FedNILM: Applying Federated Learning to NILM Applications at the Edge

Yu Zhang, Guoming Tang, Member, IEEE, Qianyi Huang, Yi Wang, Member, IEEE, Kui Wu, Senior Member, IEEE, Keping Yu, Member, IEEE, and Xun Shao, Member, IEEE

Abstract—Non-intrusive load monitoring (NILM) helps disaggregate a household’s main electricity consumption to energy usages of individual appliances, greatly cutting down the cost of fine-grained load monitoring towards the green home vision. To address the privacy concern in NILM applications, federated learning (FL) could be leveraged for NILM model training and sharing. When applying the FL paradigm in real-world NILM applications, however, we are faced with the challenges of edge resource restriction, edge model personalization, and edge training data scarcity. We present FedNILM, a practical FL paradigm for NILM applications at the edge client. Specifically, FedNILM delivers privacy-preserving and personalized NILM services to large-scale edge clients, by leveraging i) collaborative data aggregation through federated learning, ii) efficient cloud model compression via filter pruning and multi-task learning, and iii) personalized edge model building with unsupervised transfer learning. Our experiments on real-world energy data show that FedNILM can achieve personalized energy disaggregation with the state-of-the-art accuracy, while preserving the user privacy.

Index Terms—Non-intrusive load monitoring, green home, federated learning, model compression, transfer learning.

I. INTRODUCTION

NON-INTRUSIVE load monitoring, also known as energy disaggregation, was first introduced by Gorge W. Hart in 1980s [1]. By nature it is a single-channel blind source separation (BSS) problem that aims to decompose the aggregated power readings of a household into appliance-wise power consumption. One of the major purposes of NILM is to reduce energy consumption efficiently and pursue the green home vision. Evidences have shown that the itemized information could encourage householders to use energy in a more sustainable way, achieving about 15% energy saving [2], [3]. Besides, NILM can be leveraged to evaluate conservation programs, improve the quality of load forecasting, and provide references for power grid management [4]. For example, with real-time NILM, utility companies could suggest switching operations on particular appliances (e.g., air conditioners) for load shifting in peak power hours [3].

Although proposed for decades, NILM has not been deployed broadly, mainly due to the difficulty of defining appliance signatures with manually labeled features [5], [6], [7]. Most recently, it shows that the deep neural network (DNN) based approaches could greatly improve the performance of NILM [8], [9], [10], [11], as neural networks are able to automatically learn appliance signatures. Various neural network architectures have been proposed for NILM, including denoising auto-encoder [8], recurrent neural networks [12], and GAN [13], etc. Among those DNN-based methods, the Seq2Point model [9], a one-dimensional CNN-based auto-encoder architecture, is the state-of-the-art model for energy disaggregation.

Arguably, the DNN-based NILM models largely rely on sufficient and diverse training data, whereas realistic datasets often exist in the form of isolated islands. Although there are plenty of meter data in different buildings, it is almost impossible to transmit or integrate these local user data into a centralized storage, due to limits in communication bandwidth and legislation in user privacy and data security [14]. Actually, in the last few years, the emphasis on data security and user privacy has become a global issue. For example, in the United States, China, and the European Union, relevant regulations have been enforced to protect data security and privacy [15], [16], making it legitimately risky to gather massive user energy consumption data. In consequence, it is impractical to train powerful NILM models with existing paradigms.
A. Motivations and Challenges

To address such problems in a collaborative way, federated learning (FL) has recently emerged as a promising paradigm. In a canonical FL system, user data is kept on client devices and the NILM model training is realized by i) local model updates with users’ own data and ii) cloud model fusion with all users’ models. In this way, client data remains locally, and separate clients collaboratively build a more accurate global model [17]. There has been a lot of interest towards employing FL framework in Internet of Things (IoT) [18], [19], [20]. It seems natural to apply the FL paradigm to the NILM problem for privacy preserving, which was theoretically verified in [21] recently. When dealing with real-world NILM applications at local houses, however, we are still faced with the following major challenges.

Challenge 1: One critical limitation of adopting the typical FL system to NILM applications is the constrained resource in local homes (or edge clients), in terms of computation power, communication bandwidth, memory and storage size, etc. In reality, usually low-end devices (or edge devices) rather than powerful servers are used in local homes. Hence, as NILM models usually take large memory and computation overheads, network training on such resource-constrained edge devices could take prohibitively long time or even be terminated immediately [22]. Since we cannot perform the model fusion at cloud without local updates, the resource constraint at the edge client literally hinders the FL paradigm from applying in NILM applications.

Challenge 2: The second one relates to the personalization of NILM models. Although recent DNN-based approaches are promising for NILM, it is not clear whether existing models are transferable among the diverse edge clients, since most of the models are trained on public datasets (e.g., REDD [23] and UKDALE [24]). In other words, even with satisfactory performance on the common training datasets (or source domain), these models may perform badly in a testing house (or target domain), owing to the distribution difference between training and testing data [25]. Generally speaking, different households usually have different appliances as well as energy usage patterns, making those models hardly capture edge clients’ heterogeneity and resulting in poor scalability in practice [25].

Challenge 3: The third one is on training data scarcity at edge clients. For NILM applications, obtaining unlabelled testing data (i.e., mains power data) is quite easy, as current utility smart meters report the whole-home energy consumption periodically. To acquire labelled training data (i.e., power consumption data of individual appliances), however, is extremely expensive if not impossible. Although there are various power sensing devices and we could equip each appliance with a power sensor, this would incur enormous installation and maintenance costs, and thus is unscalable across large-scale households. As current DNN-based NILM models are generally trained from large amounts of labelled data [25], the scarcity of training data further limits the scalability of NILM systems and applications.

