PFDRL: Personalized Federated Deep Reinforcement Learning for Residential Energy Management

Jiechao Gao, Wenpeng Wang, Fateme Nikseresht, Viswajith Govinda Rajan, Bradford Campbell
University of Virginia, Charlottesville, VA, United States of America
{jg5ycn,ww2cg,fn5an,gyx4bw,bradjc}@virginia.edu

ABSTRACT

The rise of the Internet of Things (IoT) has increased standby energy consumption due to the growing number of smart devices in homes. Existing approaches use real-time energy data and machine learning to identify and minimize standby energy for residential energy management but rely on cloud-based data aggregation and collaborative training due to limited edge device data. However, such an approach incurs extra cloud service costs, risks personal data leakage, and fails to capture residence diversity, resulting in suboptimal energy management performance.

In this paper, we propose a privacy-preserving and cloud-service-free residential energy management system (EMS) that utilizes personalized federated deep reinforcement learning (PFDRL) to reduce household standby energy consumption. PFDRL consists of three components: First, we develop a decentralized federated learning (DFL) framework instead of using a centralized cloud service to aggregate the model to keep both the data and the model in the local area. Second, we apply DFL with deep reinforcement learning (DRL) to share the EMS plan among local residences for collaborative training. Third, we divide the neural network in the DRL into two parts, base layers and personalization layers to enhance model convergence while maximizing EMS performance for each client in the system. We evaluate the proposed PFDRL framework on the real-world Pecan Street dataset [3], demonstrating superior performance compared to centralized settings and conventional solutions.

CCS CONCEPTS

• **Computer systems organization** → *Neural networks*; • **Security and privacy** → **Privacy protections**.

KEYWORDS

Data Privacy, Personalized Federated Learning, Reinforcement Learning

ACM Reference Format:
Jiechao Gao, Wenpeng Wang, Fateme Nikseresht, Viswajith Govinda Rajan, Bradford Campbell. 2023. PFDRL: Personalized Federated Deep Reinforcement Learning for Residential Energy Management. In *52nd International Conference on Parallel Processing (ICPP 2023), August 7–10, 2023, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3605573.3605641

1 INTRODUCTION

Over the past few years, an increasing number of IoT devices have been installed in residential and commercial buildings to satisfy the increased demand for users' comfort, well-being, and quality of life. However, this increase in buildings' convenience, functionality, and connectivity requires producing more energy which raises monetary costs and harmful gas emissions. In 2019, residential and commercial buildings were responsible for 40% of U.S. carbon emissions [19]. Standby energy is one of the reasons that cause the increasing energy consumption. By 2019, standby energy represents approximately 10% of residential electricity use in most developed countries and a rising fraction in developing countries [17]. Previous work [30] has investigated the major causes of increased standby energy consumption of smart devices, and for the most part, energy is just wasted by devices waiting for commands.

Accurately distinguishing devices' standby energy usage from regular usage is essential for achieving standby energy reduction in residential buildings. [13]. Collaborative training of collected energy data from multiple residents can lead to improved performance in identifying standby energy consumption, which is especially important as individual users may not have enough data to train a reliable model [15]. Traditional approaches [6, 25] require collecting device-level energy data for load forecasting using machine learning and deep learning methods to identify standby energy, then making energy management plans based on the prediction result. This prediction usually happens in the cloud. Although cloud platforms can bring high computational power and aggregate data from different users, they introduce the critical risk of potential personal data leakage [4], such as the central server becoming a malicious party.

Federated learning (FL) is an emerging tool to address privacy issues, which allows IoT devices to collaboratively train a model. Existing approaches [5, 27] present a similar framework for load forecasting and energy management. However, such FL-based frameworks still face three problems: (1) Despite storing data locally, the FL framework still requires cloud support to aggregate a global model that contains all the training information from different local IoT devices. The model remains vulnerable to training data recreation attacks by model inversion and faces the potential risk of data leakage [12]. The cost of cloud service can also disincentivize users from participating in the energy management system. (2) Localized EMS may take a prolonged time to converge due to insufficient training examples. Despite load forecasting being done through FL, the performance of local EMS in achieving optimal energy management is still challenging and time-consuming. (3) Existing FL paradigms for EMS training produce a singular global model for all residents. Realistically, energy data residing across devices is inherently statistically heterogeneous (i.e., non-IID distribution). The diversity of users' data can lead to model convergence delays and suboptimal EMS performance for certain residents.

In this paper, we propose a framework where residents can share their energy management plans while achieving a privacy-preserved and cloud-service-free residential EMS with a personalized federated deep reinforcement learning (PFDRL) framework. Instead of sharing just load forecasting results to identify standby energy, our framework also allows residents to share their energy management plans to collaboratively train their EMS by aggregating the reinforcement learning agents.

First, we introduce a decentralized federated learning (DFL) framework to enable distributed edge devices in the residential area to collaboratively train a model without a cloud server. The training parameters from each local model are broadcasted and aggregated between the smart home agents owned by each residence at a certain frequency. This removes the need of using a central cloud server, and reduces the possibility of data leakage (e.g., malicious cloud server) and monetary cost from cloud service. Second, we applied deep reinforcement learning (DRL) with DFL to share the EMS plan to enhance the time required to attain optimal EMS performance. We also use the load forecasting result as an input feature in the DRL framework to help with decisive RL action. Third, we further introduce personalized federated DRL (PFDRL) to handle the energy data heterogeneity and maximize the residential-level EMS performance as well as speed up the model converging time. We dynamically divide the network inside the DRL into base layers and personalized layers. Our training algorithm comprises the base layers being trained in a federated fashion and personalization layers being trained only from local data with stochastic parameter descent. We evaluate the proposed PFDRL on the real-world Pecan Street dataset [3] and compare our results with four different types of EMS. Experimental results show that the proposed framework can achieve 92% load forecasting accuracy and save 98% of total standby energy consumption in a day which outperforms all other methods. We summarize the contributions of this paper as follows:

(1) We propose a decentralized FL framework that allows the processing of all collected data locally on the edge, and only the model parameters are transferred among the network, which removes the need to use cloud service in a residential area.
(2) We proposed a federated reinforcement learning based framework to reduce standby energy for each residence in a residential area. Through sharing EMS plans, this framework can achieve optimal EMS performance in a short time.
(3) We proposed a personalization aspect in federated learning by treating the neural networks in deep reinforcement learning models as a combination of base and personalization layers. The base layers serve as shared layers while the personalization layers are trained locally, allowing for the capture of personal information from IoT devices.
(4) We conduct experiments on real-world energy dataset. The results demonstrate that our proposed method outperforms other comparison methods in energy reduction and achieves the fastest convergence speed.

