Privacy-Preserving, Personalized Federated Learning for EMG Biosignal Decoding

Kai Malcolm
Department of Electrical and Computer Engineering
Rice University
Houston, USA
kai.malcolm@rice.edu

Momona Yamagami
Department of Electrical and Computer Engineering
Rice University
Houston, USA
momona@rice.edu

***Abstract*—Personalized interfaces offer an exciting step forward in consumer and assistive technology. In such a highly-individualized setting, machine learning models that can personalize to individuals may become necessary to reach peak performance in a quick and reliable way. However, such unique biosignal data present a high risk to users' privacy, particularly if all their data are stored on a centralized server vulnerable to breaches or leaks. Thus, we explore a distributed form of this personalization problem, employing federated learning such that each user retains custody of their own data. In this work, we find a global model which outperforms non-federated models and further show that the global model reduces adversarial record linkage capabilities.**

***Index Terms*—federated learning, data privacy, EMG, personalization, interface**

I. Introduction

As our lives increasingly integrate with the digital world and we use more personalized interfaces to improve our lives, the collection of fine-grained multimodal data presents a critical privacy risk. Many applications will benefit from personalization, but this may come at the cost of compromising individual privacy. Common risks include identity theft, leakage of personal health information, facilitation of social engineering, lack of user trust, and direct effects on insurance premiums [11].

A number of works in the past two decades have highlighted the fallacy of anonymization in a digital world: Latanya Sweeney's 2000 paper Simple Demographics Often Identify People Uniquely [10] showed that using only three fields of information (5-digit ZIP code, gender, date of birth), 87% of the US population from the 1990 census (216 million people) were likely to be uniquely identified. This pivotal work showed that anonymization via removing names and other explicit identifiers is not sufficient; Sweeney later famously identified the mayor of Boston (and his corresponding medical issue), after he had to unexpectedly check into a hospital, by combining publicly available voter registration data with an anonymized medical record database she purchased for $20 [11]. In records that contain protected health information (PHI), it is common to strip explicit identifiers but to include "harmless" demographic information: thus, by executing a record linkage attack, supposedly-anonymized datasets existing online from previous breaches can be linked to one another (e.g. by cross-referencing matching demographic information) to facilitate following individuals across datasets and/or extracting other private information. Such leaked datasets may already uniquely identify people (e.g. if they contain name, phone, email, or credit card number leaks) and furthermore provide an attack vector for linking to anonymized sensitive medical data (e.g. diagnoses, hospital stays, etc.) that has been leaked or published. This risk is exceptionally acute for information-dense biosignals which could link to chronic conditions or genetic predispositions, particularly via analyses that may not exist yet but may be developed in the future.

With the advent of neural networks, recent works have shown that an adversary can recover information about model inputs if they have access to the model, which may indicate that it is possible to recover input PHI data [13] [21]. For instance, more advanced attacks (i.e. GAN reconstruction attacks) have been shown to generate synthetic data that is statistically representative of the training data [13]. Clearly, there exists a critical privacy risk to user heath information, both in terms of leaking the acquired biosignals but also in terms of leveraging leaked demographic information to be used in record linkage attacks. Furthermore, these dangers cannot be fully mitigated by traditional data anonymization (e.g. stripping explicit identifiers), and information leakage may come from a number of modalities (e.g. data breaches of the acquired biosignals, adversarial recovery from a released machine learning model, or potentially using other released personalization parameters or statistics to reconstruct unique profiles to be used in record linkage attacks). Other pertinent risks include the disclosure, leak, or inference of sensitive attributes, inferring the membership of a user to a sensitive community, and the lack of plausible deniability for users [5].

Privacy-preserving machine learning systems are important tools for ensuring user trust and safety. Many of today's privacy methods were not designed for data that is not independent and identically distributed (non-IID), as is necessarily the case for biosignal data. This weakness is exacerbated in cases of repeated trials or extended monitoring.

Popular privacy mechanisms include differential privacy, homomorphic encryption, secure multiparty computation, and federated learning, each coming with their own advantages and disadvantages. The purpose of this work is to explore how applying federated learning can maintain data utility while offering users protection from privacy attacks.