B. Our Ideas and Contributions

To tackle the resource limitation at the edge client (Challenge 1), our idea is to compress the complex NILM model to a simpler one that the edge device can afford. More specifically, the cloud NILM model under the FL paradigm could be pruned before being circulated to edge clients, and thus the edge devices would carry out local computation based on the compressed model. Besides, a compressed NILM model is also desirable for cloud-edge communications. Then, to build personalized NILM models for diverse edge clients (Challenge 2), particularly under the condition of training data scarcity (Challenge 3), we propose to leverage the unsupervised transfer learning. By aligning the feature map distributions between the source and target domains, the unsupervised transfer learning techniques (e.g., CORAL in Section VI-B) could be incorporated with existing NILM modelling process, and properly address the domain shift issue for NILM model personalization.

In this work we present FedNILM, a practical FL paradigm for NILM applications at the edge. FedNILM aims to provide privacy-preserving and personalized energy disaggregation for large-scale edge clients, aided by the cutting-edge DNN model compression and transfer learning techniques at resource constrained edge devices. This paper shows that, the adapted FL framework can be highly preferable for implementing and deploying large-scale NILM applications. Our major contributions can be summarized as follows.

- We propose FedNILM, a practical FL paradigm designed for NILM applications among large-scale edge clients. Aided by the FL paradigm and with practical considerations, FedNILM can provide scalable NILM services with the state-of-the-art accuracy while retaining data privacy for edge clients.
- We adopt cloud model compression for edge adoption in FedNILM. By compressing deep learning models, we are able to effectively cut down the computation cost by about 60% and reduce memory cost by up to 84% at the edge while retaining satisfactory performance.
- We incorporate unsupervised transfer learning with FedNILM for client model personalization. By introducing the CORAL loss into the state-of-the-art NILM model, we manage to realize local transfer learning without relying on labelled training data at the edge client.
- We make extensive evaluations to validate the performance of FedNILM. The results demonstrate that, aided by model compression at cloud and model personalization at edge, FedNILM can provide a comparable performance to the state-of-the-art without compromising the user privacy.

The rest of the paper is organized as follows. In Section II, we give the background knowledge and related literature. Section III briefs the state-of-the-art NILM model and the federated learning rationale. Then, we present the design of FedNILM in Section IV, including the paradigm overview and workflow. Two key operations of FedNILM, cloud model compression and edge model personalization, are presented in Section V and Section VI, respectively. The experimental evaluations are performed in Section VII. The paper is concluded in Section VIII.
II. BACKGROUND AND RELATED WORK

Recently, there are growing interests in deploying NILM applications and systems by energy service providers, energy aggregators and distribution system operators [13]. With the urgent request of privacy preserving, the FL paradigm has been exploited [21], where the NILM model/service is deployed/delivered across the edge clients for energy disaggregation. In real-world implementations, however, some challenges could arise.

A. NILM Model Compression

The first challenge is how to perform DNN-based NILM model inference at edge clients where only resource constrained devices could be installed. Deploying compressed NILM models on edge devices might be a promising approach. It has been demonstrated that model compression may well maintain the NILM performance while significantly reducing the computational overhead [22], [26].

There are various algorithms for compressing or pruning neural networks. In [22], the authors leverage filter pruning [27] and tensor decomposition [28] methods to compress the convolutional layers, where the former refers to sparsify the neural network by removing less important parameters and the latter is to perform a low-rank decomposition of the learnt filter matrix. Surprisingly, experimental results in [22], [29] show that compressing neural networks might bring some performance gain due to better generalization in test cases.

Although the compressed NILM models could be deployed at the edge client for inference, the training of original NILM models still needs to be conducted on the powerful GPU/CPU servers at the cloud end [30]. Thus, one client needs to upload their energy data to the cloud for NILM model training, and the data privacy concern still exists.

B. NILM Model Transfer

The other challenge is on the degradation of the cloud model at the edge client. When applying the FL paradigm for NILM, the same model trained at the cloud would be delivered to various client ends. When the distributions between the client data and the cloud data are different, we may observe performance degradation from the NILM model at the client.

To this end, the transferability issue of NILM models has been preliminarily studied. In [25], the authors investigate two transfer learning schemes, i.e., the appliance transfer learning (ATL) and cross domain transfer learning (CTL). Both ATL and CTL freeze the convolutional layers and merely tune the fully-connected layers during retraining. Recently, apart from the canonical CNN structure, the authors of [13] develop TrGAN-NILM, which is based on the generative adversarial networks (GANs), to automatically extract common feature representations between source and target domains through minimizing the statistical distance between different domains.

Nevertheless, the aforementioned transfer learning techniques need labelled training data on the target domain. In other words, they cannot be directly applied in the situation where the target domain is unlabeled, which is quite common in real-world NILM applications. In consequence, for buildings that are newly added to the NILM service provider portfolio, obtaining labelled appliance meter data and performing transfer learning are prohibitive and time consuming.

In this work, we design FedNILM, a practical FL paradigm for real-world NILM applications. Particularly, FedNILM tackles the computation limitation issue by incorporating model compression techniques, and addresses the model personalization problem with unlabelled target domain.

III. PRELIMINARY

A. NILM Problem Definition

The goal of NILM is to recover the energy consumption of individual appliances from the mains meter signals. Given the aggregate power readings from T time periods, we can denote them by \( y = (y_1, y_2, \ldots, y_T) \), where \( y_t \in R_+ \). Then, let \( x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_w^{(i)}) \) in which \( x_t^{(i)} \in R_+ \) denotes the power reading of the i-th appliance at time t. Therefore, at each time instant t, \( y_t \) is assumed to be the sum of all N appliances' power readings. Normally, we are only interested in the top \( N' \) appliances used widely and consuming the most energy. Then, the power consumption of the remaining appliances can be represented as \( u = (u_1, u_2, \ldots, u_T) \) and the aggregate power consumption could be represented as follows:

\[
y_t = \sum_{i=1}^{N'} x_t^{(i)} + u_t + \epsilon_t
\]

where \( \epsilon_t \) denotes a Gaussian noise.