2 RELATED WORK

2.1 Load Forecasting & Energy Management

Various works in load forecasting majorly focus on two different aspects: aggregated household-level forecasting and device-level forecasting. Since device-level forecasting suffers from high volatility and uncertainty of device usage that is not significant on aggregated loads, their major approaches are different. Kong et al. [16] introduced a centralized density-based clustering technique to evaluate and compare the inconsistency between the aggregated load and individual loads, then they adopted Long Short-Term Memory (LSTM) and designed a load forecasting framework for individual residential households. However, they require all data to be transmitted to a central hub, which can cause privacy concerns regarding sensitive user data. Din et al. [8] introduced a single device level load forecasting based on the Deep Neural Network approach, which they do not require the collection of lengthy historical data. However, their approach failed to conduct any load control and data privacy, which is a key contributor to our research.

2.2 Data Privacy & Federated Learning

Only a few works in home energy management systems have considered data privacy [10, 21, 34]. Most of their work aims to deal with the tradeoffs between energy data privacy and energy costs. For example, Yang et al. [34], developed an online control algorithm using the Lyapunov optimization technique to balance the problem between cutting down electricity bills and keeping the privacy of load requirements and electricity bill processes. A recent trend in maintaining user privacy is using federated learning approaches, where each agent processes its own collected data locally, and only transmits the calculated parameter to a central node. This reduces the amount of data transmission and preserves user privacy. There have been several papers [18, 29] using federated learning. Lee et al. [18] propose a novel federated reinforcement learning (FRL) approach for the energy management of multiple smart homes with home appliances, a solar photovoltaic system, and an energy storage system. However, the current FL focus on building a global model instead of a personalized model to maximize the energy management performance for each client. On the other hand, FL still utilizes cloud service to aggregate the model, which can be malicious and vulnerable and lead to personal data leakage from model inversion.

2.3 Energy Management Personalization

To better serve a large number of users with custom settings, personalized federated learning is proposed. In the FL approaches proposed in [9, 11, 31], the lower layers between all users are shared to the cloud server while keeping several user-specific upper layers on the edge. In this design, the more general features are saved on the lower layers while the upper layers capture a higher level of abstraction, which contains more user-specific features. However, these personalized methods rely on a pre-defined structure of model sharing, thus limiting the optimization of performance for each user. In this paper, we added a parameter α that determines the number of layers to be transmitted, thus can optimizes the performance for every single user. Other personalization energy management systems for residential buildings focus on personalized scheduling and demand response, with many using Bayesian networks for decision making [23]. However, these methods rely on the full support of the cloud service, which might be malicious and bring data privacy issues.

3 SYSTEM DESIGN

3.1 System Overview

We present a personalized federated deep reinforcement learning (PFDRL) framework, which contains two parts: a decentralized federated learning (DFL) framework and a personalized deep federated reinforcement (PRL) framework. Figure 1 shows the structure of the proposed system.

Our proposed DFL framework removes the requirement for cloud services. First, each agent collects the device load data from every IoT device deployed in its local residence and trains a load forecasting model with the same machine learning method. Second, the agents broadcast the parameters of each device at a certain time frequency β, so that each agent has the parameter information from the same kind of device in other residences. Third, the load forecasting result is calculated based on the updated model. After each agent updates its local model by aggregating the qualified parameter for each device, the agent predicts the future power draw for each device to identify standby energy. In the meantime, the devices are still recording load data for the next training phase, and this process happens at the same interval, by default hourly. Fourth, the output of DFL, which is the load forecasting result, will be used as the input for the PFL framework along with the real-time load data. The proposed PRL framework will optimize the percentage of the base layer and personalized layer to achieve the best performance for each client in the system by the performance optimization parameter α. The base layer will be broadcast to other agents. The PFL framework is deployed on each agent in each resident. The load forecasting result can reflect the predicted mode for each device; the real-time load data can reflect the current mode for each device. Such information will be fed to the DRL agent to take the standby energy-reducing actions, such as switching the IoT device mode from standby to off.

3.2 Decentralized Federated Learning for Load Forecasting

Traditional distributed machine learning techniques require a certain amount of private data to be aggregated and analyzed at central servers (e.g., cloud servers) during the model training phase using distributed stochastic parameter descent (DSGD). Such a training process suffers from potential private data leakage risks. To address such privacy challenges, a collaboratively distributed machine learning paradigm, called federated learning (FL), was proposed for edge devices to train a global model while keeping the training datasets local and without sharing raw training data. However, traditional FL requires a cloud aggregator to obtain the global model by aggregating sparse parameters and sending this global model to the local agent. Such a method will create a global model in the cloud which suffers from not only potential private data leakage risks but expensive communication costs caused by the cloud.

