SemiPFL: Personalized Semi-Supervised Federated Learning Framework for Edge Intelligence

Arvin Tashakori\textsuperscript{a}, Graduate Student Member, IEEE, Wenwen Zhang\textsuperscript{b}, Graduate Student Member, IEEE, Z. Jane Wang\textsuperscript{c}, Fellow, IEEE, and Peyman Servati\textsuperscript{d}, Member, IEEE

Abstract—Recent advances in wearable devices and Internet of Things (IoT) have led to massive growth in sensor data generated in edge devices. Labeling such massive data for classification tasks has proven to be challenging. In addition, data generated by different users bear various personal attributes and edge heterogeneity, rendering it impractical to develop a global model that adapts well to all users. Concerns over data privacy and communication costs also prohibit centralized data accumulation and training. We propose SemiPFL that supports edge users having no label or limited labeled data sets and a sizable amount of unlabeled data that is insufficient to train a well-performing model. In this work, edge users collaborate to train a Hyper-network in the server, generating personalized autoencoders for each user. After receiving updates from edge users, the server produces a set of base models for each user, which the users locally aggregate using their own labeled data set. We comprehensively evaluate our proposed framework on various public data sets from a wide range of application scenarios, from wearable health to IoT, and demonstrate that SemiPFL outperforms state-of-the-art federated learning frameworks under the same assumptions regarding user performance, network footprint, and computational consumption. We also show that the solution performs well for users without label or having limited labeled data sets and increasing performance for increased labeled data and number of users, signifying the effectiveness of SemiPFL for handling data heterogeneity and limited annotation. We also demonstrate the stability of SemiPFL for handling user hardware resource heterogeneity in three real-time scenarios.

Index Terms—Edge computing, edge heterogeneity, edge intelligence, federated learning, meta learning, personalized federated learning, semi-supervised learning, sensor analytics, transfer learning.

I. INTRODUCTION

OVER the past years, the evolution of sensor and wearable technologies and the Internet of Things (IoT) devices has led to a wealth of personalized time-series multisensory data, tackling various real-life problems, including human activity recognition \cite{1}, \cite{2}, \cite{3}, sleep stage identification \cite{4}, and fall detection \cite{5}. The success in these application domains relies on supervised learning frameworks \cite{6}, \cite{7}, \cite{8}, where high-quality labeled data is necessary to train classification models in a strictly controlled context. However, acquiring a large amount of personalized labeled data in a centralized server is cost prohibitive, time consuming, and impractical due to complexity and privacy concerns \cite{9}, \cite{10}, \cite{11}. On the other hand, users own devices with different processing powers, which makes training on user devices impractical for applications requiring extensive calculations \cite{12}. In summary, these problems have made it challenging to use an extensive amount of multisensor time-series data for predictive analytics tasks for each edge user.

Addressing the above concerns motivates us to answer the following research questions: Can we develop a framework to help users participate in cooperative training without violating their privacy? Can it perform better than supervised learning methods with a few labeled data points? What is the influence of the number of available labeled instances on quality of training? Can the framework provide personalized models adapted to individual users instead of one global model? What is the impact of the number of users who are collaborating? In case of required processing on the user side, what is the effect of user hardware resource heterogeneity on the performance? And what is the performance of the proposed system compared to other methods?

Federated learning can address the privacy concern regarding sharing of users’ data with a centralized server \cite{10}, \cite{11}, \cite{13}. It has gained much attention in recent years, with applications in detection of industrial anomalies \cite{14}, \cite{15}, air quality sensing \cite{16}, traffic forecast \cite{17}, \cite{18}, resource management in wireless networks \cite{19}, \cite{20}, \cite{21}, \cite{22}, human–computer interaction \cite{23}, smart ocean \cite{24}, COVID-19 detection \cite{25}, \cite{26}, medical imaging \cite{27}, clinical decision systems \cite{22}, \cite{28}, and embedded intelligence \cite{9}, \cite{29}, \cite{30}, \cite{31}, \cite{32}, \cite{33}. In federated learning frameworks, users collaboratively train a global model while keeping their data isolated on the edge devices \cite{34}, \cite{35}, \cite{36}. The first method introduced by Google in 2016, FedAVG \cite{37}, is widely used as the baseline \cite{33}, \cite{38}, \cite{39}. In FedAVG, the server first sends a global model to each user. Later, edge users update the model using their locally labeled data and communicate the results back with the server. Finally, the server updates the global model by averaging the received model parameters from the users. Despite its simple framework, FedAVG demonstrated exceptional performances in several scenarios and applications \cite{34}, \cite{37}, \cite{40}. At the same time, the performance of...
federated learning methods including FedAVG, degrades in scenarios where edge users have heterogeneous attributes, such as 1) heterogeneity in computational power, which means users have different hardware settings, resulting in various processing delays [34], [41], [42], [43]; 2) heterogeneity in the distribution of data and labels [43], [44], [45], [46], [47], [48]; 3) heterogeneity in the learning task [34], [41], [43]; 4) heterogeneity in learning models [29], [43]; and 5) heterogeneity in the number of devices per user [42]. To address this challenge, recent works aimed at developing a personalized model for each user [31], [32], [42], [43], [49], [50]. An overview of personalized federated learning models is illustrated in Fig. 1. In personalized federated learning, users collaborate to develop personalized models instead of generating a global model [33], [38], [51], [52], [53], [54], [55], [56]. However, these works primarily focus on image classification benchmarks, such as CIFAR10 [57] and CIFAR100 [58], and applications, such as multisensor and IoT classifications, are left less unattended. Also, in most of these works, the main assumption is that sufficient labeled data exists at each user side, which may be impractical.

Data annotation is another big challenge to address. Current solutions mainly employ supervised learning for sensor-based predictive analytic tasks. For example, in activity recognition, supervised learning is used to deduce activity categories from time-series sensory input data generated by edge sensors [59], [60], [61]. Data annotation is required by an expert in the field. In practice, we mostly have either no label or some labeled data leaving the bulk of data unlabeled on the edge devices.