B. State-of-the-Art NILM Model

We introduce the state-of-the-art model, i.e., sequence to point (Seq2Point) [9], recently developed for solving the NILM problem. The Seq2Point learning model maps a window of the mains signal readings to the midpoint point of the corresponding window of the target individual appliances. For each time instant t, given a fixed time window with size of w, the Seq2Point model uses the mains power signal sequence \( y_{t:t+w-1} = (y_t, y_{t+1}, \ldots, y_{t+w-1}) \) as the input and the middle element \( x_{t+w/2}^{(i)} \) in power readings of the target appliance i as the output. In other words, instead of estimating the whole power signal sequence of the target appliance, the Seq2Point model merely predicts the middle signal element of the appliance in the corresponding time window.

Mathematically, for a target appliance i, assume that there exists a function \( f^{(i)}: R_+^w \rightarrow R_+ \), and the function gives the power estimation of appliance i at time t + w/2 by:

\[
f^{(i)}(y_{t:t+w-1}) = x_{t+w/2}^{(i)}
\]

Thus, the key task in Seq2Point is to learn the specific form of function \( f^{(i)} \). Once \( f^{(i)} \) is obtained, we are able to estimate the power signal of the target appliance i with the aggregate (mains) signals, thus achieving energy disaggregation.

More specifically, to learn the parameters of \( f^{(i)} \), Seq2Point employs the convolutional neural network (CNN) as the training structure, as illustrated in Fig. 1(a). It was demonstrated
that such a DNN structure could inherently learn the signatures of target appliances and shows superior performance than other models [8].

C. Federated Learning

At the beginning of FL, the server trains a cloud model based on the original dataset. When we adopt deep neural networks to learn both the cloud and client models, the learning objective of the global model could be formulated as:

$$\arg\min_{\omega} f_\omega(\omega) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i; \omega)$$

where $L(x_i, y_i; \omega)$ denotes the loss of prediction on server samples $\{x_i, y_i\}_{i=1}^{n}$ made with model parameters $\omega$, i.e., the weights and bias.

The server then sends the current global state $\omega$ to each of the clients, thus enabling each client to perform local computation based on the global state and its local dataset. Technically, the learning objective of client $u$ can be denoted as:

$$\arg\min_{\omega_u} f_u(\omega_u) = \frac{1}{n^u} \sum_{i=1}^{n^u} L(x^u_i, y^u_i; \omega_u)$$

where $\{x^u_i, y^u_i\}_{i=1}^{n^u}$ are local samples with $n^u$ denoting their sizes. After the client model $f_u$ is trained, the local computation results, namely local model parameters $\omega_u$, would be uploaded to the server. For parameter transmission between cloud and clients, homomorphic encryption is usually adopted to avoid information leakage [31].

Then, with enough local updates, the cloud server performs federated aggregation [17] to align user models and obtain a new global state. Assuming that there are $K$ clients which are indexed by $k$ and each client locally updates its gradient to $\omega_{t+1}^k$ using its local data. The server then takes a weighted average of all these local models, which can be formulated as:

$$\omega_{t+1} = \frac{1}{K} \sum_{k=1}^{K} \omega_{t+1}^k$$

where $\omega_{t+1}$ denotes the updated global parameters. After adequate rounds of iterations and the global model has satisfactory generalization ability, the global model is distributed to the client for local deployment.

IV. FedNILM Paradigm

Modern DNN-based NILM models cannot succeed without access to large amount of training data at the client side. However, this causes severe concerns on user privacy and data security. To tackle these issues, we leverage the federated learning approach to the NILM problem and design FedNILM in this section.

A. Overview

Without loss of generality, we consider one cloud server and multiple edge clients, as illustrated in Fig. 2.

- At the cloud side, the cloud server trains a Seq2Point model with readily-available open datasets, prunes it into a slim version, and shares it among the associated edge clients.
- At the client side, each client further tailors the pruned model through a local transforming process, and adopts the personalized model for individual appliance monitoring.

B. Workflow

As shown in Fig. 2, the workflow of FedNILM mainly consists of four steps.

- **Step-1:** Based on the state-of-the-art NILM model (introduced in Section III-B) and public datasets, the cloud server develops a compressed model with model pruning techniques. Refer to Section V for the details of model pruning. Then, the compressed NILM model is distributed to all the associated edge clients.

- **Step-2:** Based on the shared model from the cloud, each edge client further trains a personalized NILM model with their own data at hand. An unsupervised transfer learning method is applied for the local model personalization. Refer to Section VI for the detailed transfer learning process.
• **Step-3**: The parameters of personalized models at the edge are encrypted and uploaded to the cloud server, where the original cloud model is updated through the federated aggregation process (introduced in Section III-C).

• **Step-4**: The updated cloud model could be distributed to either new edge clients for personalized model building, or existing edge clients for continuous model refining with fresh local data.

Generally speaking, the complete FedNILM workflow includes the starting process (i.e., Step-1 and Step-2 in sequential) and a repeating process (i.e., Step-3 and Step-4 in iterative). Next we introduce the two key operations in the workflow, i.e., cloud model compression in Step-1 and client model personalization in Step-2, respectively.

V. CLOUD MODEL COMPRESSION

As we have mentioned in Section III-B, the Seq2Point model trains a separate model for each individual appliance. That is to say, to monitor $n$ appliances for an edge client, we have to train and deploy $n$ appliance-specific models on the corresponding edge device. This demands tremendous resources that may not be satisfied by the resource constrained edge devices. In this section, we show how model compression techniques could be leveraged in FedNILM.