In this paper, we focus on solving the above issue in a residential building. We remove the need for a central server by allowing the residents to process all collected data locally on edge and broadcasting the model updates between the smart home agent in each residence inside the residential building. Such setup has the following benefits: (1) Remove the need

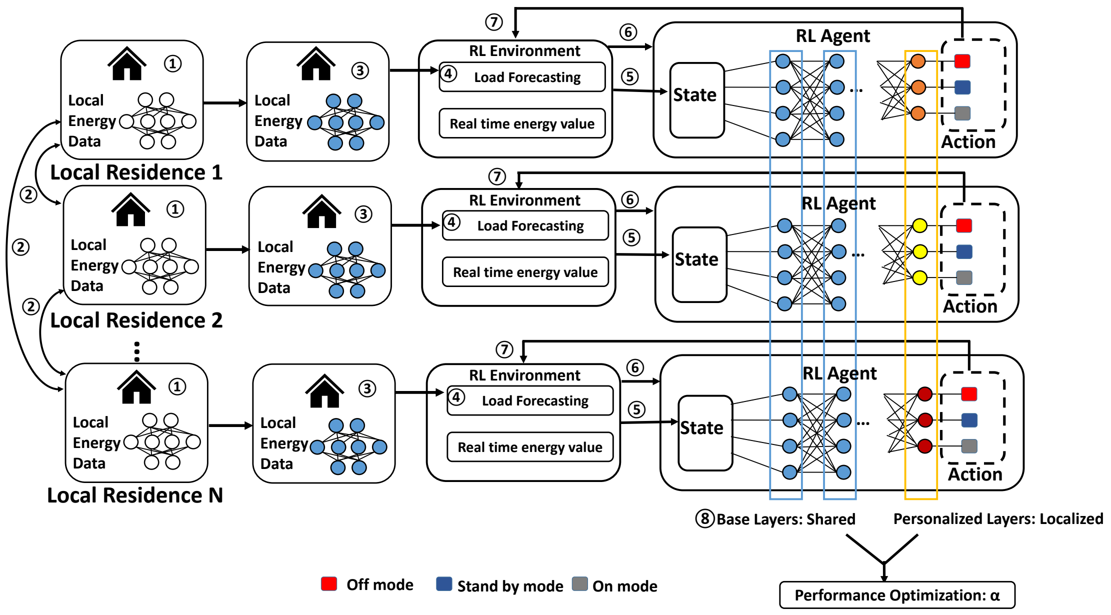

Figure 1: Overview of the PFDRL Framework: ① Local energy data collection and training for standby energy identification; ② Parameters broadcasting at certain time frequency β between each residence; ③ Local testing with the updated model; ④ Feed load forecasting result together with real-time energy value as deep reinforcement learning environment; ⑤ State observation; ⑥ Reward calculation; ⑦ Action selection to determine device mode; ⑧ Divide neural networks inside the deep reinforcement learning models as base and personalization layers using optimization parameter α. Broadcast base layers at a certain time frequency γ and keep the personalization layers locally. Each residence will use the updated PFDRL framework to perform energy management locally.

for cloud service, which will save extra monetary cost from cloud usage, and avoid the possibility of a malicious central server. (2) The model information will only be broadcasted inside the residential building, which we believe is much easier to operate and privacy-preserved for residences.

3.2.1 ***Locally Training Process:*** In our system, we consider a residential home that includes N residents. Each residence n has an agent such as Google Home or Amazon Alexa, which has the connection between the IoT devices in certain residence n, where $n \in \{1, 2, ..., N\}$. We denote A_n as the agent in the system, which represents its residence n. For each agent A_n, we have the same default training model initially, such as Long Short Term Memory (LSTM). We denote D_{X_n} as different IoT devices, in different residents, and X refers to the type of certain IoT device. Since each IoT device, D_{X_n} has its own local dataset (i.e., sensing time-series data from IoT nodes), we train the model separately for each device on the connected agent. For example, the TV in residence one, we define as D_{TV_1}, TV in residence two, we define as D_{TV_2}, the lighting in residence one, we define as D_{light_1}, etc. We train the model for each device in each residence on the connected agent locally and separately. Thus every device has a certain parameter. For example, a parameter G_{TV_1} is calculated for the TV in residence one in a certain time period. In each resident's home, an agent will record the parameter for all the resident's devices. As an instance, for residence one, the agent A_1 has the parameter information G_{TV_1}, G_{light_1}, G_{HVAC_1} and all the other devices that are connected to A_1.

Given a local dataset recorded by an IoT device D_{X_n}, our goal is to predict the future energy consumption for this certain device for the following hour. We choose different machine learning and deep learning models for the training set to learn the usage pattern of the device, and the testing set is used to predict the estimated energy consumption after the parameter aggregation. We also select the prediction method with the best performance for the following step. For each device D_{X_n}, we first predict the energy consumption V_{X_n} in every minute for the next hour. Then, we calculate and record the parameter for each device D_{X_n} locally, respectively, for broadcasting. To avoid high frequency broadcasting, we set β hours as the parameters broadcast rate to reduce the frequency of broadcasting parameters. In our experiment, we will determine the hyperparameter β that has the best result.

3.2.2 ***Parameter Aggregation****.* After each agent A_n determines the selected parameters from other agents for each device D_{X_n}, each agent will update the model with such parameters G_{X_n}. To do so, we use DSGD for iterative updates, and the loss function to be optimized is defined as follows:

$$F(w) = \frac{1}{D_{X_n}} \sum_{n \in D_{X_n}} f(n, w) \tag{1}$$

where $F(w)$ is the loss function for the updated model, $f(n, w)$ is the loss function for the previous model, and w is the model's weight.

In the parameter aggregate phase, the agents obtain an updated model w_{t+1} for the next iteration as follows:

$$w_{t+1} = w_t - \eta \frac{1}{Nb} \sum_{n=1}^{N} \sum_{x \in B_n, k} \triangledown f(x, w_t) \tag{2}$$

where η is the learning rate, B_n, k is the data sample for the $k_t h$ round of training, and each local dataset size of b. All the collaborated agents repeat the above process until the model reaches convergence. Then we use the updated model to predict the energy consumption for a certain device for the next hour.