Although federated learning can protect user privacy, most of the existing methods rely on high degree of annotated data. However, due to the unpredictably variable characteristics of edge users and devices, achieving labeled data at the edge makes the current solutions impractical. To address these shortcomings, semi-supervised learning, which is defined between supervised and unsupervised learning, is proposed to deal with the insufficient labeled data problem [62]. In semi-supervised learning, we have a small amount of labeled data and a large amount of unlabeled data. There have been multiple attempts to unify federated learning with semi-supervised learning [9], [12], [33]. Zhao et al. [12] considered users have a large amount of unlabeled data, while the server has a set of labeled data. Saeed et al. [9] developed a self-supervised method to learn valuable representations from users’ unlabeled data. None of these work investigates methods to solve the data heterogeneity problem.

In this work, we aim to integrate semi-supervised learning with personalized federated learning for multisensory time-series-based classification. We assume that we have a set of labeled data from different distributions at the server and a small set of labeled data, and a large set of unlabeled data at the users’ side. The proposed framework contains three main steps: first, the server learns to generate a personalized autoencoder using a Hyper-network for each user. Second, the server selects samples close to each user distribution, transforms its data set using the corresponding user’s encoder, and trains a set of base models that map the latent representation to output classes and send them to that specific user. Finally, the user aggregates the received models using its labeled data sets.

Our main contributions are as follows.

1) We propose SemiPFL, a semi-supervised personalized federated learning framework, for edge intelligence. SemiPFL applies to a wide range of scenarios, from supervised learning settings to having no labeled data available at the user side. To the best of our knowledge, our solutions is the first attempt to integrate personalized federated learning and semi-supervised learning for multisensory classifications.

2) To our knowledge, this is the first work developing personalized models using a Hyper-network for multisensory classification.

3) To our knowledge, this is the first work that studies the effect of hardware resource heterogeneity among users.

4) The proposed method outperforms recognition accuracy compared to recent publications using fully connected neural network architectures.

5) We extensively evaluate the proposed framework on publicly available data sets gathered from edge devices, such as smartphones and wearable devices. We show better performance of the proposed method as more users collaborate in the framework and more labeled instances are available on the user side.

II. BACKGROUND AND RELATED WORK

IoT solutions face several challenges, including 1) scarcity and heterogeneity of computational and storage resources on edge devices [9], [12], [42]; 2) heterogeneity of data on edge devices from distribution [33], [42], availability of labeled data; and 3) the requirement for having a smaller and less complex model architecture [9], [33], [42]. The proposed SemiPFL aims at unifying semi-supervised learning with personalized

![Fig. 1. Overview of the general structure of personalized federated learning. The server communicates a personalized model to each user. Each user locally updates the model using their sensory data and returns the fine-tuned model back to the server.](Image)
TASHAKORI et al.: SemiPFL: PERSONALIZED SEMI-SUPERVISED FEDERATED

Fig. 2. Overview of the semi-supervised learning method used in this work. First, we train an autoencoder on the whole labeled and unlabeled data. Then, we use the encoder to transform the data to its latent representation, and we use the labeled data to train a base classifier. The final model is the trained encoder followed by the base classifier.

TABLE I
SUMMARY OF NOTATIONS USED IN THIS STUDY

| Notation | Description |
|----------|-------------|
| $R$      | Number of training rounds |
| $K$      | Number of users |
| $S$      | Number of sensors |
| $W$      | Sliding window size |
| $M$      | Number of different samples available on server side |
| $T$, $T'$| Number of fine tuning epochs on user side |
| $h_{\theta}(\cdot)$ | Hyper-network model |
| $\{(\alpha_i)\}_{i=1}^N$ | User embedding vector |
| $D_S$    | Server data |
| $D_j$    | User $j$ labeled data |
| $U_j$    | User $j$ unlabeled data |
| $E_j$    | User $j$ evaluation data |
| $\eta$, $\mu$ | Learning rate |
| $\tau$   | Sample selection threshold |
| $L^a(\cdot)$ | Autoencoder loss function |
| $L^e(\cdot)$ | Supervised Learning loss function |
| $L^{r_{\theta}}(\cdot)$ | Local loss function for user $j$ |
| $f^a_{\theta_{m,j}}(\cdot)$ | Base model for user $j$ in round $r$ |
| $f^m_{\theta_{m,j}}(\cdot)$ | $m$-th base model for user $j$ in round $r$ |
| $f^a_{\theta_{m,j}}(\cdot)$ | Autoencoder model for user $j$ in round $r$ |
| $X^r_{m,j}$ | Averaging weight for $m$-th base model for user $j$ in round $r$ |
| $f_j(\cdot)$ | Final classification model for user $j$ |
| $\Omega(\cdot)$ | Regularization term |

One major challenge in supervised learning frameworks is the availability of labeled instances. Data annotation is usually time consuming, expensive, and not accessible at each edge point. SemiFL [30] addresses scenarios where users have unlabeled instances and collaborate to generate a global model. The authors assume that a set of labeled instances are available on the server side. FedSCN [9] proposes a self-supervised approach based on wavelet transform (WT) to learn valuable representations from unlabeled sensor inputs. However, both methods try to generate a global model for users, which does not perform well in scenarios where data heterogeneity exists among users.

One popular approach to tackling semi-supervised learning is feature extraction by training an autoencoder on unlabeled instances. Autoencoder, an artificial neural network, learns data representation from unlabeled data. Its objective is to transform the original data to its compressed representation and reconstruct it back to its original form without losing valuable information. An autoencoder $f^a(\cdot) = f^a_{\text{dec}}(f^a_{\text{enc}}(\cdot))$ contains two sections, the encoder $f^a_{\text{enc}}(\cdot)$, which maps instances to their latent representation, and the decoder $f^a_{\text{dec}}(\cdot)$, which reconstructs the original data from its simplified representation. The learning objective of the autoencoder is to minimize the following loss function [63]:

$$L^a(X; f^a(\cdot)) = L^a(X; f^a_{\text{dec}}(f^a_{\text{enc}}(X)))$$

where $L^a(\cdot)$ is the autoencoder loss function, and $X \in \mathbb{R}^{S \times W}$ is the input data, where $S$ is the number of sensors, and $W$ is the window width. This loss function penalizes the output of the autoencoder for being dissimilar to input $X$ [63]. Training a classification model under a semi-supervised learning using the autoencoder requires two steps. In the first step, we train an autoencoder on all labeled and unlabeled instances. Then, using the encoder part of the trained autoencoder, we compress the original data to its latent representation, which provides a compact representation of original instances. Subsequently, we train a base classifier using the transformed labeled data points. The final classifier will be the encoder, followed by the base classifier [30], [63]. Fig. 2 gives an overview of the proposed approach.

