FedNILM: Applying Federated Learning to NILM Applications at the Edge

Yu Zhang, Guoming Tang, Qianyi Huang, Yi Wang, Xudong Wang, Jiadong Lou

Abstract-Non-intrusive load monitoring (NILM) helps disaggregate the household's main electricity consumption to energy usages of individual appliances, thus greatly cutting down the cost in fine-grained household load monitoring. To address the arisen 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.

In this paper we present FedNILM, a practical FL paradigm for NILM applications at the edge client. Specifically, FedNILM is designed to deliver privacy-preserving and personalized NILM services to large-scale edge clients, by leveraging i) secure 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 is able to achieve personalized energy disaggregation with the state-of-the-art accuracy, while ensuring privacy preserving at the edge client.

Index Terms-NILM, federated learning, model compression, transfer learning.

I. INTRODUCTION

N ON-INTRUSIVE load monitoring, also known as energy disaggregation. was first proposed 1. U disaggregation, was first proposed by Hart in 1992 [1]. 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 help reduce household energy consumption efficiently. Evidences have shown that such feedback of 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 applications, 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, the NILM problem has not been addressed completely, and traditional solutions referring to man-made appliance signatures have run into bottlenecks [5], [6]. Most recently, it shows that the deep neural network (DNN) based approaches could greatly improve the performance of NILM [7]–[10], as neural networks are able to automatically

Y. Zhang, G. Tang, Q. Huang and Y. Wang are with the Peng Cheng Laboratory, Shenzhen, China. X. Wang and J. Lou are with the Chinese University of Hong Kong, Shenzhen, China.

learn appliance features (or appliance signatures), either obvious or latent ones. Various neural network architectures have been proposed for NILM, including denoising autoencoder [7], recurrent neural networks [11], and GAN [12], etc. Among those DNN based methods, the *Seq2Point* model [8], a one-dimensional CNN based auto-encoder architecture, is the current 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. 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 [13], [14], making it high-risk 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, 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 are integrated to build a up-to-date cloud model, thus allowing local models to collectively reap the benefits of cloud model trained from rich data [15]. It seems natural to apply the FL paradigm to the NILM problem for privacy preserving, which was theoretically verified in [16] 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) appear in local homes, rather than those powerful CPU/GPU servers. Hence, as the NILM models usually take large memory and computation overheads, network training on such resourceconstrained edge devices could take prohibitively long time or even be terminated immediately. Since we cannot perform the model fusion at cloud without local updates, the resource

Corresponding author: G. Tang (tanggm@pcl.ac.cn).

lack 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 [17] and UK-DALE [18]). In other words, even with satisfied 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 [19]. 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 [19].

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 [19], 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 be able to implement local computation based on the compressed global state. 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 Sec. 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 real-world NILM applications at the edge. FedNILM aims to provide privacy-preserving and personalized energy disaggregation for the large-scale edge clients, aided by the cutting-edge DNN model compressing and transfer learning techniques at resource constrained edge devices. This paper shows that, the adapted FL framework can be highly preferable to be applied in tackling NILM problems, which is expected to promote NILM applications and make it practical in real-world implementations. Our major contributions can be summarized as follows.

- We propose FedNILM, a FL paradigm designed for realworld NILM applications. Aided by the FL paradigm and with practical considerations, FedNILM is expected to provide scalable NILM services with the state-of-the-art accuracy across large-scale households while retaining data privacy for edge clients.
- We adopt cloud model compression for edge adoption in FedNILM. By exploring how compression techniques influence accuracy and overhead of the NILM model, we are able to effectively cut down the computation cost at the edge while retaining satisfied performance.
- We incorporate unsupervised transfer learning with Fed-NILM 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 the 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 Sec. II, we give the background knowledge and related literature. Sec. III briefs the state-of-the-art NILM model and the federated learning rationale. Then, we present the design of FedNILM in Sec. IV, including the paradigm overview and workflow. Two key operations of FedNILM, cloud model compression and edge model personalization are presented in Sec. V and Sec. VI, respectively. The experimental evaluations are performed in Sec. VII. The paper is concluded in Sec. VIII.