A. MTL-Seq2Point

Multi-task learning (MTL) refers to share representation, usually layers in neural network, between analogous tasks, and empowers the neural network to have better generalization ability [32]. Intuitively, leveraging excess information that comes from auxiliary tasks enables the MTL model to perform well in the main task [33]. By leveraging MTL techniques in the Seq2Point modeling (we name the variant MTL-Seq2Point), we are able to train one single model for all appliances for an edge client, however, we are faced with the domain shift of the filters are measured through calculating the sum of the relative importance of the filters. More specifically, the relative importance of the filters is measured through calculating the sum of their importance. We name the model pruned MTL-Seq2Point.

Specifically, in the network structure of MTL-Seq2Point, let $n_i$ denote the number of input channels for the $i$-th convolutional layer and $w_i$ the window size of input. Given the kernel size $k$, we have 1D kernel $K \in \mathbb{R}^{k \times 1}$ (e.g., $10 \times 1$ in the first layer). Supposing that the number of output channels in the $(i+1)$-th convolutional layer is $n_{i+1}$, the 3D filters matrix $F_i \in \mathbb{R}^{n_{i+1} \times n_i \times k}$ could transform the input $X_i \in \mathbb{R}^{n_i \times w_i}$ into the output $X_{i+1} \in \mathbb{R}^{n_{i+1} \times w_i \times 1}$. Then, given the stride size $s$, we have $w_{i+1} = w_i - k + s$. Thus, the number of operations in the $i$-th convolutional layer is $n_i(n_i+1)k(w_i+1)$. By removing one of the $n_{i+1}$ filters in $F_i$, we could eliminate $n_i(kw_i+1)$ operations, as illustrated by Fig. 3. Moreover, pruning a filter also results in the removal of corresponding feature maps and kernels of the following layer, thus cutting down another $n_{i+1}kw_{i+1}$ operations. Hence, we can conclude that, by pruning $m$ filters of layer $i$, we could reduce the computation cost by $m/n_{i+1}$ at both the $i$-th and $(i+1)$-th layers.

Since not all trained filters are equally important, in order to minimize the performance drop, we choose and prune the less instrumental filters. More specifically, the relative importance of the filters are measured through calculating the sum of $L_1$-norm [27] or $L_2$-norm [34], [35] on convolutional filters. As there are no noticeable differences between these two criteria in filter selection [27], we leverage the $L_1$-norm to score the filters and prune $k\%$ of them in each layer with the least $L_1$-norm values. By increasing the pruning percentage of the network iteratively, we can also investigate the variation of model performance and thus find the optimal pruning percentage for our model.

Overall, the procedure to obtain the best pruned MTL-Seq2Point model consists of the following four steps: i) train a convergent MTL-Seq2Point model, ii) score corresponding convolutional filters based on the sum of $L_1$-norm values, iii) prune the least important filters according to their scores, and iv) retrain the pruned MTL-Seq2Point model to compensate for incurred performance degradation.

VI. CLIENT MODEL PERSONALIZATION

When adopting the compressed cloud NILM model at the edge client, however, we are faced with the domain shift problem, i.e., the difference of distributions between the cloud and client data. Thus, the common model can perform very
well on the cloud dataset but poorly on the client ends. To address the problem, we leverage the transfer learning technique and build personalized models for different clients. Particularly, we adopt the unsupervised transfer learning to work with unlabelled client data, hence addressing the training data scarcity problem at the edge client.

A. Transferability Analysis

Transfer learning works under the scenario where observations from the source domain (denoted by $D_s$) have a different distribution with those from the target domain (denoted by $D_t$). In our case, the source domain refers to the public dataset at cloud, while the target domain is the personal dataset at each client. There have been interests towards identifying the transferability of features in each layer of the DNN model, especially in Computer Vision [36].

For our NILM model, to investigate the transferability of different model layers, we propose the following three-step procedure.

1) Train an initial NILM model (i.e., the pruned MTL-Seq2Point model) on $D_s$.
2) Fix and fine-tune one of the multiple layers in the pretrained model, and randomly initialize the parameters in the rest of layers.
3) Retrain the model on $D_t$ and compare the prediction performance from the multiple transferred models.

Note that the NILM model in our scenario is composed of five convolutional layers and two fully-connected layers, as illustrated in Fig. 1. Thus, through procedurally freezing or tweaking these layers, we have $(2 \times 7)$ different transfer learning models, which are then retrained and tested on target domain data. Wrapping up all the results, we have the following observations (refer to Section V-C for more details).

- The convolutional layers of the model are good at extracting low-level and generic features. The transferable load features, such as the ON/OFF switching points, power level of appliances and typical usage durations, are insusceptible to the difference between source and target domains.
- The fully-connected layers take responsibility for learning high-level features for specific appliances. Thus, when applied to a new edge client (target domain), they need to be further fine-tuned. The client model personalization can then be realized by referring to the transferred features from convolutional layers and fine-tuning the fully-connected layers for specific appliances.

These observations are coincide with those from [25] that, the features at lower layers are highly transferable as lower layers tend to learn common and coarse information, whereas the features at higher layers are mainly for specific tasks and thus are more personalized. Accordingly, in our case of transfer learning at the client side, we propose to freeze the convolutional layers in model transforming (i.e., keep their parameters fixed in back propagation) and merely update the weights on fully-connected layers.

B. Correlation Alignment

Correlation alignment (CORAL) is an instrumental domain adaptation method, which tackles the domain shift via aligning the feature distributions of source and target samples [37]. More specifically, deep CORAL aligns the source and target data distributions through learning a nonlinear transformation, i.e., a differentiable loss function (CORAL loss), between source and target layer activations [38]. This nonlinear transformation is designed to minimize the distance between the second-order statistics (covariance) of the source and domain layer features.