B. Personalized Federated Learning

IoT devices, such as smartphones and wearable devices, generate a massive amount of data every day. Traditional
machine learning approaches require us to accumulate user data in a centralized database to train supervised models. However, this task is not practical due to several challenges, such as privacy and bandwidth limitations [9], [40], [64]. Moreover, the growing computational power of edge devices makes them suitable for local computation and data storage [12].

Federated learning seeks to provide the same collaborative training without sharing data. In FedAVG [37], the server aggregates all user models without particular non-iid data operations (2). \( f^j_r(.) \) means user-\( j \) model parameters in the \( r \)th round, \( K \) is the total number of users, and \( f^j_{r+1}(.) \) is the global model parameters for round \( r+1 \) that will be sent to all users at the beginning of round \( r+1 \). However, in IoT applications, heterogeneity always exists among users, such as data heterogeneity or hardware resource heterogeneity, making it challenging to train a global model that performs adequately well for all users

\[
f^r_{r+1}(.) \leftarrow \frac{1}{K} \sum_{j=1}^{K} f^j_r(.) .
\]  

The recent literature focuses on personalized federated learning to address data heterogeneity, generating personalized models for different users. FedBN [65] assumes that each user keeps the local batch normalization while collaborating to generate a global model. FedPer [66] tries to generate a personalized model for each user by preserving some local layers. FedProx [39] is a generalized version of FedAVG. It allows partial information aggregation and adds a proximal term to FedAVG. FedHealth [32] is the first personalized federated learning method introduced for wearable healthcare devices through transfer learning. In Fedhealth, users freeze the global model and fine-tune the last layers using their labeled data set. FedHealth 2 [31] measures users’ similarities with the pre-trained model and then aggregates all weighted models while users keep their batch normalization layer. Flame [42] focuses on a multidevice-environment (MDE), where each user can have multiple devices with different processing power. They propose personalized federated learning for scenarios where users can have sufficient labeled data on one of their edge devices. In PfedHN [52], the authors generate personalized models for users using Hyper-networks. Hyper-networks are one of the widely used meta-learning techniques that output the weights of another target network that performs the learning task [52], [67]. The idea is that the output weights vary depending on the input to the Hyper-network [52], [67].

Hyperl-networks are widely used in various machine learning domains, including language modeling, computer vision, continual learning, hyper-parameter optimization, multiobjective optimization, and decoding block codes [52], [67]. Hyper-networks are naturally suitable for learning a diverse set of personalized models, as Hyper-networks dynamically generate target networks conditioned on the input [52], [67].

In IoT applications, hardware heterogeneity is another major challenge that requires users to have personalized models. However, in the literature, no work has focused on the effect of differences in user processing power on the overall performance of the method. In this work, we provide a baseline to compare the effect of hardware resource heterogeneity on the overall performance of federated learning algorithms, which we will discuss in Section IV.

III. SEMIPFL: PROPOSED FRAMEWORK

This section introduces SemiPFL, our personalized semi-supervised federated learning framework for time-series multisensory data. SemiPFL consists of three main steps: first, the server generates a personalized autoencoder using a Hyper-network for each user. Second, the server selects samples close to each user distribution, transforms its data set using the corresponding user’s encoder, and trains a set of base models that map latent representation to classes and send it to that specific user. In the third and final step, the user aggregates the received models using its labeled data set. We will first describe the problem formulation in subsequent sections and then discuss our training pipeline.

A. Problem Description

In SemiPFL, similar to a semi-supervised learning scenario, we assume that users have a small set of labeled and a large set of unlabeled data sets. We also consider a set of labeled instances in the server. In other words, we assume that we have \( K \) users and each user \( j \) has a small set of labeled data \( D_j \)

\[
D_j = \{X_j^i, Y_j^i\}_{i=1}^{l_j} \quad \forall j \in 1, 2, \ldots, K
\]

where \( \{X_j^i, Y_j^i\} \) are the \( i \)th input and output pairs for user \( j \), respectively. We also have a large unlabeled data set for user \( j \), denoted as \( U_j \)

\[
U_j = \{\hat{X}_j^i\}_{i=1}^{u_j} \quad \forall j \in 1, 2, \ldots, K.
\]

To evaluate our method, we define \( E_j \) as user \( j \)'th evaluation data set, where \( \{\hat{X}_j^i, \hat{Y}_j^i\} \) are the \( i \)th input and output pairs for user \( j \) evaluation data set, respectively

\[
E_j = \{\hat{X}_j^i, \hat{Y}_j^i\}_{i=1}^{e_j} \quad \forall j \in 1, 2, \ldots, K.
\]

Also, there is a high-resolution data set from \( M \) different distributions available on the server side

\[
D_S = \{D_{Sm}\}_{m=1}^{M} = \left\{ \{X_{Sm}^i, Y_{Sm}^i\}_{i=1}^{l_{Sm}} \right\}_{m=1}^{M}.
\]

The goal of traditional federated learning methods such as FedAVG is to find a global model \( f(.) \) through minimizing

\[
L = \text{argmin}_{f(.)} \frac{1}{K} \sum_{j=1}^{K} \frac{1}{l_j} \sum_{i=1}^{l_j} L \left( \hat{Y}_j^i \left( f(X_j^i) \right) \right)
\]

where \( L(\cdot, \cdot) \) denotes the loss function. In traditional models, although we could find a model that minimizes the general loss (7), due to the domain shift between users, the general model would not perform satisfactorily for all users. The goal of SemiPFL is to generate a set of personalized models for each user. Particularly, our objective is to propose a framework where the server provides a personalized model \( \{f_j(.)\}_{j=1}^{K} \) for
In federated learning from the server’s perspective can be personalized from each user. Based on [68], the overall objective of personalized federated learning from the server’s perspective can be formulated as
\[
\{f_j(\cdot)\}_{j=1}^{K} = \text{argmin} \frac{1}{K} \sum_{j=1}^{K} \mathcal{L}_j(f_j(E_j)) + \Omega(f_1, \ldots, f_K)
\]
where \(\mathcal{L}_j(\cdot)\) is user \(j\)'s local loss function (9), and \(\Omega(\ldots)\) denotes a regularization term that differentiates personalized federated learning from separately training \(K\) personalized models. Therefore, our objective is to find personalized models for each user, assuming that we have limited or no labeled data, large set of unlabeled data available on the users’ side, and a set of publicly available labeled data on the server side
\[
\mathcal{L}_j(f_j(E_j)) = \frac{1}{e_j} \sum_{i=1}^{e_j} \mathcal{L}(\hat{y}_j, f_j(X_j)).
\]