II. BACKGROUND AND RELATED WORK

Recently, there are growing interests on deploying NILM applications and systems by energy service providers, energy aggregators and distribution system operators [12]. With the urgent request of privacy preserving, the FL paradigm has been exploited [16], 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 not do harm to the NILM performance while significantly reducing the computational overhead [20], [21].

There are various algorithms for compressing or pruning neural networks. In [21], the authors leverage filter pruning [22] and tensor decomposition [23] methods to compress the convolutional layers, where the former refers to sparsify the neural network by removing less important parameters and the later is to perform a low-rank decomposition of the learnt filter matrix. Surprisingly, experimental results in [21], [24] show that compressing of 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 personal NILM models still needs to be conducted on the powerful GPU/CPU servers at the cloud end. Thus, one client needs to upload his/her energy data to the cloud for personal 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, which is the most likely case, we would see more or less performance degradation from the NILM model at the client.

To this end, the transferability issue of NILM models has been preliminarily explored. In [19], 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 [12] develop the 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 are in need of labelled training data on target domain. In other words, they cannot been 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 prohibitively expensive 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 $\mathbf{y} = (y_1, y_2, ..., y_T)$, where $y_t \in R_+$. Then, let $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_T^{(i)})$ 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 first N' appliances used widely and consuming the most energy. Then, the power consumption of the remaining appliances can



Fig. 1. (a) The DNN structure adopted by Seq2Point model. (b) The DNN structure for MTL-Seq2Point model, i.e., shared convolutional and fully connected layers for N appliances.

be represented as $\mathbf{u} = (u_1, u_2, ..., u_T)$ and the aggregate power consumption could be represented as follows:

$$y_{t} = \sum_{i=1}^{N'} x_{t}^{(i)} + u_{t} + \epsilon_{t}$$
(1)

where ϵ_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) [8], 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 $\mathbf{y}_{t:t+w-1} = [y_t, y_{t+1}, \cdots, y_{t+w-1}]$ 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 corresponding time window.

Mathematically, for a target appliance i, it assumes that there exists a function $f^{(i)} : \mathbb{R}^w_+ \to \mathbb{R}^1_+$, and the function gives the power estimation of appliance i at time t + w/2 by:

$$f^{(i)}(\mathbf{y}_{t:t+w-1}) = x^{(i)}_{t+w/2} \tag{2}$$

Thus, the key task in Seq2Point is to learn the specific form of function $f^{(i)}$. Once obtained $f^{(i)}$, 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 [7].

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:

$$\underset{\omega}{\operatorname{arg\,min}} f_s(\omega) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(x_i, y_i; \omega)$$
(3)

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

The server then sends the current global state ω 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:

$$\operatorname*{arg\,min}_{\omega^{u}} f_{u}(\omega^{u}) = \frac{1}{n^{u}} \sum_{i=1}^{n^{u}} \mathcal{L}(x_{i}^{u}, y_{i}^{u}; \omega^{u})$$
(4)

where $\{x_i^u, y_i^u\}_{i=1}^{n^u}$ 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 ω^u , would be uploaded to the server. For parameter transmission between cloud and client, homomorphic encryption is usually adopted to avoid information leakage [25].

Then, with enough local updates, the cloud server performs federated aggregation [15] 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 ω_{t+1}^k using its local data. The server then takes a weighted average of all these local models, which can be formulated as:

$$\omega_{t+1} = \frac{1}{K} \sum_{k=1}^{K} \omega_{t+1}^k$$
 (5)

where ω_{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 also 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 served clients at the edge, as illustrated in Fig. 2.

- At **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 **client** side, each takes over the pruned model from cloud, further tailors it through a local transforming process, and adopts the personalized model for individual appliance monitoring.



Fig. 2. Overview of the FedNILM framework.

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 Sec. III-B) and public datasets, the cloud server develops a compressed model with model pruning techniques. Refer to Sec. 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 train a personalized NILM model with their own data at hand. An unsupervised transfer learning method is applied for the local model personalization. Refer to Sec. 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 Sec. 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 Sec. 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 could trigger tremendous resource demands and hardly 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 [26]. Intuitively, leveraging excess information that comes from auxiliary tasks enables the MTL model to perform well in main task [27]. By leveraging MTL techniques in the Seq2Point modeling (we name the variant *MTL-Seq2Point*), we are able to train one single model for all target appliances for NILM applications. This can significantly reduce the resource overhead at the edge device.