Given the source domain training data $D_s$ (with labels $L_s$) and target input data $D_t = u_t$, we aim to align the distribution in layer $l$ which generates the $\sigma$-dimensional deep layer features $x$ of input $D_s$ and $u$ of input $D_t$. With $x$ and $u$, we could compute the covariance matrices $C_s$ and $C_t$. Therefore, the CORAL loss is defined as the distance between the second-order statistics (covariance) of the source and domain layer features:

$$L_{CORAL} = \frac{1}{2\sigma^2} \| C_s - C_t \|_F^2$$  \hspace{1cm} (6)

where $\| \cdot \|_F^2$ denotes the squared matrix Frobenius norm. Let $\lambda$ denote the trade-off parameter and $L_R$ the regression loss of the edge model. The loss in the client model training can be designed as:

$$L_{EDGE} = L_R + \lambda \cdot L_{CORAL}$$  \hspace{1cm} (7)

The above loss serves as a constraint and regulates the distance between source and target domains during the fine-tuning process. By jointly optimizing the regression loss and CORAL loss, we can obtain a personalized client model with both generic signatures pre-trained on the source domain and specific features working well on the target domain.

Fig. 4 shows the detailed model structure tailored for the unsupervised transfer learning process. In this way, the local devices are able to reap the shared benefits of common features pre-trained on a large generic dataset while retaining the speciality fine-tuned on client data.
TABLE I
MAE RESULTS FROM CLOUD MODEL COMPRESSION ON REDD DATASET (SEQUENCE LENGTH 499)

| Model             | Washing machine | Fridge | Dish washer | Microwave | Time (s) | Size (MB) |
|-------------------|-----------------|--------|-------------|-----------|----------|-----------|
| Seq2Point         | 20.25           | 25.61  | 14.09       | 13.30     | 9.36     | 367.82    |
| 30% pruning       | 18.03           | 26.16  | 12.82       | 9.95      | 4.78     | 180.05    |
| 60% pruning       | 18.30           | 25.75  | 16.91       | 10.65     | 3.06     | 58.80     |
| 90% pruning       | 20.53           | 44.57  | 48.36       | 13.76     | 2.33     | 3.69      |
| MTL-Seq2Point     | 18.74           | 27.04  | 19.40       | 12.37     | 2.69     | 91.97     |
| 30% pruning       | 18.55           | 27.93  | 21.94       | 12.74     | 1.26     | 45.02     |
| 60% pruning       | 19.18           | 28.39  | 19.26       | 13.15     | 0.83     | 14.7      |
| 90% pruning       | 19.39           | 75.21  | 28.86       | 20.60     | 0.68     | 0.93      |

VII. EXPERIMENTS

In this section, we use real-world energy datasets to evaluate the proposed FedNILM paradigm, including cloud model compression and client model personalization, respectively.

A. Experimental Settings

1) Cloud and Edge Systems: In our experiments, an AMAX server with four Tesla V100 GPUs serves as the cloud end. A low-end desktop (with Nvidia GTX 960M) and an edge device (NVIDIA Jetson Nano) serve as two different client ends.

2) Datasets: Three benchmark datasets are used to evaluate the performance of FedNILM, including REFIT [39], UK-DALE [24] and REDD [23]. All contain similar appliance categories, allowing the evaluation of model transferability. Also, they have been widely applied in previous NILM research [8], [9], [25], and thus enabling performance comparison with the state-of-the-art solutions.

3) Performance Metrics: Mean absolute error (MAE), signal aggregate error (SAE) and F1-score are used to evaluate the performance of FedNILM, all of which have been leveraged in prior NILM research [8], [9], [10], [22]. In particular, the former two metrics, i.e., MAE and SAE, measure the performance of power consumption estimation, while the F1-score reflects the performance of appliance ON/OFF states estimation.

B. Evaluations on Cloud Model Compression

At the cloud end, we train the Seq2Point model with/without pruning and MTL on the REDD dataset, and compare the NILM performance, run-time and memory overhead from different model adoptions. To show the impacts of model compression, we prune the parameters of Seq2Point and MTL-Seq2Point by 30%, 60% and 90%, respectively. The results are summarized in Table I and Table II, for input sequence lengths of 499 and 99, respectively.

Based on the experimental results in Table I and Table II, we observe that both multi-task learning and filter pruning could significantly reduce the model size along with decent inference time benefits, without drastically compromising NILM performance. In particular, for the sequence 499 (see Fig. 5(a) and Fig. 5(b)), pruning 60% of filters on classical Seq2Point model can save approximately two-thirds of running time and reduce memory cost by up to 84% while retaining comparable prediction accuracy. Meanwhile, the multi-task learning structure virtually takes one fourth of the original computation and memory overhead, and it simultaneously generates the disaggregation results of four selected appliances. The combination of filter pruning and multi-task learning techniques could help save nearly 92% of inference time and 96% of model space with slight (<10%) performance degradation.

For sequence length 99 (see Fig. 5(c) and Fig. 5(d)), the unoptimised model occupies 6× more space and requires more inference time than the optimised model pruned 60% of filters with similar performance. Similar with what we have found with the model of sequence length 499, the multi-task learning structure helps save nearly three fourths of running time and space for sequence length 99; meanwhile filter pruning may not have much influence on inference time, as the time required for these three pruned models with different pruning percentages are almost the same. Note that we merely prune the filters in convolutional layers and in a forward pass. The majority of computations takes place in fully-connected layers, not convolutional layers, so the reduction of parameters in convolutional layers may not reduce much operations in inference procedure, thus explaining this phenomenon.

In conclusion, the general NILM performance decreases with the increasing of pruning percentage, and the employment of multi-task learning structure also contributes to the performance degradation as all the four appliances use the same set of signatures for prediction. However, we also observe that in some circumstances, model pruning might lead to better NILM results, such as the pruned Seq2Point model performance versus the original Seq2Point model accuracy on distinguishing the washing machine. This can be explained by...
“regularisation effect” as removing the least important weights from a neural network might lead to better model generalisation. In general, we choose to build our general cloud model based on Seq2Point model with multi-task learning and prune 60% of the filters in its convolutional layers.