### B. Training Pipeline

This section introduces SemiPFL, our novel personalized semi-supervised federated learning framework. It consists of several steps that require communication between the server and users. The overall system model can be found in Fig. 3.

The server has a list of user embeddings \(\{a_j\}_{j=1}^{K}\) for every user. At the first step, the server randomly selects one user in every round and generates a personalized autoencoder model with its Hyper-network model using user embedding \(a_j\), Hyper-network parameters \(\theta\), and sends it to the selected user (10). The Hyper-network implementation is the same as in [52]
\[
f^{a_j}_{\theta}(\cdot) = h(a_j, \theta).
\]
In the second step, the user updates the received autoencoder model using the whole unlabeled and labeled data set over \(T\) epochs and sends the updated autoencoder back to the server. In every epoch, the server selects a mini-batch \(\gamma \subset U \bigcup_{j=1}^{K} \{X | (X, Y) \in D_j\}\) and updates the received autoencoder via
\[
f^{\gamma}_{\theta}(\cdot) \leftarrow f^{\gamma}_{\theta}(\cdot) - \eta \nabla f^{\gamma}_{\theta}(\cdot) \mathcal{L}^a(\gamma).
\]

In the third step, the server updates the Hyper-network parameters \(\theta\) and the corresponding user embedding feature \(a_j\). This means that the server calculates the difference between the sent and the received model parameters and updates the Hyper-network parameter and user embedding as in [52]
\[
\Delta f^{\gamma}_{\theta}(\cdot) \leftarrow f^{\gamma}_{\theta}(\cdot) - f^{\gamma}_{\theta}(\cdot)
\]
\[
\theta \leftarrow \theta - \mu \nabla \theta \Delta f^{\gamma}_{\theta}(\cdot)
\]
\[
a_j \leftarrow a_j - \zeta \nabla a_j \Delta f^{\gamma}_{\theta}(\cdot).
\]

In the fourth step, the server generates a set of \(M\) personalized classifiers for user \(j\). The server owns a labeled data set from \(M(\ll K)\) different distributions. For each of those \(M\) data sets, the user first selects samples more similar to the user \(j\) data set. In order to do that, the server checks the distance between its data values and the reconstructed values using user \(j\) fine-tuned autoencoder model through calculating an autoencoder loss function
\[
D^{(j)}_{Sm} \leftarrow \{(X, Y) | \mathcal{L}^a(X, f^{a_j}_{\theta}(X)) < \tau, (X, Y) \subset D_{Sm}\}
\]
In the fifth step, the server encodes its samples using the encoder part of user \(j\) autoencoder
\[
D^{(j)}_{Sm} \leftarrow \{(f^{\gamma}_{r,\text{enc}}(X), Y) | (X, Y) \subset D_{Sm}\}
\]

The server trains a set of base models using selected samples from each of \(M\) available data sets \(\{f^{\gamma}_{\theta, b_j}(\cdot)\}_{b=1}^{M}\) and sends them to the corresponding user. The user initializes a set of weights
for each of these base models and forms an initial personalized base model

\[
\{x_m\}_{m=1}^M = \left\{ \frac{1}{M} \right\}_{m=1}^M
\]

\[
f^{b,j}_r(\cdot) \leftarrow \sum_{m=1}^M x_m \cdot f^{b,j}_r(\cdot).
\]

In the sixth step, the user encodes its labeled instances using its fine-tuned autoencoder that was calculated earlier, freezes base model parameters in (15), and optimizes model weights via (16) using its labeled data set

\[
\{x_m\}_{m=1}^M = \arg\min_{\{x_m\}_{m=1}^M} \frac{1}{L} \sum_{i=1}^L \mathcal{L}_j\left(Y^j_i, f^{b,j}_r(X^i_j)\right)
\]

Subject to: \( \sum_{m=1}^M x_m = 1 \).

A summary of the SemiPFL algorithm can be found in Algorithm 1.

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of our method using sets of available activity recognition, stress detection, and sleep stage scoring data sets. A summary of data sets can be found in Table II.

In order to evaluate SemiPFL in real time, we designed a set of experiments using Raspberry pi 4 boards. We considered the Raspberry pi as the user communicating over a LAN with a PC as the server. In our experiments, we considered three scenarios: 1) all users have system 1 processing power [Fig. 4(a)]; 2) about half of the users are considered with system 1, system 2 hardware specs [Fig. 4(b)]; and 3) we assume around half of the users have system 2 hardware, and the rest are considered with system 3 processing power [Fig. 4(c)]. A comparison between system 1, system 2, and system 3 hardware resources can be found in Table III. To provide a fair comparison, each type of users receives the same processing time.

In this chapter, first, we cite data sets used in this study with their corresponding processing. Second, we explain our experimental setup. Third, we compare SemiPFL with other related federated learning frameworks. Fourth, we explain the impact of the number of available labeled instances, fifth, we evaluate the impact of the number of users collaborating during training, and finally, we study the effect of user hardware heterogeneity on the overall performance.

A. Data Set and Prepossessing

We employed five available human action recognition, one stress detection, and one sleep stage scoring data sets to evaluate our method. A summary of data sets can be found in Table II. In this study, we used the following data sets.