Algorithm 1 shows the training process of DFL load forecasting. Each home agent initialized its load forecasting model with the same structure for each device. The parameter for each model is set as random. For each device D_{X_n}, first, it locally trains its own model and finds the convergence $W_{n,t}$. After all models are converged, the smart home agent will broadcast the fine-turned model parameters to all other smart agents from other residences. The broadcast frequency β will be determined from our experiment. Upon receiving all W_s, each agent aggregates the model parameters locally and updates its own model. The load forecasting result for standby energy identification is also predicted using the updated model locally.

Algorithm 1: DFL load forecasting Algorithm

```
Initialize load forecasting model for each device $D_{X_n}$;
Initialize model weight $W_{n,0}$ at random;
for $n$ =0 to $N$ do
    for $t$ =0 to $T$ do
        $W_{n,t} \leftarrow$ SGD($W_{n,t-1}, \eta$); (Local Training Step)
        Check if each device $D_{X_n}$ finished local
          training;
        Broadcast $W_{n,t}$ to other residences;
        Receive $W_s$ from all other residences;
        $W_{n,t+1} \leftarrow \sum_{n=1}^{N} \frac{W_{n,t}}{N}$
    end
    Update local model with $W_{n,t+1}$;
    Localized load forecasting
end
```

3.3 Personalized Federated DRL For Energy Management

After the load forecasting is made from the DFL framework, each DRL agent A_n needs to decide whether the mode of a certain device D_{X_n} should be changed or not. In the DRL framework, the action is made every minute based on the result from the DFL framework. So, the DRL agent performance is highly influenced by the DFL load forecasting accuracy. In our system, we first formulate this problem as a Markov Decision Process (MDP) and use a reinforcement learning (RL) method based on Deep Q-Network (DQN) as our base energy management system. Then, we divide the neural network inside the DRL as base layers and personalization layers. We broadcast the base layers to other residences and keep the personalization layers locally. The combined model with global aggregated base layers and localized personalization layers can achieve a collaborative training process as well as better energy management results for each residence.

3.3.1 ***DRL for energy management:*** Reinforcement learning is a method where at each time step t, an agent observes a state s_t in an environment, takes an action a_t based on the observation, and receives a positive or negative reward r_t for the action taken. The objective of the DRL agent is to find an action policy π that would maximize the expected cumulative reward $[\sum_{t=0}^{\infty} r_t]$.

We first formulate this problem as a Markov Decision Process (MDP), denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state, $\mathcal{A}$ is the action, $\mathcal{P}$ is the probability between each two states and $\mathcal{R}$ is the reward. Below, we introduce the elements in the MDP for our problem.

State Space: The state space $\mathcal{S}$ is defined as the input of the DRL model. For each device D_{X_n}, it has three operation modes: off, standby, and on. Each mode can be reflected by energy consumption. In our system, the state space consists of two separate parts: The first part is the predicted energy consumption, which reflects the predicted device mode. The second part is the real-time energy consumption, which reflects the real device mode.

We denote the predicted energy consumption by:

$$\mathcal{V} = \{V_1, V_2, ...V_t, ...V_T\} \tag{3}$$

where V_t means the t^{th} predicted energy consumption and T is the total number of predicted energy consumption. Since in the DFL prediction phase, each prediction is made for the next hour, the total number T is set as 60 minutes. The real-time energy consumption is recorded at the same timestamp as the predicted value:

$$\mathcal{RV} = \{RV_1, RV_2, ...RV_t, ...RV_T\} \tag{4}$$

where the value of t for RV_T is the same as V_t.

For each device D_{X_n}, it has a default value for each operation mode where V_{off} means the device is off, V_s means the device is in standby mode, V_{on} means the device is in on mode. For each RV_T and V_t, if the value is 0, we define the predicted and real mode of a certain device as off mode. If the value is between $0.9 * V_s$ and $1.1 * V_s$, we define the predicted and real mode of a certain device as in standby mode. If the value is between $0.9 * V_{on}$ and $1.1 * V_{on}$, we define the predicted and real mode of a certain device as on mode [14]. We denote S as predicted mode for certain device, where $S \in \{ S_{off}, S_s, S_{on}\}$ as off, standby and on mode. We denote RS as the real mode for certain device, where $RS \in \{RS_{off}, RS_s, RS_{on}\}$ as off, standby and on mode.

Action Space: The action space $\mathcal{A}$ is defined as the agent A_n can make decisions on whether the mode of a certain device D_{X_n} should be changed or not at time t. After receiving the predicted and real mode of certain devices, the action a_t is expressed in the following:

$$a_t = \begin{cases} 0, & \text{Off mode} \\ 1, & \text{Standby mode} \\ 2, & \text{On mode} \end{cases} \tag{5}$$

Probability: The state space is changed with certainty, so the probability between states is always 1.

Reward: Based on action space $\mathcal{A}$, the DRL agent receives a reward $r(t)$ for each action taken at time t. In the system, the agent can receive positive rewards in one scenario: the predicted mode and the real mode for a certain device are the same at the time. In this case, the reward is set as 10. On the other hand, the agent can receive negative rewards in two scenarios: first, the predicted mode and the real mode for a certain device are different at time t, and the predicted mode should be moved up or down for one mode. In this case, the reward is set as -10. Second, the predicted mode and the real mode for a certain device are different at time t, and the predicted mode should be moved up or down for two modes. In this case, the reward is set as -30. An exception for the reward is that we wish once the real mode is standby, we want to change it to off mode, so we mark the reward as 30 in this scenario. Based on the aforementioned statement, we define reward r_t at time t for action a_t as shown in table 1.

Q-value calculation: The agent takes the action a_t that satisfies $\max Q(s_t, a_t)$ and receives a reward $r(s_t, a_t)$. The goal is to find a policy π that will approximate the optimal Q-function $Q^*(s_t, a_t)$ which always satisfies Bellman's optimality equation. That is,

$$Q^*(s_t, a_t) = r(s_t, a_t) + \kappa \cdot \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \tag{6}$$

where, $Q^*(s_{t+1}, a_{t+1})$ is the next-step's optimal Q value.