The inspiration to this work comes from the original paper which introduced federated learning in 2017 [4], since which others have looked at flaws and extensions relevant to federated learning [19]. A federated setting possesses data distributed across a number of client nodes (hospitals, wearable devices, etc.) that wish to collaboratively train a single model to achieve better performance than any given client could solely by using their own dataset. [4] developed such a so-called global model, which is updated according to gradient-based information received from a few selected clients executing local training in parallel during each communication round according to the FederatedAveraging algorithm (FedAvg). Effectively, this allows clients to train on all of the data in the network without having to directly share that data and open themselves to the accompanying privacy risks. It has been found that when clients have heterogeneous data (e.g. non-trivial differences between client data distributions) that the "vanilla" FL approach is highly sensitive to its parameters and may perform arbitrarily poorly. Thus, personalized FL algorithms have gained popularity, as found in [17] [20] [14] [15].

Personalized FL has exciting implications for applications with high privacy risks, and thus is of particular interest in biosignal-based domains (e.g. EEG/EMG classification and decoding tasks) and other health-related applications where the data is considered protected health information (PHI) under federal law. For example, [20] explores an adaptive mixing parameter for model interpolation in order to arrive at an acceptable personalized model, dynamically finding a weighting between the global and local models in order to create a personalized model for each client. [3] looks to add a "federated adaptation" step to their federated learning approach, such that clients with similar data distributions can be trained in the same group and thus have models trained over data that is the most similar to their own distribution. To do so, they compute clusters using the percent of zeros (output sparsity) from the ReLU output feature map from the first layer in their neural network.

This work makes the following contributions:

1) Compares popular privacy mechanisms (k-anonymity, differential privacy, federated learning, homomorphic encryption, secure multiparty computation), listing their advantages and disadvantages as relevant to continuous time biosignal data.
2) Shows that federated learning is one method to train useful, personalized models in a privacy-preserving fashion.
3) Shows that the weights of the machine learning model (the decoder) can be used to link to participant ID codes.
4) Shows that as the decoder is trained, its ability to be linked to its corresponding participant ID code increases, indicating both a form of personalization and increased privacy risk for the user.

II. Background

A. Motivation

The main question is how to achieve personalization for an individualized interface while maintaining user privacy: here, we define privacy as preventing the linkage between released data and a sensitive attribute (such as a diagnosis) or a personal identifier (in this case, the participant ID code). In this study, each participant trained in multiple disparate trials (with no learning from one trial transferred to the next), and thus the dataset each trial's model trained on was fairly limited: despite some expected heterogeneity between clients, we expect that training a global model will result in a better interface performance, particularly so for initializing the decoder. Furthermore, because of said heterogeneity between clients, we believe that some degree of personalization may be beneficial for achieving optimal performance. Thus, the goal of personalization in this context is to train models which not only start with a lower initial cost, but also converge more quickly to a lower final cost, namely by extending the shared global model by taking a few gradient steps on each local dataset to find a fully personalized model.

B. Privacy Mechanisms

Today's state of the art privacy techniques include anonymization, differential privacy, federated learning, homomorphic encryption, and secure multiparty computation, among others. Together, these methods form the foundation of today's privacy protection standards, are can be found in applications ranging from electronic health records to the 2020 census [7].

1) Anonymization: A number of methods exist for anonymizing entries in database, namely k-anonymity and its progeny, including l-diversity and t-closeness, among others. However, anonymization is typically applied to electronic health records (EHR) and is generally not well-suited for high-frequency, continuous-time numeric data streams [11].

2) Differential Privacy: Differential privacy can be broken down into two broad categories: global and local. Global differential privacy is a query-based model that assumes a trusted data curator who can choose to reject queries if the dataset's privacy budget would be exceeded. Global differential privacy focuses on ensuring that the replacement of any record within the database will have a sufficiently small effect on the output of any computation, thus preventing the query results from being used to infer information about any records in the dataset. On the other hand, local differential privacy does not assume any trusted entity, and aims to ensure that each individual's data is protected no matter who obtains said data.