1) Mobiact [69]: Mobiact includes accelerometer, gyroscope, and orientation data gathered from smartphones in participants’ pockets. Twenty activities are recorded, such as standing, walking, jogging, and jumping. In Mobiact, 61 subjects with six trials for each subject are recorded. To compare our results with FedAR [38] and FedSCN [9], we evaluated our algorithm based on two different sets of outputs, five activities: standing, walking, sitting, jumping, and jogging, and
Algorithm 1: SemiPFL

Output: Personalized models: \( f_j(x) \) for \( r = 0, 1, \ldots, R - 1 \)

1. for \( r = 0, 1, \ldots, R - 1 \) do
   1.1. Server randomly select user \( j \) from set \( [K] \);
   1.2. For each user \( j \), \( f_j(x) = h(a_j, \theta) \);
   1.3. For each user \( j \), \( f_j(x) = f_j(x) \);
2. end for

3. for each user \( j \) do
   3.1. User \( j \) sample mini-batch \( \gamma \subset U_j \)
   3.2. For each \( r \) in \( r \)
      \( f_r(\cdot) \Rightarrow f_r(\cdot) - \eta \nabla f_r(\cdot) \gamma \);
   4. end for

5. Freeze \( f_j(x) \) for \( j \in [K] \);

6. Initialize \( \{x_m \}_{m=1}^M = \left\{ \frac{1}{MM} \right\} \)

7. for each user \( j \) do
   7.1. User \( j \) sample mini-batch \( \gamma \subset D_j \)
   7.2. For each \( r \) in \( r \)
      \( f_r(\cdot) \Rightarrow f_r(\cdot) - \eta \nabla f_r(\cdot) \gamma \);
   8. end for

9. For each user \( j \) do
   9.1. User \( j \) sample mini-batch \( \gamma \subset D_j \)
   9.2. For each \( r \) in \( r \)
      \( f_r(\cdot) \Rightarrow f_r(\cdot) - \eta \nabla f_r(\cdot) \gamma \);
   10. end for

11. For each user \( j \) do
   11.1. User \( j \) sample mini-batch \( \gamma \subset D_j \)
   11.2. For each \( r \) in \( r \)
      \( f_r(\cdot) \Rightarrow f_r(\cdot) - \eta \nabla f_r(\cdot) \gamma \);
   11.3. Subject to: \( \sum_{m=1}^M x_m = 1 \);
   11.4. For each \( r \) in \( r \)
      \( f_r(\cdot) \Rightarrow f_r(\cdot) - \eta \nabla f_r(\cdot) \gamma \);
   11.5. For each \( r \) in \( r \)
      \( f_r(\cdot) \Rightarrow f_r(\cdot) - \eta \nabla f_r(\cdot) \gamma \);

end for

end for
Fig. 5. Comparison of SemiPFL and other methods in terms of the average F1 score over different communication rounds for Mobiact [(a) five activities and (b) ten activities], (c) WISDM, (d) HAR-UCI, (e) HHAR, (f) PAMP2, (g) WESAD, and (h) sleep-EDF data sets.

analysis experts to five classes (wake, N1, N2, N3, and rapid eye movement).

B. Experimental Setup

In the past years, many researchers tried to introduce metrics to evaluate performance in federated learning settings [78]. To evaluate our method, we calculated federated average F1 score and Kappa score for all users (18), where \( E_j \) is the evaluation data set for user \( j \), and \( f_j \) is the personalized model for user \( j \), and \( K' \) is the total number of available users during training. We run each scenario ten times with different seed values to calculate each performance metric’s average and standard deviation values. It is important to note that evaluation data is not used during the training phase. Also, since the server randomly selects users, we eliminated those who did not participate in the training session from the final evaluation.

\[
F_{1\text{total}} = \frac{1}{K'} \sum_{j=1}^{K'} F1(E_j, f_j) \tag{17}
\]

\[
\text{Kappa}_\text{total} = \frac{1}{K'} \sum_{j=1}^{K'} \text{Kappa}(E_j, f_j) \tag{18}
\]

We randomly selected a set of users for each data set to be considered server data sets. We kept the same users for our complete analysis to have consistent results. For Mobiact, HAR-UCI, WISDM, HHAR, PAMP2, WESAD, and Sleep-EDF, we selected six, six, six, three, two, three, and two users data sets, respectively, as server instances. We selected our hyperparameters using grid search. For the model architecture, we used four linear layers in the autoencoder (two layers for encoder and two layers for decoder). For the base model, we use two linear layers. After each linear layer, we added a ReLU layer as an activation function and a dropout layer (dropout = 0.2).

We use the Adam optimizer with the learning rate of 0.001. The batch size for training was 128. For threshold value, we selected \( \tau = 0.05 \). We created our data tensor from data sets using a sliding window with length \( W = 30 \). We also normalize and balance class distributions before use.

For all data sets, we considered 30% of the data as an evaluation data set and used the rest for training. We considered 20% of the remaining data points as labeled data sets and the rest as unlabeled data points. We randomly added 10, 20, and 40 labeled datapoints per class, respectively, to our setup and started the training from scratch. We reported all the results at \( r = 200 \) rounds. We also choose \( T = T' = 10 \). For the Hyper-network, we borrowed the structure from pFedHN [52] and updated the architecture so that it outputs our autoencoder parameters.

C. Comparison With Other Methods

1) Comparison in Terms of Overall Performance: Table IV demonstrates a comparison amongst our method and the most related studies investigating the effectiveness of federated learning frameworks for embedded edge intelligence. Here, we have classified these methods based on their objectives: personalized, label in server, label in user, and the model type. In order to be able to accurately compare our results with selected methods, we chose similar preprocessing steps.

Most of these works focus on federated learning, where users collaborate to generate a global model. These methods are not performing well, having lower average Kappa or F1 scores (as seen in Table IV). While SemiPFL uses FCNN which is simpler and faster to train than the more complex architectures, such as CNN and LSTM, it outperforms them all in terms of respective average F1 and Kappa values. SemiPFL performs better than recent semi-personalized work [31], [33], [38], [39], [65], [66], and earlier federated learning methods [9], [12], [37]. It also has a more relaxed attribute regarding the need for labeled data on the user side.

