in case of no noisy label and more than 12% in case of 30% noisy labels, showing the effectiveness of learning with non-iid data using the aggregation function (defined in Eq. 6).

Table 4: Performance comparison of the proposed approach with the existing ones using accuracy (in %). [P: Precision, R: Recall, F: F_1 score, A: Accuracy]

Datasets	**Approaches**	**Concentration of noisy labels in training data**															
		0%				10%				20%				30%			
		P	**R**	**F**	**A**	**P**	**R**	**F**	**A**	**P**	**R**	**F**	**A**	**P**	**R**	**F**	**A**
Res-AF	HARB [39]	84.2	87.2	85.7	89.5	81.2	83.1	82.1	84.9	72.4	67.2	69.7	70.2	56.1	60.1	58.0	58.6
	LSTM-AR [28]	60.2	59.2	59.7	60.5	59.2	55.2	57.1	60.9	52.2	50.2	50.7	51.1	43.2	45.7	44.4	46.6
	CNN-AR [28]	55.2	53.4	54.2	57.3	52.4	52.3	52.3	50.8	45.1	44.1	43.6	47.3	48.2	41.1	42.1	41.7
	FedAR (Proposed)	92.4	93.8	93.1	94.4	90.1	87.5	88.8	89.1	82.1	79.5	80.8	81.5	73.6	72.4	73.0	73.4
	FedAR+ (Proposed)	**93.1**	**95.1**	**94.1**	**94.8**	**91.2**	**89.5**	**90.3**	**92.3**	**88.5**	**87.1**	**87.8**	**89.5**	**87.2**	**85.7**	**86.4**	**88.4**
UK-DALE	HARB [39]	87.6	89.5	88.5	90.5	85.7	81.2	83.4	83.5	70.2	68.6	69.4	72.5	60.2	56.1	58.1	59.2
	LSTM-AR [28]	59.1	58.9	59.0	60.5	56.3	59.4	57.8	58.9	45.6	47.3	45.9	43.2	43.2	41.1	42.1	43.6
	CNN-AR [28]	56.1	50.1	53.0	54.8	50.6	59.2	59.9	51.2	43.5	43.2	41.3	42.2	36.4	38.1	37.7	41.9
	FedAR (Proposed)	92.4	91.2	91.8	94.9	87.2	89.2	88.2	90.3	75.2	76.1	75.6	81.1	76.2	70.3	73.1	74.2
	FedAR+ (Proposed)	**94.1**	**90.5**	**92.3**	**95.1**	**94.8**	**90.2**	**92.4**	**93.5**	**86.4**	**87.7**	**87.0**	**90.8**	**84.2**	**80.9**	**82.5**	**87.2**
Tracebase	HARB [39]	87.4	88.1	87.7	90.8	78.6	82.1	80.3	80.3	71.5	68.2	69.8	70.3	58.2	52.1	55.0	57.4
	LSTM-AR [28]	56.2	60.3	58.2	59.5	58.2	60.2	59.2	59.9	50.5	51.2	49.3	51.2	45.2	45.4	45.0	42.6
	CNN-AR [28]	49.2	50.8	50.0	51.2	48.7	52.4	50.5	50.5	43.6	38.7	41.1	43.3	37.4	36.9	40.1	39.2
	FedAR (Proposed)	94.2	92.8	**93.5**	93.5	91.7	90.7	91.2	90.2	80.5	78.7	79.6	80.8	70.1	71.9	71.0	72.7
	FedAR+ (Proposed)	**94.6**	**91.4**	**93.0**	**93.8**	**93.7**	**90.5**	**92.1**	**92.9**	**89.2**	**86.3**	**87.7**	**88.9**	**87.5**	**84.3**	**85.9**	**86.1**

- Even with 30% mislabeled data, FedAR+ secured the accuracy and F_1 of more than 85% on all the datasets, validating the success of our noise handling method. It is worth to notice that the performance gain of FedAR+ over other methods increases significantly with the surge in noisy labels.

6.5.1 IID versus non-IID data. Considering 10 clients in FL setup, we now report the test accuracy results in Figure 8 with iid (or uniformly) and non-iid data (simulated using Dirichlet distribution). To make fair comparison, this experiment does not include noisy labels. For both datasets, FedAR+ shows its capability to learn with non-iid data by achieving almost equal accuracy as with iid data, however it is not true for prior approaches; for instance, LSTM-AR loses 16.4% accuracy when clients possess non-iid training data. Although the performance of prior approaches seem to improve significantly with iid data, they could never reach beyond 81%, which is easy to observe from part (b) of the results.



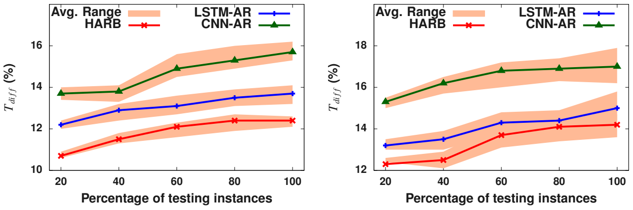
(a) For Res-AF dataset **(b) For Tracebase dataset**

Figure 8: Comparing accuracy results obtained with iid and non-iid data (with 0% noisy labels) across clients in FL based approaches.