As there will also be new load data coming from each device, we keep training our DRL agent to achieve the best performance over time. After the first time DRL agent training process, each DRL agent has its own DRL model ready for broadcast. We apply a performance optimization parameter α to determine which part of the model will be broadcasted and updated to achieve the best energy management performance in the following section.

Table 1: Reward Function

Ground truth mode	DRL action	Reward Value
On	On	10
On	Standby	-10
On	Off	-30
Standby	On	-10
Standby	Standby	10
Standby	Off	30
Off	On	-30
Off	Standby	-10
Off	Off	10

3.3.2 *Federated Personalization for Reinforcement Learning:*

We divide the neural network in DRL into two parts: base layers and personalization layers. Base layers act as the shared layers which are trained in a collaborative manner. Equation (7) shows the model aggregation step for base layers:

$$W(DRL_B)_{n,t+1} = W(DRL_B)_{n,t} - \delta \sum_{n=1}^{N} \sum_{x \in B_n,k} \frac{\nabla f(x, W(DRL_B)_{n,t})}{N} \tag{7}$$

where $W(DRL_B)_{n,t}$ denotes the base layers model weight for the n^{th} residence at time t. δ is the learning rate, B_n, k is the data sample for the $k_t h$ round of training, and each local dataset size of b. By uploading and aggregating only part of the models, PFDRL requires less computation and communication overhead, which is essential in IoT environments. On the other hand, in PFDRL, the participants share the parameters of their DRL model, which can achieve better performance in a shorter time because of the collaborative training.

While the personalization layers are trained locally, thereby enabling to capture of personal information of IoT devices. Equation 8 shows the model aggregation step for PFDRL:

$$W(PFDRL)_{n,t+1} = W(DRL_P)_{n,t} + W(DRL_B)_{n,t+1} \tag{8}$$

where $W(DRL_P)_{n,t}$ is the weight of personalization layers, and $W(PFDRL)_{n,t+1}$ is the updated model weight for PFDRL. In this way, the globally-shared base layers can be broadcasted to participating IoT devices for constituting their own EMS plan with their unique personalization layers. Thus, our proposed PFDRL is able to capture the fine-grained information on a particular device for superior personalized inference or classification and address the statistical heterogeneity to some extent. We define an 8 hidden layers architecture of neural networks inside the DRL agent and determine a performance optimization parameter α to decide the proportion of base layers and personalization layers. To avoid high-frequency broadcasting of base layers, we set γ hours as the parameters broadcast rate to reduce the frequency of broadcasting parameters. In our experiment, we determine the hyperparameter γ that has the best result.

Algorithm 2 shows the training process of personalized deep federated reinforcement (PFDRL). DRL agents will take actions based on DRL state space, which is the load forecasting value and real-time energy value. The reward will be calculated based on the action. The system calculates the Q-value and finds the maximum value based on the selection of action space. We adopt the Huber Loss function [22] which acts quadratic for small errors and linear for large errors. This prevents the network from having a dramatic change while processing outliers. After the local training step is done, we select α base layers of the model broadcasted to other residences, each residence combines the updated model with the aggregate base layers and localized personalization layers to perform energy management plans.

4 EVALUATION

In this section, we evaluate our proposed framework using various hyperparameter configurations.

Energy Consumption: To demonstrate the performance of the proposed framework, we apply it to a real-world dataset called the Pecan Street dataset [3]. The Pecan Street dataset contains the energy consumption data for various home appliances in 669 residents located in Texas from January 1, 2013, to December 31, 2017 [24].

Electricity Price: Since the electricity price can be divided into fixed-rate electricity plans and variable rate electricity plans. We obtain the electricity price from the websites of Texas Electricity Rates [2] and the websites of Energy Information Administration [1]. For fixed-rate electricity prices in TX, the average rate is 11.67 cents per kilowatt-hour (kWh). For variable rate electricity prices in TX, the range is between 0.08 cents to 20 cents per kWh depending on the time. We compare both in our experiments to see the price difference.

Experiment Settings: The experiments are deployed in our local server with a GTX 2070 super GPU. For both energy load forecasting and management, we first use 80% of the energy consumption dataset as the training set and use them to calculate the parameter and aggregate the parameters from other agents to get the updated model. Second, we use the rest 20% of the dataset for testing. For the hyperparameters in PFDRL, we set the learning rate as 0.001, discounted rate as 0.9, the memory capacity as 2000, and the target replace iteration as 100. Each hidden layer has 100 neurons followed by a ReLU function. The output layer has 3 neurons providing Q-values of the state of the three modes.

Algorithm 2: PFDRL Algorithm

```
Initialize DRL environment with load forecasting
  result $\mathcal{V}$ and real-time energy value $\mathcal{RV}$ for each
  device $D_{X_n}$;
for $n$ =0 to $N$ do
    for $t$ =0 to $T$ do
        $a_t$ = random(0,2);
        or, $a_t = \arg\max_a Q(s_t, a)$;
        TakeAction($a_t$);
        $A$.append($a_j$), ;
        $S$.append($s_j$);
        $a_t = A[t]$;
        $s_t = S[t]$;
        Compute $r_t$ from $s_t, a_t$;
        $s_{t+1} = S[t+1]$;
        for each iteration do
            Calculate Q value,
              $y_i = r(s_i, a_i) + \kappa \cdot \max_{a'} Q(s_{i+1}, a')$ ;
              Calculate error, $\tau_i = y_i - Q(s_i, a_i)$;
            Calculate loss, $\mathcal{L} = \frac{1}{B} \sum_{e_t \in B} \mathcal{L}(\tau)$;
            Update network parameters;
        end
        for $\alpha \in$ (1, 8) do
            Broadcast $\alpha$ layers in DRL to other
              residences;
            Keep $(8 - \alpha)$ layers locally;
            Receive $\alpha$ layers from all other residences;
            Do Equation 7
        end
    end
    Do Equation 8;
    $\alpha = \alpha + 1$
end
```

Compared Methods: For load forecasting, we compare four prediction algorithms: Linear regression (LR) [32], Support vector machine (SVM) [7], Back-propagation network (BP) [28], and Long short-term memory (LSTM) [26] with the same experiment settings in order to choose the method with the best performance applied to the PFDRL model.