2) Comparison in Terms of Convergence Speed: Fig. 5 demonstrates a comparison of SemiPFL with other methods in average F1-score over communication rounds, using Mobiact 5 activities [Fig. 5(a)], Mobiact 10 activities [Fig. 5(b)], WISDM [Fig. 5(c)], HAR-UCI [Fig. 5(d)], HHAR [Fig. 5(e)],
Fig. 6. Comparison of SemiPFL and FedAVG in terms of inference time for Mobiact (five activities and ten activities), WISDM, HAR-UCI, HHAR, PAMP2, WESAD, and Sleep-EDF data sets.

PAMP2 [Fig. 5(f)], WESAD [Fig. 5(g)], and Sleep-EDF [Fig. 5(h)] data sets. The center lines in plots correspond to average values, and the shaded area represents the standard deviation, calculated over ten different trials. In all data sets, FedAR (blue line) and FedHAR (black line) converge slightly faster than SemiPFL (red line) initially. However, SemiPFL in the same context converges to a higher average F1-score. This can result from having one user communicating with the central server at a time. In both FedAR and FedHAR, every round, a set of users collaborate with the central server. SemiFL (green line) and FedAVG (brown line) have slower or lower convergence values compared to SemiPFL, FedAR, and FedHAR in the average F1-score.

3) Comparison in Terms of Inference Time: Fig. 6 reports a comparison between SemiPFL and FedAVG regarding inference time tested on Raspberry Pi 4 with system 1 specs using Mobiact 5 activities, Mobiact 10 activities, WISDM, HAR-UCI, HHAR, PAMP2, WESAD, and Sleep-EDF data sets. (A summary can be found in Fig. 6.) Similar to [30], which uses autoencoder to train a global model for users, the processing time of SemiPFL is significantly lower ($p < 0.001$) than FedAVG. The reason is that in SemiPFL, users first encode their data set into smaller representations. Despite the fact that the autoencoder adds up to the user’s processing time, the encoder transforms the user data set into a smaller representation, which requires smaller models compared to methods such as FedAVG.

4) Comparison in Terms of Network Footprint: Similar to [79], we have compared the network footprint between SemiPFL, FedAR [38], FedHAR [33], SemiFL [30], and FedAVG[37], based on the average F1-score for all users using Mobiact 5 activities [Fig. 7(a)], Mobiact 10 activities [Fig. 7(b)], WISDM [Fig. 7(c)], HAR-UCI [Fig. 7(d)], HHAR [Fig. 7(e)], PAMP2 [Fig. 7(f)], WESAD [Fig. 7(g)], and Sleep-EDF [Fig. 7(h)] data sets. Based on the results, SemiPFL (blue bar) has less network footprint than FedAR (cyan bar), FedHAR (black bar), SemiFL (red bar), and FedAVG (green bar) in terms of the same average F1 score. One of the main reasons is that in our method, only one user communicates with the server every communication round, making transmission cost more efficient.

D. Impact of Number of Available Labeled Instances

SemiPFL covers a wide range of scenarios from no labeled data to fully supervised setting, unlike other previous publications. This design has lent itself to investigate the effect

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
TABLE V
MOBIACT DATA SET: PERFORMANCE EVALUATION OF SEMI-PFL FOR THE DIFFERENT NUMBER OF USERS AND NUMBER OF AVAILABLE LABELED DATA PER CLASS AT THE USER SIDE. (A) FIVE ACTIVITIES—SCENARIO 1. (B) FIVE ACTIVITIES—SCENARIO 2. (C) FIVE ACTIVITIES—SCENARIO 3. (D) TEN ACTIVITIES—SCENARIO 1. (E) TEN ACTIVITIES—SCENARIO 2. (F) TEN ACTIVITIES—SCENARIO 3

(a)

| Number of labeled instances per class | Metric | AverageStd | Number of system 1 users | number of system 2 users | number of system 3 users |
|--------------------------------------|--------|------------|--------------------------|--------------------------|--------------------------|
| 0                                   | FI score | 92.47(0.06) | 95.31(1.87) | 97.87(1.37) | 93.00(1.05) | 95.31(0.81) | 93.78(0.82) | 94.33(0.48) | 94.89(0.39) | 95.70(0.21) | 96.87(0.09) |
| Kappa                                |        |            |              |              |              |              |              |              |              |              |              |
| 10                                  | FI score | 94.92(2.79) | 94.64(1.77) | 94.75(1.15) | 95.03(0.88) | 95.41(0.48) | 95.90(0.34) | 96.56(0.10) | 97.12(0.24) | 97.86(0.18) | 98.71(0.08) |
| Kappa                                |        |            |              |              |              |              |              |              |              |              |              |
| 20                                  | FI score | 95.22(1.88) | 95.24(3.53) | 95.03(0.90) | 96.00(0.61) | 96.04(0.57) | 96.55(0.44) | 97.02(0.33) | 97.50(0.20) | 98.40(0.33) | 99.34(0.07) |
| Kappa                                |        |            |              |              |              |              |              |              |              |              |              |
| 40                                  | FI score | 97.17(4.77) | 97.27(5.11) | 98.60(1.40) | 98.30(0.48) | 98.40(0.34) | 98.60(0.65) | 98.40(0.50) | 98.50(0.45) | 98.60(0.35) | 98.70(0.13) | 99.07(0.06) |
| Kappa                                |        |            |              |              |              |              |              |              |              |              |              |

of increased available labeling by incrementally adding to the labeled instances per class and measuring the improvements in average Kappa and FI scores as shown in Tables V(a) and (d), VI(a), VII(a), VIII(a), IX(a), X(a), and XI(a). We also see the same trend in Tables V(b), (c), (e), and (f), VI(b) and (c), VII(b) and (c), VIII(b) and (c), IX(b) and (c), X(b) and (c), and XI(b) and (c), where hardware heterogeneity exists among half of users. Having some labeled data sets on the server side in the SemiPFL design enables us to achieve high scores even without any user labels and increasing the performance as the number of labeled data increases. This trend has been observed in this study for different data sets, while in some instances it highlights the possibility to have an optimum number of labeled instances as the performance metrics saturate beyond certain number of labeled data sets.