C. Evaluations on Client Model Personalization

At the client end, we investigate the transferability of each layer of the NILM model, determine which layer’s parameters to freeze or fine-tune during model transfer, and validate the necessity to perform local transfer learning.

1) Layer Transferability of NILM Model: Based on the procedure given in Section VI-A, we train a Seq2Point model to disaggregate the mains signal for a dish washer on REFIT dataset, and then transfer this model to detect the dish washer in REDD dataset. More specifically, we gradually transfer the seven layers in Seq2Point model, by either fixing or fine-tuning them and initializing the rest layers. The experimental results are shown in Fig. 6, based on which we have the following observations and empirical findings.

- First, we observe the significantly increasing of both MAE and SAE along with the decreasing of F1-score, compared with the nearly steady performance from conv1 to conv2, once we begin to share the fully-connected layers. This implies that the convolutional layers virtually learn the generic features that are common in source and target domains, while the fully-connected layers focus on extracting domain specific signatures. In other words, the number of transferred convolutional layers generally have little, if any, influence on the final prediction, while the more the dense layers we transfer the less the accuracy we get on the target domain.
F1-score, transfer learning model shows better performances for SAE, the accuracy is improved for three appliances; for is improved for all four appliances with transfer learning; F1-score, respectively. In particular, for MAE, the performance one by up to 45.84% in MAE, 86.63% in SAE and 40% in the model with local transfer learning outperforms the original observe that: for both models with sequence length 99 and 499, their performance and see whether transfer learning helps. With results from the above two tests, we can compare leverage the retrained model to detect appliances in UK- disaggregation. For the test with transfer learning, we first employ the pre-trained cloud model on UK-DALE for energy in the latter dataset are in the U.K. households in the former dataset are in the U.S. while those dataset. Arguably, REDD is distinct from UK-DALE, as the pre-trained cloud model to detect appliances in UK-DALE data of house 1 in REDD. The local clients are comprised of data of house 2 in REFIT, data of house 1 in UK-DALE and leave it as our future work.

2) Effectiveness of Transfer Learning: To validate the effectiveness of local transfer learning, particularly the CORAL approach, we conduct experiments on the edge device of Nvidia Jetson Nano to simulate the procedure of local updates. Basically, we train a pruned MTL-Seq2Point model with one-month REDD data (for four appliances: washing machine, fridge, dish washer and microwave), which serves as the cloud model trained on source domain. Then, to validate the effectiveness of transfer learning, we leverage the pre-trained cloud model to detect appliances in UK-DALE dataset. Arguably, REDD is distinct from UK-DALE, as the households in the former dataset are in the U.S. while those in the latter dataset are in the U.K.

For the test without applying transfer learning, we directly employ the pre-trained cloud model on UK-DALE for energy disaggregation. For the test with transfer learning, we first retrain the model using the deep CORAL approach and then leverage the retrained model to detect appliances in UK-DALE. With results from the above two tests, we can compare their performance and see whether transfer learning helps. The results are shown in Table III, from which we could observe that: for both models with sequence length 99 and 499, the model with local transfer learning outperforms the original one by up to 45.84% in MAE, 86.63% in SAE and 40% in F1-score, respectively. In particular, for MAE, the performance is improved for all four appliances with transfer learning; for SAE, the accuracy is improved for three appliances; for F1-score, transfer learning model shows better performances on dish washer and microwave and comparable performance on fridge. More specifically, the F1-scores are improved or comparable for 7 out of 8 cases using transfer learning. In general, local transfer learning takes effect on minimizing the difference in source and target domains, compensating for the performance degradation caused by domain shift.

We can also find that in Table III, dish washer, washing machine and microwave seem to have good MAE yet bad F1-score, compared to fridge. Arguably, these three appliances contributes significantly less to the overall consumption than fridge in the dataset. Therefore, their power signals may be recognized as noise in the mains readings. Also, they are not as frequently used as the fridge. Hence, it is hard to disaggregate their readings from the mains readings. All these factors explain the poor F1-score results obtained in Table III. Indeed, the low F1-score of occasionally-used appliances is one of the key challenges in tackling NILM problem [22], [40], and we leave it as our future work.

In our implementation on the edge device, the model training time for each appliance is around 100 seconds in average, whereas the inference merely takes less than one second. During each local updating process, once receiving the cloud model, the edge client could perform transfer learning based on its own power readings to fine-tune the model parameters in dense layers, and conduct energy disaggregation with this personalized model promptly. Meanwhile, the updated local model would be uploaded to the cloud for further federated averaging, with the goal to collaboratively train more powerful and up-to-date cloud models.

D. FedNILM

In the FedNILM framework, the cloud model is trained on data of house 2 in REFIT, data of house 1 in UK-DALE and data of house 1 in REDD. The local clients are comprised of houses 5 and 6 in REFIT, houses 2 and 5 in UK-DALE and houses 2 and 3 in REDD, which is a variant of the framework in Fig. 2 with one cloud server and 6 clients. The cloud model initiation as well as the model fusion are performed at the cloud end, a powerful server with four Tesla V100 GPUs for cloud model training. The local transfer learning models are obtained on an edge device of the client end, a NVIDIA Jetson Nano with a 1.43 GHz CPU and 4 GB RAM.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
multi-task learning structure and prune 60% of parameters in convolutional layers. Then, the pruned cloud model is distributed to local servers to perform transfer learning and update local parameters based on local data. We set to fine-tune local parameters with 7-day user data in each iteration. To be more specific, the local transfer learning model uses the following hyperparameters. The learning rate is $1.0 \times 10^{-4}$, the batch size is 64, and we train 60 epochs for this multi-task structure model to ensure convergence. Meanwhile, the Adam optimiser is employed for faster convergence. Then the updated local parameters of all 6 houses are uploaded to the cloud end for federated averaging (based on Equation (5)).

Moreover, we adopt homomorphic encryption in the FedNILM system to avoid data leakage during uplink and downlink data transmissions.