6.5.2 Execution time. Finally, we compare the execution time of FedAR+ with the existing approaches. Here, the execution time indicates the total time taken by an approach to classify the entire testing dataset. To better understand the comparison, we compute the percentage difference (T_{diff}) in the execution times of any existing approach from FedAR+, as follows:

$$T_{diff} = \frac{T_x - T_{FedAR+}}{T_x} \times 100,$$

where T_x denotes the execution time of an existing approach x. Figure 9 shows the comparison results in terms of T_{diff}, which is an average over 50 executions. It is apparent that T_{diff} is positive in all the cases, indicating that FedAR+ is faster (10.5% to 16.8%) than the existing approaches.



(a) For Res-AF dataset **(b) For Tracebase dataset**

Figure 9: Comparison results using percentage difference in the execution time of the existing approaches from FedAR+.

7 CONCLUSION AND DISCUSSIONS

This paper proposed an FL approach, FedAR+, for identifying household appliances using their electricity consumption patterns. The approach dealt with two important issues related to the appliance recognition model: 1) presence of noisy labels in the training dataset, and 2) model building at the client (consumer) side without sharing local data to the server. By employing deep learning and incorporating a noise handling method, we developed an accurate appliance recognition model that can learn from mislabeled training data. By deploying smart plugs in an apartment complex, we collected a real world dataset to validate the effectiveness of FedAR+. Through rigorous experimental analysis, we demonstrated the superiority of FedAR+ over existing ones and showed that it can effectively accommodate up to 30% noisy labels while compromising the accuracy only slightly. Considering the availability of sufficient training data, FedAR+ can be scaled to large number of clients, enabling its adoption to real-world energy monitoring applications.

In future, we plan to work on theoretical guarantees of FedAR+ and scale the solution to distinguish different operating states of the appliances. We will also explore the robustness aspects of the model under the presence of alien and malicious clients.

Alien appliance: An alien (unseen) appliance is one for which there exists no instance in the training dataset to build the recognition model. Identifying alien appliances is an interesting problem as it gives flexibility to the consumer to introduce new household appliances without providing any additional information to the service provider. We plan to develop effective strategies for extracting semantic information, thereby enhancing the capability of our recognition model in FedAR+.

Malicious client: A client with wrong intention may try to attack the model by altering its weight parameters when transmitted from the client to server. Such malicious client can affect the performance of the global model in FedAR+. We plan to detect the malicious clients who send incorrect parameters by either adding random-noise or backdoor patterns in the dataset. Our idea is to exploit the history of each client' gradients with an appropriate similarity measures (e.g., cosine distance) to distinguish malicious clients from the normal ones and exclude their parameters from the aggregation.

ACKNOWLEDGEMENTS

This work is partially supported by the NSF grant award #2008878 (FLINT: Robust Federated Learning for Internet of Things) and the NSF award #2030624 (TAURUS: Towards a Unified Robust and Secure Data Driven Approach for Attack Detection in Smart Living).

REFERENCES

[1] A. Allik, S. Muiste, and H. Pihlap. 2020. Smart meter data analytics for occupancy detection of buildings with renewable energy generation. In *9th International Conference on Renewable Energy Research and Application.* 248–251.

[2] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He. Short-term load forecasting with deep residual networks. *IEEE Transactions on Smart Grid* 10, 4 (2019), 3943–3952.

[3] J. Codispoti, A. R. Khamesi, N. Penn, S. Silvestri, and E. Shin. Learning from Non-experts: An Interactive and Adaptive Learning Approach for Appliance Recognition in Smart Homes. *ACM Transactions on Cyber-Physical Systems* 6, 2 (2022), 1–22.

[4] M. Farrokhifar, F. Momayyezi, N. Sadoogi, and A. Safari. Real-time based approach for intelligent building energy management using dynamic price policies. *Sustainable cities and society* 37 (2018), 85–92.

[5] T. Ganu, D. Rahayu, D. P. Seetharam, R. Kunnath, A. P. Kumar, V. Arya, S. A. Husain, and S. Kalyanaraman. 2014. SocketWatch: an autonomous appliance monitoring system. In *IEEE International Conference on Pervasive Computing and Communications (PerCom).* 38–43.

[6] B.-B. Gao, C. Xing, C.-W. Xie, J. Wu, and X. Geng. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing* 26, 6 (2017), 2825–2838.

[7] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep learning.* MIT press.

[8] J. Han, P. Luo, and X. Wang. 2019. Deep self-learning from noisy labels. In *IEEE/CVF international conference on computer vision (ICCV).* 5138–5147.

[9] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[10] K. He and J. Sun. 2015. Convolutional neural networks at constrained time cost. In *IEEE conference on computer vision and pattern recognition.* 5353–5360.

[11] D. Jang, L. Spangher, T. Srivistava, M. Khattar, U. Agwan, S. Nadarajah, and C. Spanos. 2021. Offline-online reinforcement learning for energy pricing in office demand response: lowering energy and data costs. In *8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys).* 131–139.