More specifically, we leverage hard parameter sharing method in Seq2Point model, where the set of five convolutional layers and one fully connected layers are commonly shared for all appliances. Then after these shared layers, this model diverges to several task-specific layers for different appliances. Refer to Fig. 1(b) for the network structure of MTL-Seq2Point. In contrast with the original Seq2Point model, MTL-Seq2Point could help save large computation and memory overheads at the edge devices in our scenario.

B. Pruned MTL-Seq2Point

In addition to the multi-task learning, we also seek other effective techniques to further compress the NILM model in compatible with the less powerful edge devices. There has been some progress recently towards this direction, including weights pruning, filter pruning [22], neuron pruning and tensor decomposition [23]. As convolutional layers accounting for most of the computation cost in our NILM model [20], [21], we propose to leverage filter pruning in convolutional layers to build a "slimmer" MTL-Seq2Point model while retaining a comparable NILM performance. We name the model *pruned MTL-Seq2Point* in this work.

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 $\mathcal{K} \in \mathbb{R}^{k \times 1}$ (e.g., 10×1 in the first layer). Supposing that the number of output channels in the *i*-th convolutional layer is n_{i+1} , the 3D filters matrix $\mathcal{F}_i \in \mathbb{R}^{n_{i+1} imes n_i imes k}$ could transform the input $\mathrm{X_i} \in \mathbb{R}^{n_i imes w_i}$ into the output $X_{i+1} \in \mathbb{R}^{n_{i+1} \times w_{i+1}}$. 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 $n_{i+1}n_ikw_{i+1}$. By removing one of the n_{i+1} filters in \mathcal{F}_i , we could eliminate $n_i k w_{i+1}$ 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 $n_{i+1}kw_{i+2}$ 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, thus 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 \mathcal{L}_1 -norm [22] or \mathcal{L}_2 -norm [28], [29] on convolutional filters. As there are no noticeable differences between these two criteria in filter selection [22], we leverage the \mathcal{L}_1 -norm to



Fig. 3. Pruning a filter results in reduction of computation overhead.

score the filters and prune k% of them in each layer with the least \mathcal{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 the following four steps: i) train a convergent MTL-Seq2Point model, ii) score corresponding convolutional filters based on the sum of \mathcal{L}_1 -norm values, iii) prune the least important filters as per their scores, and iv) retrain the pruned MTL-Seq2Point model (for a certain iterations) 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 upon the cloud dataset, while may be greatly degraded at different client ends. To this end, 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 [30].

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 .
- 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,



Fig. 4. The transfer learning process at the edge client.

as illustrated in Fig. 1. Thus, through procedurally freezing or tweaking these layers, we have (2×7) different transfer learning models, which are then retrained and tested on target domain data. By wrapping up all the results, we have the following observations (refer to Sec. VII-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 applying to a new edger client (target domain), they need to be further fine-tuned. The client model personalization can then be realized by largely referring to the transferred features from convolutional layers and substantially finetuning the fully connected layers for specific appliances.

These observations are coincide with those from [19] that, the features in lower levels of layers are highly transferable as lower layers tend to learn common and coarse information, whereas the features in higher layers are more tailored 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 [31]. 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 [32]. 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 $D_t = u_i$, we aim to align the distribution difference in layer I which generates the σ -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, namely covariance, of the source and target features:

$$\mathcal{L}_{CORAL} = \frac{1}{4\sigma^2} \|C_s - C_t\|_F^2 \tag{6}$$

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

$$\mathcal{L}_{EDGE} = \mathcal{L}_R + \lambda \cdot \mathcal{L}_{CORAL} \tag{7}$$

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.

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 [33], UK-DALE [18] and REDD [17]. All contain similar appliance categories, allowing the evaluation of model transferability. Also, they have been widely applied in previous NILM researches [7], [8], [19], 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 [7]–[9], [21]. In particular, the former two metrics, i.e., MAE and SAE, aim to measure the performance of power consumption estimation, while the F1-score could reflect 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,



Fig. 5. (a) The MAE of transferred models (lower is better). (b) The SAE of transferred models (lower is better). (c) The F1-score of transferred models (higher is better). The x-axis indicates till which layer we transfer the parameters and initialize the rest.