In order to demonstrate the effectiveness of FedNILM, we choose three baselines: Recurrent Neural Network (RNN) [8], Denoising Autoencoder (DAE) [8] and Seq2Point [9]. The FedNILM results for each client is shown in Fig. 7. We can observe that FedNILM achieves much better performance than RNN and DAE, especially for MAE and SAE. Compared with Seq2Point, FedNILM reaches comparable while more stable performance on all 6 clients, indicating a better generalization ability. Indeed, the multiple information obtained from the distributed clients contributes to a more general server model. Meanwhile, local models become more personalized to clients thanks to transfer learning. The above reasons explain the decent performance of FedNILM, and the results demonstrate the effectiveness of our framework.

Fig. 7. (a) The MAE of RNN, DAE, Seq2Point (S2P) and FedNILM (lower is better). (b) The SAE of RNN, DAE, S2P and FedNILM (lower is better). (c) The F1-score of RNN, DAE, S2P and FedNILM (higher is better). The x-axis indicates 6 clients in our experiment.

VIII. CONCLUSION

We presented FedNILM, a federated learning paradigm designed for NILM applications towards the green home vision. FedNILM realized data privacy-preserving through federated learning, efficient model compression via filter pruning and multi-task learning, and personalized model building by unsupervised transfer learning. Experimental results on real-world energy data demonstrate that FedNILM can achieve accurate and personalized energy disaggregation without compromising users' privacy. Future research may be extended in two possible directions. The first is to employ more agile neural networks to tackle the low F1-score issue of occasionally-used appliances. The second might be increasing the aggregation efficiency by clustering our clients and specifying hyper-parameters for each cluster.

REFERENCES

[1] G. W. Hart, “Nonintrusive appliance load monitoring,” Proc. IEEE, vol. 80, no. 12, pp. 1870–1891, Dec. 1992.
[2] C. Fischer, “Feedback on household electricity consumption: A tool for saving energy?” Energy Efficiency, vol. 1, no. 1, pp. 79–104, 2008.
[3] K. Ehrhardt-Martinez, K. A. Donnelly, and S. Laitner, Advanced Metering Initiatives and Residential Feedback Programs: A Meta-Review for Household Electricity-Saving Opportunities, Amer. Council Energy-Efficient Econ., Washington, DC, USA, 2010.
[4] J. E. Froehlich, E. C. Larson, S. Gupta, G. A. Cohn, M. S. Reynolds, and S. N. Patel, “Disaggregated end-use energy sensing for the smart grid,” IEEE Pervasive Comput., vol. 10, no. 11, pp. 28–39, Jan.-Mar. 2011.
[5] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, “Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey,” Sensors, vol. 12, no. 12, pp. 16838–16866, 2012.
[6] G. Tang, K. Wu, J. Lei, and J. Tang, “A simple model-driven approach to energy disaggregation,” in Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm), 2014, pp. 566–571.
[7] G. Tang, K. Wu, and J. Lei, “A distributed and scalable approach to semi-intrusive load monitoring,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 6, pp. 1553–1565, Jun. 2016.
[8] J. Kelly and W. Knottenbelt, “Neural NILM: Deep neural networks applied to energy disaggregation,” in Proc. 2nd ACM Int. Conf. Embedded Syst. Energy-Efficient Built Environ., New York, NY, USA, 2015, pp. 55–64. [Online]. Available: https://doi.org/10.1145/2821650.2821672
[9] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, “Sequence-to-point learning with neural networks for nonintrusive load monitoring,” 2016, arXiv:1612.09106.

[10] C. Shin, S. Joo, J. Yim, H. Lee, T. Moon, and W. Rhee, “Subtask gated networks for non-intrusive load monitoring,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 1150–1157.

[11] S. D. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, China’s Cyber Power, 2020, pp. 1–7.

[12] B. Sun, N. Bai, H. Wang, and K. Whitehouse, “A tree-structured neural network model for household energy breakdown,” in Proc. World Wide Web Conf., 2019, pp. 2872–2878.

[13] G. Bejarano, D. DeFazio, and A. Ramesh, “Deep latent generative models for energy disaggregation,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 850–857.

[14] A. Li, H. Cui, Y. Shang, and F. Eliassen, “Generative adversarial networks and transfer learning for non-intrusive load monitoring in smart grids,” in Proc. IEEE Int. Conf. Commun. Control Technol. Smart Grids (SmartGridComm), 2020, pp. 1–7.

[15] K. Yu, Z. Guo, Y. Shen, W. Wang, J. C.-W. Lin, and T. Sato, “Secure artificial intelligence of things for implicit group recommendations,” IEEE Internet Things J., vol. 9, no. 4, pp. 2698–2707, Feb. 2022.

[16] “General Data Protection Regulation (GDPR),” European Union. 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT

[17] N. Inkster, China’s Cyber Power. Abingdon, U.K.: Routledge, 2016.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Int. Conf. Artif. Intell. Stat., 2017, pp. 1273–1282.

[19] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau, Y. Ji, and J. Li, “Federated learning for vehicular Internet of Things: Recent advances and open issues,” IEEE Open J. Comput. Soc., vol. 1, pp. 45–61, 2020.

[20] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji, “Computation offloading in beyond 5G networks: A distributed learning framework and applications,” IEEE Wireless Commun., vol. 28, no. 2, pp. 56–62, Apr. 2021.

[21] S. Agrawal, S. Sarkar, M. Alazab, P. K. R. Maddikunta, T. R. Gadekallu, and Q.-V. Pham, “Genetic CFL: Optimization of hyper-parameters in clustered federated learning,” 2021, arXiv:2107.07233.

[22] H. Wang, C. Si, and J. Zhao, “A federated learning framework for non-intrusive load monitoring,” 2021, arXiv:2104.01618.

[23] R. Kukunuri et al., “NILM: An open source toolkit for non-intrusive load monitoring,” in Proc. 5th Int. Conf. Future Energy Syst., 2014, pp. 265–276.