[12] J. Kelly and W. Knottenbelt. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Scientific Data* 2, 150007 (2015).

[13] W. Kleiminger, C. Beckel, and S. Santini. 2015. Household occupancy monitoring using electricity meters. In *ACM international joint conference on pervasive and ubiquitous computing (UbiComp).* 975–986.

[14] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau. 2019. Federated learning for keyword spotting. In *IEEE International Conference on Acoustics, Speech and Signal Processing.* 6341–6345.

[15] Q. Li, J. Wang, Z. Yao, Y. Li, P. Yang, J. Yan, C. Wang, and S. Pu. 2022. Unimodal-Concentrated Loss: Fully Adaptive Label Distribution Learning for Ordinal Regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 20513–20522.

[16] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li. 2017. Learning from noisy labels with distillation. In *IEEE/CVF international conference on computer vision (ICCV).* 1910–1918.

[17] K. J. Liang, S. B. Rangrej, V. Petrovic, and T. Hassner. 2022. Few-shot learning with noisy labels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 9089–9098.

[18] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang. 2020. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI Conference on Artificial Intelligence*, Vol. 34. 13172–13179.

[19] Y. Liu, X. Wang, and W. You. Non-Intrusive Load Monitoring by Voltage-Current Trajectory Enabled Transfer Learning. *IEEE Transactions on Smart Grid* 10, 5 (2019), 5609–5619.

[20] M. Ma, W. Lin, J. Zhang, P. Wang, Y. Zhou, and X. Liang. Toward Energy-Awareness Smart Building: Discover the Fingerprint of Your Electrical Appliances. *IEEE Transactions on Industrial Informatics* 14, 4 (2018), 1458–1468.

[21] D. J. MacKay and D. J. Mac Kay. 2003. *Information theory, inference and learning algorithms.* Cambridge university press.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics (AISTATS).* 1273–1282.

[23] J. Nocedal and S. Wright. 2006. *Numerical optimization.* Springer Science & Business Media.

[24] K. Paridari, A. E.-D. Mady, S. La Porta, R. Chabukswar, J. Blanco, A. Teixeira, H. Sandberg, and M. Boubekeur. 2016. Cyber-physical-security framework for building energy management system. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS).* IEEE, 1–9.

[25] N. B. S. Qureshi, D.-H. Kim, J. Lee, and E.-K. Lee. 2022. Poisoning Attacks against Federated Learning in Load Forecasting of Smart Energy. In *IEEE/IFIP Network Operations and Management Symposium.* 1–7.

[26] N. Rajagopal, S. Giri, M. Berges, and A. Rowe. 2013. A magnetic field-based appliance metering system. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems (ICCPS).* 229–238.

[27] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz. 2012. On the accuracy of appliance identification based on distributed load metering data. In *Sustainable Internet and ICT for Sustainability (SustainIT).* 1–9.

[28] R. Schwermer, J. Buchberger, R. Mayer, and H.-A. Jacobsen. 2022. Federated office plug-load identification for building management systems. In *13th ACM International Conference on Future Energy Systems.* 114–126.

[29] W. T. Soe and C. Belleudy. 2019. Load Recognition from Smart Plug Sensor for Energy Management in a Smart Home. In *IEEE Sensors Applications Symposium (SAS).* 1–6.

[30] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[31] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. 2018. Joint optimization framework for learning with noisy labels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 5552–5560.

[32] United States Energy Information Administration (US EIA). [n.d.]. How much energy is consumed in U.S. buildings? https://www.eia.gov/tools/faqs/faq.php?id=86&t=1 Accessed: 10 Apr 2021.

[33] V. Vadakattu and S. Suthaharan. 2018. Feature extraction using apparent power and real power for smart home data classification. In *17th IEEE International Conference on Machine Learning and Applications (ICMLA).* 1290–1295.

[34] A. F. d. S. Veloso, R. G. de Oliveira, A. A. Rodrigues, R. A. Rabelo, and J. J. Rodrigues. 2019. Cognitive Smart Plugs for Signature Identification of Residential Home Appliance Load using Machine Learning: From Theory to Practice. In *IEEE International Conference on Communications (ICC) Workshops.* 1–6.

[35] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. 2019. Symmetric cross entropy for robust learning with noisy labels. In *IEEE/CVF international conference on computer vision (ICCV).* 322–330.

[36] Z. Wang, G. Hu, and Q. Hu. 2020. Training Noise-Robust Deep Neural Networks via Meta-Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 4524–4533.

[37] Z. Wang and H. Wang. 2021. Improving load forecast in energy markets during COVID-19. In *8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys).* 168–171.

[38] N. Xu, J.-Y. Li, Y.-P. Liu, and X. Geng. Trusted-Data-Guided Label Enhancement on Noisy Labels. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[39] D. Yan, Y. Jin, H. Sun, B. Dong, Z. Ye, Z. Li, and Y. Yuan. Household appliance recognition through a Bayes classification model. *Sustainable Cities and Society* 46 (2019), 101393.

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