TABLE I MAE results from cloud model compression on REDD dataset (sequence length 499)

Model	Washing m.	Fridge	Dish w.	Micro wave	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

TABLE II MAE RESULTS FROM CLOUD MODEL COMPRESSION ON REDD DATASET (SEQUENCE LENGTH 99)

Madal	Washing	Enidos	Dish	Micro	Time	Size
Widdei	m.	Fluge	w.	wave	(s)	(MB)
Seq2Point	14.97	25.80	19.35	9.16	3.13	55.32
30% pruning	14.59	23.77	11.97	9.86	2.74	27.10
60% pruning	14.73	22.71	16.60	8.65	2.67	8.87
90% pruning	15.46	30.70	32.23	10.52	2.55	0.58
MTL-Seq2Point	17.62	26.78	16.06	14.51	0.83	13.84
30% pruning	17.26	25.40	13.05	12.65	0.75	6.79
60% pruning	18.67	28.78	10.66	15.78	0.72	2.22
90% pruning	19.59	30.45	18.34	18.83	0.74	0.15

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, 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, as 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, the unoptimised model occupies $6 \times$ more space and requires more inference time than the optimised model pruned 60% of filters with similar performance. Same as we have found with the model of sequence length 499, for sequence length 99, the multi-task learning structure helps save nearly three fourths of running time and space, whereas 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 contribute to the ED 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 rather give rise to better NILM results, such as the pruned Seq2Point model performance versus the original Seq2Point model accuracy on distinguishing the washing machine, which 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 Sec. 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. 5, 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.
- Second, we find that the NILM performance of fine-tuning transferred layers is better than fixing them, with lower MAE and SAE and higher F1-score. This indicates that fine-tuning the last dense layers could further help us promote the NILM performance on target domain.
- Finally, we also observe a slight performance improvement as we procedurally transfer layers from *conv1* to *conv5*. Note that if we choose to freeze or fine-tune one particular convolutional layer, the parameters in this convolutional layer are pre-trained in source domain. Thus when transferring to target domain, the transferred layers could also leverage the information pre-trained in source domain, rather than learning from scratch. Arguably, the extra information from source domain enables the transfer learning model to generalize better on target domain.

2) Effectiveness of Transfer Learning: To validate the effectiveness of local transfer learning, particularly the CORAL approach, we conduct experiments on the edge devicxe 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, the REDD is literally distinct from the UK-DALE, as the households in former dataset are located in the US while those in latter dataset are sited in the UK.

For the test without applying transfer learning, we directly employ the pre-trained cloud model on UK-DALE dataset 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 dataset. With results from the above two tests, we can compare their performances and see whether transfer learning takes effect.

The results are shown in Table VII-C2, from which we could observe that: for both model with sequence length 99 and 499, the model with local transfer learning outperforms the original one by up to 45% in MAE, 85% in SAE and 40% in F1score, 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 performance on dish washer and microwave and comparable performance on fridge. In general, local transfer learning takes effect on minimizing

TABLE III THE EFFECTIVENESS OF TRANSFER LEARNING

	Metric	None		Transfer		Average	
Appliance		Transfer		Learning		Improvement	
		99	499	99	499	Improvement	
Washing Machine	MAE	38.83	51.59	25.43	23.54	45.84%	
	SAE	7.79	1.58	0.87	0.78	82.39%	
	F1-score	0.47	0.30	0.25	0.31	-27.27%	
Fridge	MAE	68.78	68.75	55.67	66.95	10.84%	
	SAE	11.40	11.15	0.88	2.36	86.63%	
	F1-score	0.58	0.58	0.58	0.58	0	
Dish Washer	MAE	13.54	14.21	11.18	10.29	22.63%	
	SAE	1.17	0.76	2.21	2.49	-143.52%	
	F1-score	0.05	0.18	0.02	0.31	43.48%	
Microwave	MAE	41.87	31.85	25.21	23.76	33.57%	
	SAE	3.01	2.17	2.08	1.97	21.81%	
	F1-score	0.30	0.15	0.31	0.32	40.00%	

the difference in source and target domains, compensating for the performance degradation due to domain shift.

In our implementations 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.