We choose the following four methods as comparisons. (1) Local based load forecasting + Local based EMS [33] (Local). (2) Cloud based load forecasting + Local based EMS [20] (Cloud). (3) Federated learning based load forecasting + Local based EMS [27] (FL). (4) Federated learning based load forecasting + Federated learning based EMS [18] (FRL). We train the models with the same dataset under their settings. Table 2 shows the details of compared methods.

Table 2: Comparison methods.

Method	Load Forecasting	EMS	Local Area	Data Privacy	Small Batch Model Training	Sharing EMS	Personalization
Local	Local NN	Local RL	✓	✓	×	×	✓
Cloud	Cloud NN	Local RL	×	×	✓	×	×
FL	Federated Learning	Local RL	×	×	✓	×	×
FRL	Federated Learning	Federated RL	×	×	✓	✓	×
PFDRL	Decentralized Federated Learning	Personalized Federated RL	✓	✓	✓	✓	✓

4.1 Performance Metrics

1. *Hyperparameters selection.* We measure the hyperparameter α, β, and γ to determine the threshold for broadcast frequency and the number of broadcasted layers in our system.

2. *Prediction accuracy.* To measure the prediction accuracy of energy consumption, we measure the prediction accuracy as below: $Ac_n = 1 - \frac{|V_n - RV_n|}{RV_n}$ where Ac_n is the prediction accuracy of n^{th} prediction, V_n is the predicted value of n^{th} prediction and RV_n is the real value of n^{th} prediction.

3. *Saved energy value.* To measure the saved energy value, we use $RV_n - V_n$ to calculate the value.

4. *Saved monetary cost.* We calculate the total monetary cost based on real price dataset [1, 2] for fixed-rate electricity plans and variable rate electricity plans under $C_{D_{X_n},t} = (RV_{n,t} - V_{n,t}) \cdot p_t$ where $C_{D_{X_n},t}$ is the monetary cost for device D_{X_n} at time t. p_t is the energy price at time t.

5. *Time overhead.* We use training time latency and testing time latency to show the time overhead of the load forecasting methods and the energy management methods.

5 EXPERIMENTAL RESULTS

Hyperparameters Selection of PFDRL: Figure 2 shows the saved standby energy of the PFDRL framework with different shared layers α. We use $\alpha \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ to determine the best α of the proposed framework. We observe that $\alpha = 6$ has the best result, which means the best performance comes from we set 6 layers as base layers and 2 layers as personalization layers. Figure 3 shows the accuracy of our proposed DFL framework with different broadcast frequencies β. We employ $\beta \in \{0.1, 0.5, 1, 2, 6, 12, 24\}$ to adjust the best frequency of the proposed framework. We observe that $\beta = 6$ and 12 have the best prediction accuracy for load forecasting, which means the parameters should be broadcasted every 6 or 12 hours. Since higher broadcast frequency will cause lower communication efficiency, we choose $\beta = 12$ as the best frequency of our framework. Figure 4 shows the saved standby energy of our proposed PFDRL framework with different broadcast frequencies γ. We employ $\gamma \in \{0.1, 0.5, 1, 2, 6, 12, 24\}$ to adjust the best frequency of the proposed framework. We observe that $\gamma = 2$, 6, and 12 have the best performance, which means the parameters should be broadcasted every 2, 6, or 12 hours. Due to the

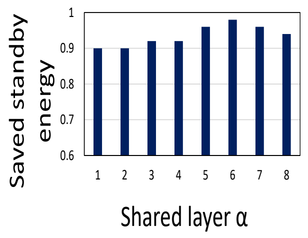

Figure 2: The saved standby energy with different shared layers α.

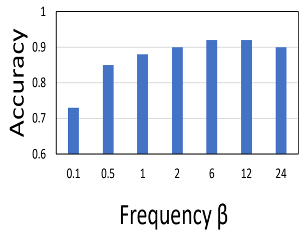

Figure 3: The accuracy of DFL with different broadcast frequency β.

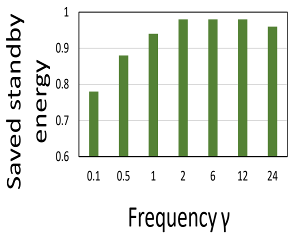

Figure 4: The saved standby energy PFDRL with different broadcast frequency γ.

same reason as described in Figure 3, we choose $\gamma = 12$ as the best frequency of our framework.

Load forecasting Accuracy: Figure 5 shows the cumulative distribution function (CDF) of the load forecasting result. The result follows LR<SVM<BP<LSTM. For LR, it's normal to face under-fitting and low precision, so the load forecasting accuracy is lower. For SVM, its performance with large datasets is lower than the others. For BP, it is easy to fall into a local extreme value, and the weights converge to a local minimum point, which causes the network training to fail. For LSTM, it can capture the long-term pattern based on the memory cell, which can bring higher load forecasting accuracy. Figure 6 shows the load forecasting accuracy in a day at different times. The result follows LR<SVM<BP<LSTM due to the same reason as explained in Figure 5. We also observe that the accuracy from 2 AM to 6 AM and from 12 PM to 16 PM are higher than the other time in the day. The reason is that, in such a time frame, residences usually have the same energy usage patterns for each device. From 8 AM to 10 AM and in the evening time, the energy usage in different residences is vary depending on the date.