E. Impact of Number of Users Collaborating During Training

To investigate the impact of the number of users collaborating during the training process, we randomly selected five users for Mobiact and WISDM, and then added five more
TABLE VI
WISDM DATA SET: PERFORMANCE EVALUATION OF SEMIPFL FOR THE DIFFERENT NUMBER OF USERS AND NUMBER OF AVAILABLE LABELED DATA PER CLASS AT THE USER SIDE. (A) SCENARIO 1. (B) SCENARIO 2. (C) SCENARIO 3

| Number of labeled instances per class | Metric | Average(Sd) | Number of system users | number of system users | number of system users | number of system users |
|--------------------------------------|--------|-------------|------------------------|-----------------------|----------------------|----------------------|
|                                      |        |             | 0                      | 10                    | 20                   | 30                   |
| 0                                    |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
| 10                                   |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
| 20                                   |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
| 40                                   |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |

TABLE VII
HAR-UCI DATA SET: PERFORMANCE EVALUATION OF SEMIPFL FOR THE DIFFERENT NUMBER OF USERS AND NUMBER OF AVAILABLE LABELED DATA PER CLASS AT THE USER SIDE. (A) SCENARIO 1. (B) SCENARIO 2. (C) SCENARIO 3

| Number of labeled instances per class | Metric | Average(Sd) | Number of system users | number of system users | number of system users | number of system users |
|--------------------------------------|--------|-------------|------------------------|-----------------------|----------------------|----------------------|
|                                      |        |             | 0                      | 10                    | 20                   | 30                   |
| 0                                    |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
| 10                                   |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
| 20                                   |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
| 40                                   |        |             | 8.90 (1.75)           | 7.63 (1.50)           | 8.15 (1.95)          | 7.95 (1.40)          |
|                                      |        |             | 8.88 (1.79)           | 7.69 (1.55)           | 8.14 (1.91)          | 7.96 (1.41)          |
|                                      |        |             | 8.88 (1.75)           | 7.69 (1.48)           | 8.15 (1.95)          | 7.96 (1.41)          |
TABLE VIII
HHAR DATA SET: PERFORMANCE EVALUATION OF SEMI-PFL FOR THE DIFFERENT NUMBER OF USERS AND NUMBER OF AVAILABLE LABELED DATA PER CLASS AT THE USER SIDE. (A) SCENARIO 1. (B) SCENARIO 2. (C) SCENARIO 3

| Number of labeled instances per class | Metric | Number of system users - number of system users | Number of system users |
|--------------------------------------|--------|---------------------------------|----------------------|
|                                      | Average/Std. | 1.0/2.0 | 2.0/3.0 | 3.0/4.0 | 4.0/5.0 | 5.0/6.0 | 6.0/7.0 |
| 0                                    | F1-score | 87.20/2.0 | 88.92/2.0 | 89.50/2.0 | 89.80/2.0 | 90.20/2.0 |
|                                      | Kappa    | 87.80/2.0 | 88.90/2.0 | 89.20/2.0 | 89.50/2.0 | 89.70/2.0 |
| 10                                   | F1-score | 87.50/2.0 | 88.90/2.0 | 89.00/2.0 | 89.20/2.0 | 89.40/2.0 |
|                                      | Kappa    | 87.80/2.0 | 88.80/2.0 | 89.00/2.0 | 89.20/2.0 | 89.30/2.0 |
| 20                                   | F1-score | 88.00/2.0 | 88.90/2.0 | 89.00/2.0 | 89.20/2.0 | 89.40/2.0 |
|                                      | Kappa    | 87.80/2.0 | 88.80/2.0 | 89.00/2.0 | 89.20/2.0 | 89.30/2.0 |
| 40                                   | F1-score | 88.50/2.0 | 88.80/2.0 | 89.00/2.0 | 89.20/2.0 | 89.40/2.0 |
|                                      | Kappa    | 87.80/2.0 | 88.80/2.0 | 89.00/2.0 | 89.20/2.0 | 89.30/2.0 |

TABLE IX
PAMAP2 DATA SET: PERFORMANCE EVALUATION OF SEMI-PFL FOR THE DIFFERENT NUMBER OF USERS AND NUMBER OF AVAILABLE LABELED DATA PER CLASS AT THE USER SIDE. (A) SCENARIO 1. (B) SCENARIO 2. (C) SCENARIO 3

| Number of labeled instances per class | Metric | Number of system users - number of system users | Number of system users |
|--------------------------------------|--------|---------------------------------|----------------------|
|                                      | Average/Std. | 1.0/2.0 | 2.0/3.0 | 3.0/4.0 | 4.0/5.0 | 5.0/6.0 | 6.0/7.0 |
| 0                                    | F1-score | 85.90/2.0 | 86.90/2.0 | 87.80/2.0 | 88.80/2.0 | 89.20/2.0 |
|                                      | Kappa    | 85.80/2.0 | 86.80/2.0 | 87.70/2.0 | 88.70/2.0 | 89.10/2.0 |
| 10                                   | F1-score | 86.20/2.0 | 87.90/2.0 | 88.80/2.0 | 89.20/2.0 | 89.40/2.0 |
|                                      | Kappa    | 85.80/2.0 | 86.80/2.0 | 87.70/2.0 | 88.70/2.0 | 89.10/2.0 |
| 20                                   | F1-score | 86.50/2.0 | 87.90/2.0 | 88.80/2.0 | 89.20/2.0 | 89.40/2.0 |
|                                      | Kappa    | 85.80/2.0 | 86.80/2.0 | 87.70/2.0 | 88.70/2.0 | 89.10/2.0 |
| 40                                   | F1-score | 86.80/2.0 | 87.90/2.0 | 88.80/2.0 | 89.20/2.0 | 89.40/2.0 |
|                                      | Kappa    | 85.80/2.0 | 86.80/2.0 | 87.70/2.0 | 88.70/2.0 | 89.10/2.0 |

scores and not necessarily decrease the performance due to the edge data heterogeneity. This is an important outcome as it demonstrates the possibility for collaborative learning from edge nodes for the entire SemiPFL model.