[24] J. Z. Kolter and M. J. Johnson, “REDD: A public data set for energy disaggregation research,” in Proc. KDD Workshop Data Min. Appl. Sustain. (SustKDD), vol. 25, 2011, pp. 59–62.

[25] J. Kelly and W. Knottenbelt, “The U.K.-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five U.K. homes,” Sci. Data, vol. 2, no. 1, pp. 1–14, 2015.

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

[27] D. K. Dennis et al. “EdgeML Machine Learning for Resource-Constrained Edge Devices.” [Online]. Available: https://github.com/Microsoft/EdgeML (Accessed: Jan. 2020).

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and P. H. Graf, “Pruning filters for efficient ConvNets,” 2016, arXiv:1608.08710.

[29] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up convolutional neural networks using fine-tuned CP-decomposition,” 2014, arXiv:1412.6553.

[30] D. Bláloch, J. J. G. Ortiz, J. Franke, and J. Gottag, “What is the state of neural network pruning?” 2020, arXiv:2003.03032.

[31] J. Guo, C. Yu, Y. Li, G. Srivastava, and J. C.-W. Lin, “Deep learning-enhanced social Internet of Things for ambiguity-aware social recommendations,” IEEE Trans. Netw. Sci. Eng., early access, Jan. 5, 2021, doi: 10.1109/TNSE.2021.3049262.

[32] R. L. Rivest, L. Adleman, and M. L. Dertouzos, “On data banks and privacy homomorphisms,” Found. Secure Comput., vol. 4, no. 11, pp. 169–180, 1987.

[33] S. Ruder, “An overview of multi-task learning in deep neural networks,” 2017, arXiv:1706.05098.

[34] R. Caruana, “Multitask learning,” Mach. Learn., vol. 28, no. 1, pp. 41–75, 1997.

[35] H. Zhou, J. M. Alvarez, and F. Porikli, “Less is more: Towards compact CNNs,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 662–677.

[36] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, “Learning structured sparsity in deep neural networks,” in Advances in Neural Information Processing Systems, vol. 29. Red Hook, NY, USA: Curran, 2016, pp. 2074–2082.

[37] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in Advances in Neural Information Processing Systems. Red Hook, NY, USA: Curran, 2014, pp. 3320–3328.

[38] B. Sun, J. Feng, and K. Saenko, “Return of frustratingly easy domain adaptation,” in Proc. AAAI Conf. Artif. Intell., vol. 30, 2016, pp. 2058–2065.

[39] B. Sun and K. Saenko, “Deep coral: Correlation alignment for deep domain adaptation,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 443–450.

[40] D. Murray, L. Stankovic, and V. Stankovic, “An electrical load measurements dataset of united kingdom households from a two-year longitudinal study,” Sci. Data, vol. 4, no. 1, pp. 1–12, 2017.

[41] N. Batra et al., “NILM: An open source toolkit for non-intrusive load monitoring,” in Proc. 5th Int. Conf. Future Energy Syst., 2014, pp. 265–276.

Yu Zhang received the B.S. degree in statistics from Central South University, Changsha, Hunan, China, in 2019. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. She was a Research Intern with Peng Cheng Laboratory, Shenzhen, China, from June 2020 to January 2021. Her main research interests include federated learning and machine learning in EDA.

Guoming Tang (Member, IEEE) received the bachelor’s and master’s degrees from the National University of Defense Technology, China, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from the University of Victoria, Canada, in 2017. He is a Research Fellow with Peng Cheng Laboratory, Shenzhen, Guangdong, China. He was also a Visiting Research Scholar with the University of Waterloo, Canada, in 2016. His research mainly focuses on green computing and computing for green.

Qianyi Huang received the bachelor’s degree in computer science from Shanghai Jiao Tong University, and the Ph.D. degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. She is a Research Assistant Professor with the Southern University of Science and Technology, Shenzhen, China. She is also with the Department of Broadband Communication, Peng Cheng Laboratory. She has published a number of papers in top-ranking journals and conferences, including  IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, MobiCom, UbiComp, and INFOCOM. Her research interests lie in mobile computing, Internet-of-Things, and security.

Yi Wang (Member, IEEE) received the Ph.D. degree in computer science and technology from Tsinghua University in July 2013. He is a Research Associate Professor with the Sustech Institute of Future Networks, Southern University of Science and Technology. He is also with the Department of Broadband Communication, Peng Cheng Laboratory, and Heyuan Bay Area Digital Economy Technology Innovation Center. Before joined Sustech, he worked as a Senior Researcher with the Huawei Network Theory Lab, Hong Kong. His research interests include future network architectures, information-centric networking, software-defined networks, and the design and implementation of high-performance network devices.
Kui Wu (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science, University of Victoria, Canada, in 2002, where he is currently a Full Professor. His research interests include smart grid, mobile and wireless networks, and network performance evaluation.

Keping Yu (Member, IEEE) received the M.E. and Ph.D. degrees from the Graduate School of Global Information and Telecommunication Studies, Waseda University, Tokyo, Japan, in 2012 and 2016, respectively. He was a Research Associate and a Junior Researcher with the Global Information and Telecommunication Institute, Waseda University from 2015 to 2019 and 2019 to 2020, respectively, where he is currently an Assistant Professor. His research interests include smart grids, information-centric networking, the Internet of Things, artificial intelligence, blockchain, and information security.

Xun Shao (Member, IEEE) received the Ph.D. degree in information science from the Graduate School of Information Science and Technology, Osaka University, Japan, in 2013. From 2013 to 2017, he was a Researcher with the National Institute of Information and Communications Technology, Japan. He is currently an Assistant Professor with the School of Regional Innovation and Social Design Engineering, Kitami Institute of Technology, Japan. His research interests include distributed systems and networking. He is a member of IEICE.