VIII. CONCLUSION

We presented FedNILM, a federated learning paradigm designed for privacy-preserving and personalized NILM applications at low-end edge devices. 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, respectively. The results from the experiments on real-world energy data demonstrate that, FedNILM can achieve accurate and personalized energy disaggregation without compromising the user privacy.

REFERENCES

- G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 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, S. Laitner *et al.*, "Advanced metering initiatives and residential feedback programs: a meta-review for household electricity-saving opportunities." American Council for an Energy-Efficient Economy Washington, DC, 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 Computing*, 2011.
- [5] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini, "The eco data set and the performance of non-intrusive load monitoring algorithms," in *Proceedings of the 1st ACM conference on embedded* systems for energy-efficient buildings, 2014, pp. 80–89.
- [6] M. Zhong, N. Goddard, and C. Sutton, "Interleaved factorial nonhomogeneous hidden markov models for energy disaggregation," arXiv preprint arXiv:1406.7665, 2014.

- [7] J. Kelly and W. Knottenbelt, "Neural nilm: Deep neural networks applied to energy disaggregation," in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, ser. BuildSys '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 55–64. [Online]. Available: https://doi.org/10.1145/2821650.2821672
- [8] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, "Sequenceto-point learning with neural networks for nonintrusive load monitoring," 2016.
- [9] C. Shin, S. Joo, J. Yim, H. Lee, and W. Rhee, "Subtask gated networks for non-intrusive load monitoring," *Proceedings of the AAAI Conference* on Artificial Intelligence, vol. 33, pp. 1150–1157, 2019.
- [10] Y. Jia, N. Batra, H. Wang, and K. Whitehouse, "A tree-structured neural network model for household energy breakdown," in *The World Wide Web Conference*, 2019.
- [11] G. Bejarano, D. DeFazio, and A. Ramesh, "Deep latent generative models for energy disaggregation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 850–857.
- [12] A. M. Ahmed, Y. Zhang, and F. Eliassen, "Generative adversarial networks and transfer learning for non-intrusive load monitoring in smart grids," in 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2020, pp. 1–7.
- [13] E. Union, "General data protection regulation (gdpr)," https://eur-lex. europa.eu/legal-content/EN/TXT, 2016.
- [14] N. Inkster, China's Cyber Power. USA: Routledge, 2016.
- [15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273– 1282.
- [16] H. Wang, C. Si, and J. Zhao, "A federated learning framework for non-intrusive load monitoring," arXiv preprint arXiv:2104.01618, 2021.
- [17] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," *Sustkdd*, vol. 25, 2011.
- [18] J. Kelly and W. Knottenbelt, "'uk-dale': A dataset recording uk domestic appliance-level electricity demand and whole-house demand," *eprint* arxiv, 2014.
- [19] M. D'Incecco, S. Squartini, and M. Zhong, "Transfer learning for nonintrusive load monitoring," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1419–1429, 2019.

- [20] D. K. Dennis, S. Gopinath, C. Gupta, A. Kumar, A. Kusupati, S. Patil, and H. Simhadri, "Edgeml machine learning for resource-constrained edge devices," URL https://github. com/Microsoft/EdgeML. Retrieved January, 2020.
- [21] R. Kukunuri, A. Aglawe, J. Chauhan, K. Bhagtani, R. Patil, S. Walia, and N. Batra, "Edgenilm: towards nilm on edge devices," in *Proceedings* of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, 2020, pp. 90–99.
- [22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710, 2016.
- [23] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cpdecomposition," arXiv preprint arXiv:1412.6553, 2014.
- [24] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" arXiv preprint arXiv:2003.03033, 2020.
- [25] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.
- [26] S. Ruder, "An overview of multi-task learning in deep neural networks," arXiv preprint arXiv:1706.05098, 2017.
- [27] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [28] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *European Conference on Computer Vision*. Springer, 2016, pp. 662–677.
- [29] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information* processing systems, vol. 29, pp. 2074–2082, 2016.
- [30] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in Advances in neural information processing systems, 2014, pp. 3320–3328.
- [31] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
- [32] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European conference on computer vision*. Springer, 2016, pp. 443–450.
- [33] D. Murray, L. Stankovic, and V. Stankovic, "An electrical load measurements dataset of united kingdom households from a two-year longitudinal study," *Scientific data*, vol. 4, no. 1, pp. 1–12, 2017.