Figure 7 shows the prediction accuracy while we accumulatively train the DFL framework with different numbers of days. We set the number of residences as 100 in this experiment. The result follows LR<SVM<BP<LSTM due to the same reason as explained in Figure 5. Since we accumulatively train the DFL framework, for each hour, each agent has the aggregated parameter for each device. The updated parameter will be used for the next training period, which will improve the prediction accuracy over time. On

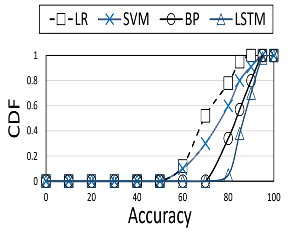

Figure 5: CDF for Load forecasting accuracy.

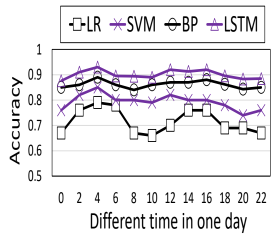

Figure 6: Load forecasting accuracy in a day.

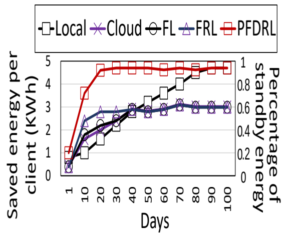

Figure 9: Saved energy per residence.

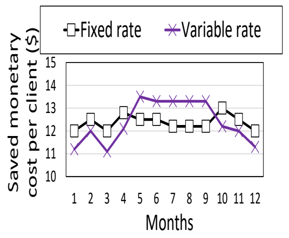

Figure 10: Saved monetary cost per residence.

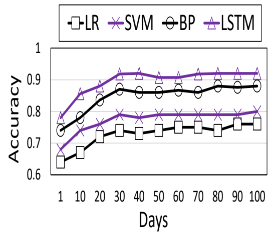

Figure 7: Prediction accuracy with different days.

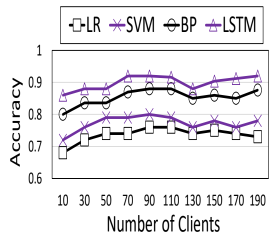

Figure 8: Prediction accuracy with different residences.

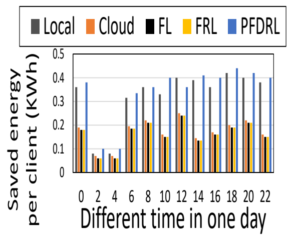

Figure 11: Saved energy per residence in a day.

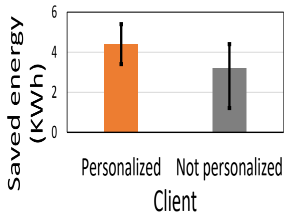

Figure 12: Performance in personalization.

the other hand, from day 1 to day 30, the growth of accuracy is higher than from day 70 to 100. The reason is that the aggregated parameter tends to approach the best value for load forecasting. Figure 8 shows the prediction accuracy with different numbers of residences participating in the DFL framework. We set the number of days as 365 in this experiment. For the number of residences under 100, the result follows LR<SVM<BP<LSTM due to the same reason as explained in Figure 5. For the number of residences above 100, the result shows a drop. The reason is that the four methods use all the parameters or data to train the model together. Such methods can indeed improve the average accuracy when the number of total participants is small. When the number of residences goes up, the number of different kinds of load patterns also goes up. In this case, using all the parameters or data to train the model may cause prediction accuracy to drop in some devices.

Performance Comparison with Compared Methods: Figure 9 shows the amount of saved energy per residence and the percentage of standby energy usage versus different training days. The result for saved energy per client follows Cloud≈FL≈FRL<Local≈PFDRL. Because the local method and PFDRL have personalization, the EMS plan is more accurate, which leads to the result that more energy can be saved from standby energy. The result for achieving the best performance time follows: PFDRL≈FRL<FL≈Cloud<Local. Because PFDRL and FRL are sharing the EMS plan with other clients in the system, which leads to the fact that sharing the EMS plan with all participants can speed up the system to achieve better performance. For FL and cloud, since they only do load forecasting in the sharing mechanism, they have a more accurate input feature for the localized EMS, but since the EMS plans are not shared, so they will need to spend more time to achieve the best performance. For local based method, since both load forecasting and EMS plan are locally based, it has the lowest speed. Figure 10 shows the saved monetary cost per residence and the percentage of the total monetary cost versus different months. We compare both the fixed-rate electricity plan and the variable rate electricity plan using our system. The result follows Fixed Rate≈Variable Rate. Since the amount of saved energy is the same, the difference in the total monetary cost in a month is between the electricity plan. On average, we can observe that Fixed Rate≈Variable Rate. From April to June, the variable rate plan will save more money for each resident. From August to October, the fixed-rate plan will save more money for each resident.

Figure 11 shows the amount of saved energy per residence at different times of the day. We can observe that the result follows Cloud≈FL≈FRL<Local≈PFDRL due to the same reason as explained in Figure 9. We can also observe that, between 2 AM and 4 AM, the saved energy is at a minimum. The reason is that the total usage of energy is at the lowest level. On the other hand, between 12 PM and 0 AM, the saved energy is at maximum since residents are using more energy at such time in the day. Figure 12 shows the system performance in personalization. We show the mean accuracy of the personalized model and not the personalized model. We can observe that the personalized model has better performance than the not personalized model. Also, from the error bar, we can observe that personalized model can achieve better performance for most residences.

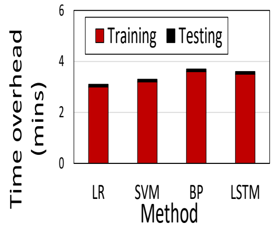

Figure 13: Load forecasting time overhead.

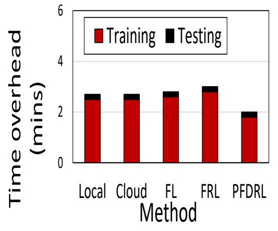

Figure 14: Energy management time overhead.