F. Impact of User Hardware Resource Heterogeneity

Tables V–XI present our experimental results for scenarios 1–3. Tables V(a) and (d), VI(a), VII(a), VIII(a), IX(a), XI(a), and XI(a) present the overall performance in terms of the F1-score and Kappa score for scenario 1, where all users have the same processing specs. Tables V(b) and (e), VI(b), VII(b), VIII(b), IX(b), X(b), and XI(b) demonstrate the effect of increasing labeled data for users, and the effect of increasing active users during training when we have scenario 2 user hardware heterogeneity. Tables V(c) and (f), VI(c), VII(c), VIII(c), IX(c), X(c), and XI(c) show the overall performance of scenario 3, when the number of labeled data increases, and...
the number of active users increases. One visible trend is that scenario 1 performance is generally always higher than scenario 2, and scenario 2 is always higher than scenario 3 in terms of the average F1-score and Kappa score. However, based on our results, all three scenarios converge to nearly the same values in the reported performance measure as the number of users increases and the number of labeled data increases. We can conclude that our method shows good performance stability when hardware resource heterogeneity exists in scenarios 2 and 3. To our knowledge, this work is the first attempt to quantify such effects, and it is of interest to explore further study of such variations.

V. CONCLUSION

This article introduces SemiPFL, a novel personalized semi-supervised learning method focusing on edge intelligence. Our approach trains a Hyper-network that generates a personalized autoencoder to enable learning from user data representation. Furthermore, based on the user autoencoder and the sets of the
available data set in server side from different distributions, the server generates a group of base models to the corresponding user. Finally, the user fine-tunes the weighted average of such base models to generate a personalized base model. We extensively evaluated the proposed method in three different real-time experimental scenarios on five publicly available human action recognition, one stress detection, and one sleep-stage scoring data sets collected from wearable devices. Our method outperforms personalized models and available federated learning frameworks under the same assumptions in terms of average F1 and Kappa scores. We also compare SemiPFL and other related methods in terms of convergence speed, inference time, and network footprint. In all three categories, SemiPFL demonstrates superior results compared to related methods. We also studied the effect of user hardware heterogeneity in three scenarios and demonstrated stable performance in the presence of such variations. However, this work is the first attempt to quantify such effects and unlock new avenues for further study of such variations. While SemiPFL can perform well for cases without labeled data at the edge, the performance increases with increasing labeled data and saturates signifying an optimum number of labeled data for a given edge node. In addition, we demonstrated that SemiPFL performance increases with increasing number of users unlike other publications highlighting the possibility to incorporate edge data heterogeneity in SemiPFL platform. By leveraging semi-supervised learning, our framework prohibitively reduces the need for annotating data.

REFERENCES

[1] S. Yousefi, H. Narui, S. Dayal, S. Ermon, and S. Valae, “A survey on behavior recognition using WiFi channel state information,” IEEE Commun. Mag., vol. 55, no. 10, pp. 98–104, Oct. 2017.
[2] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, “Sensor-based activity recognition,” IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 42, no. 6, pp. 790–808, Nov. 2012.
[3] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, “Activity recognition from accelerometer data,” in Proc. AAAI, vol. 5, Pittsburgh, PA, USA, 2005, pp. 1541–1546.
[4] A. Supratak, H. Dong, C. Wu, and Y. Guo, “DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG,” IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, pp. 1998–2008, 2017.
[5] M. Mobashir, L. Shao, and L. Seed, “A survey on fall detection: Principles and approaches,” Neurocomputing, vol. 100, pp. 144–152, Jan. 2013.
[6] E. Kim, S. Helal, and D. Cook, “Human activity recognition and pattern discovery,” IEEE Pervasive Comput., vol. 9, no. 1, pp. 48–53, Jan.–Mar. 2010.
[7] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: A survey,” Pattern Recognit. Lett., vol. 119, pp. 3–11, Mar. 2019.
[8] O. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” IEEE Commun. Surveys Tuts., vol. 15, no. 3, pp. 1192–1209, 3rd Quart., 2013.
[9] A. Saeed, F. D. Salim, T. Ozcelebi, and J. Lukkien, “Federated self-supervised learning of multisensor representations for embedded intelligence,” IEEE Internet Things J., vol. 8, no. 2, pp. 1030–1040, Jan. 2021.
[10] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, “Federated learning for Internet of Things: A comprehensive survey,” IEEE Commun. Surveys Tuts., vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.
[11] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, “Federated deep learning for cyber security in the Internet of Things: Concepts, applications, and experimental analysis,” IEEE Access, vol. 9, pp. 138509–138542, 2021.
Wenwen Zhang (Graduate Student Member, IEEE) received the B.A.Sc. degree in electrical engineering from Tianjin University, Tianjin, China. She is currently pursuing the M.A.Sc. degree with the University of British Columbia, Vancouver, BC, Canada.

She is a Research Assistant with the Flexible Electronics and Energy Lab, University of British Columbia. Her research is on pose estimation using wearable devices, focusing on data heterogeneity in federated learning for edge devices.

Z. Jane Wang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Connecticut, Mansfield, CT, USA, in 2002.

She was a Research Associate with the University of Maryland at College Park, College Park, MD, USA, from 2002 to 2004. She is currently a Full Professor with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada. Her research interests include statistical signal processing and machine learning, with applications in digital media and biomedical data analytics.

Dr. Wang is a Fellow of the Canadian Academy of Engineers.

Peyman Servati (Member, IEEE) received the B.A.Sc. degree from the University of Tehran, Tehran, Iran, in 1998, and the M.A.Sc. and Ph.D. degrees from the University of Waterloo, Waterloo, ON, Canada, in 2000 and 2004, respectively.

He is a Professor with the Department of Electrical and Computer Engineering, University of British Columbia (UBC), Vancouver, BC, Canada, whose research focus lies on wearable technology and smart sensing for health, therapeutics, and well-being and focusing on advanced smart textiles, smart fibers, and machine learning. He is the Director of SmarT Innovations for Technology Connected Health and Flexible Electronics and Energy Lab and the Co-Founder of the Centre for Flexible Electronics and Textile, UBC, and the Founder of Texavie Technologies Inc., Vancouver, a leading company in developing advanced wearable health solutions. Prof. Servati is the winner of the 2006 NSERC Canada–U.K. Millennium Research Award, the 2005 NSERC Doctoral Prize, the 2002/03 IEE Institution Premium for the Best Paper in Circuits, Devices, and Systems, and the Bronze Medal in the XXV International Physics Olympiad, Beijing, China, 1994.