Figure 13 and 14 show the time overhead of each method. The time overhead contains both training and testing for load forecasting and energy management. For load forecasting, the result follows LR≈SVM≈BP≈LSTM. For energy management, the result follows PFDRL<FL≈Cloud≈Local<FRL. The reason is that, for local based model, since both load forecasting and DRL are local, the training time overhead is small. For cloud and FL, since they need to broadcast the load forecasting parameters but the DRL is local based, so they require more training time than local based method. For FRL, both load forecasting and DRL are based on the FL framework, which means they need to broadcast two times, which will need the longest training time. As for PFDRL, because of the layer selection, the number of broadcasted parameters is smaller, which leads to a shorter training time.

6 CONCLUSION

In this paper, we propose a privacy-preserved energy management system that can achieve the best performance in a short time and minimize the energy usage caused by standby energy. First, we introduce decentralized federated learning (DFL) framework to enable distributed edge devices to collaboratively train a model without using cloud service. Second, to further improve the time to achieve the best EMS performance, we applied deep reinforcement learning (DRL) with FL in order to share the EMS plan. Third, we design a personalized federated DRL (PFDRL) to maximize the EMS performance for each individual client in the sharing system by dividing the network inside the DRL into localized-based layers and personalized layers. We evaluate the proposed PFDRL energy management system on the real-world Pecan Street dataset, which saves 98% of total standby energy consumption per day in a residential building.

ACKNOWLEDGMENTS

This work was supported by NSF CCF #2217071, NobleReach, and Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development.

REFERENCES

[1] 2020. Energy Information Administration. *https://www.eia.gov/* (2020).
[2] 2020. Texas Electricity Rates. *https://comparepower.com* (2020).
[3] 2021. Dataport Database. https://dataport.pecanstreet.org.
[4] 2021. Help Net Security. *https://www.helpnetsecurity.com* (2021).
[5] U. Aivodji. 2019. IOTFLA: A secured and privacy-preserving smart home architecture implementing federated learning. In *SPW*.
[6] M. Al Faruque and K. Vata. 2015. Energy management-as-a-service over fog computing platform. *IEEE internet of things journal* (2015).
[7] Lijuan Cao. 2003. Support vector machines experts for time series forecasting. *Neurocomputing* 51 (2003), 321–339.
[8] G. Din, A. Mauthe, and A. Marnerides. 2018. Appliance-level short-term load forecasting using deep neural networks. In *ICCNC*.
[9] Ali Fallah and Aryan Mokhtari. 2020. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948* (2020).
[10] Jiechao Gao, Mingyue Tang, Tianhao Wang, and Bradford Campbell. 2022. PFed-LDP: A Personalized Federated Local Differential Privacy Framework for IoT Sensing Data. In *Sensys*.
[11] Jiechao Gao, Wenpeng Wang, Zetian Liu, Md Fazlay Rabbi Masum Billah, and Bradford Campbell. 2021. Decentralized federated learning framework for the neighborhood: a case study on residential building load forecasting. In *Sensys*.
[12] Jonas Geiping and Hartmut Bauermeister. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *NeurIPS* (2020).
[13] K Gram-Hanssen. 2010. Standby consumption in households analyzed with a practice theory approach. *Journal of Industrial Ecology* (2010).
[14] J. Haj-Yahya. 2020. Techniques for Reducing the Connected-Standby Energy Consumption of Mobile Devices. In *2020 IEEE HPCA*.
[15] Yaochen Hu and Di Niu. 2019. FDML: A collaborative machine learning framework for distributed features. In *Proceedings of the 25th SIGKDD*.
[16] W. Kong. 2017. Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE TSG* (2017).
[17] Lawrence Berkeley Laboratory. 2008. https://standby.lbl.gov/docs/.
[18] S. Lee. 2020. Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources. *IEEE TII*.
[19] M. Lu and J. Lai. 2020. Review on carbon emissions of commercial buildings. *Renewable and Sustainable Energy Reviews* (2020).
[20] R Lu. 2019. Demand response for home energy management using reinforcement learning and artificial neural network. *TSG*.
[21] X. Luo. 2019. Development of an IoT-based big data platform for day-ahead prediction of building heating and cooling demands. *AEI*.
[22] Gregory P Meyer. 2021. An alternative probabilistic interpretation of the Huber loss. In *Proceedings of the IEEE/CVF CVPR*.
[23] Yusuf Ozturk and Prakash Jha. 2013. A personalized home energy management system for residential demand response. In *4th ICPEEED*.
[24] P Raj, M Sudhakaran, P Philomen-D-Anand Raj, et al. 2009. Estimation of standby power consumption for typical appliances. *JESTR* (2009).
[25] M. Soliman and T. Abiodun. 2013. Smart home: Integrating internet of things with web services and cloud computing. In *IEEE ICCCTS*.
[26] Idil Sülo and Theodore Brown. 2019. Energy efficient smart buildings: LSTM neural networks for time series prediction. In *Deep-ML*.
[27] Afaf Taïk and Soumaya Cherkaoui. 2020. Electrical load forecasting using edge computing and federated learning. In *ICC*. IEEE.
[28] Lin Wang. 2015. Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *ESWA*.
[29] S. Wang, T. Tuor, and T. Salonidis. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE JSAC* (2019).
[30] W Wang, J Su, Za Hicks, and Bradford Campbell. 2020. The Standby Energy of Smart Devices: Problems, Progress, & Potential. In *IoTDI*.
[31] Q Wu and X Chen. 2020. FedHome: Cloud-Edge based Personalized Federated Learning for In-Home Health Monitoring. *TMC* (2020).
[32] W Xu. 2019. A hybrid modeling method for time series forecasting based on a linear regression model and deep learning. *AI* (2019).
[33] Xu Xu and Youwei Jia. 2020. A multi-agent reinforcement learning-based data-driven method for home energy management. *TSG* (2020).
[34] L. Yang, X. Chen, J. Zhang, and H. Poor. 2014. Optimal privacy-preserving energy management for smart meters. In *IEEE INFOCOM*.