In compensating for no trusted entity, local differential privacy typically requires much more data perturbation to ensure its more stringent form of privacy, which comes at the cost of data utility. While both methods of differential privacy offer algorithmic guarantees bounding the privacy risk and often preserve privacy well, such methods often incur a drastic reduction in data utility, especially in continuous-collection applications or in scenarios where the number of records is much greater than the number of users [19].

3) Homomorphic Encryption: Homomorphic encryption allows for computations to be made over encrypted data without the need of decrypting the data. Homomorphic encryption methods also fall into two broad categories: partially homomorphic encryption and fully homomorphic encryption. Partially homomorphic encryption only supports either additive or multiplicative operations. Fully homomorphic encryption supports both additive and multiplicative operations, but suffers from higher computation costs [19].

4) Secure Multiparty Computation: This method implements a cryptographic scheme that enables distributed participants to collaboratively calculate an objective function without revealing their data. While this method does not require a trusted third-party and generally can ensure high accuracy without compromising privacy, it has a high computational overhead and high communication costs [19].

5) Federated Learning: Federated learning inverts the traditional training process, such that instead of having clients upload their local data to a server for centralized training, the model is instead sent from the server to each client for local training, and the resulting updates are aggregated in order to train a global model. Notably, clients retain custody of their data and there is no noise or data degradation required, although federated learning is typically used in conjunction with either differential privacy or various forms of encryption.

6) Other Methods: Other methods include secret sharing, garbled circuits, and the more promising but more complex approach of combining multiple privacy mechanisms, which are beyond the scope of this paper [19].

From the above investigation, it was determined that homomorphic encryption and secure multiparty computation would not be sufficient in this real-time scenario, as they suffer from high computational and communication costs and have no inherent benefits for finding personalized interfaces. As discussed, anonymization based methods were also deemed inappropriate for this scenario. Differential privacy presents a promising option, but as explained by [6], it can be extremely difficult to ensure differential privacy is satisfied in a meaningful way, especially in the context of continuous-collection applications where noise accumulation highly degrades the data utility. [6] highlighted how many popular implementations of differential privacy in continuous collection settings experience substantial privacy-protection degradation due to the sequential composition nature of differential privacy. Some of these papers have opted to considering the end of the day or week as a hard reset and thus reset their consumed privacy budget to zero: the validity of this approach remains unclear [8] [18] [2] [6].

Meanwhile, federated learning provides two clear benefits: namely, users retain full custody of their data (limiting the leakage of raw data, requiring adversaries to implement a complex attack model in order to expose basic information), and it is well-suited for large-scale distributed applications where a single entity updates and maintains a number of client devices, each of whom is able to take advantage of a personalized local models. As defined in one of the seminal works on federated learning [4], the ideal application for federated learning trains on real-world data from mobile devices in settings where proxy/synthetic data is not sufficient, the client data is highly privacy sensitive and/or substantially larger in size than the model, and the labels on data can be inferred naturally from user interaction, all of which are true for this application.

C. Federated Learning

Note that in this work, in a departure from the literature, the so-called local model refers to the performance of the global federated model after training on a given client's local data (and thus each client has their own local model), whereas the model trained solely on a given client's local data is referred to as the non-federated model (this would typically be referred to as the local model in the literature).

1) Cross-device Versus Cross-silo: In the cross-device setting, $K >> |D_i|_1 \forall i \in K$, where K is the total number of clients and $|D_i|_1$ is the number of datapoints within the local dataset of client i. Namely, this setting indicates that there are many more clients than datapoints per client, and is common in wearable or monitoring applications where the devices have relatively low storage capabilities and have a sparse connectivity to the network. This work is a small-scale implementation in the cross-device setting.

In the cross-silo setting, $K << |D_i|_1 \forall i \in K$, indicating a small number of clients with large datasets are collaborating, which typically involves more substantial compute resources and likewise a much higher up-time for each client. The cross-silo setting is evident in the scenario where multiple hospitals, who may have thousands of patients each with a wealth of their own data, would like to collaborate to train a shared model without sharing their data [13].

2) Data Partitioning Schemes: Broadly, most federated learning applications can be categorized as either having horizontal data partitioning, vertical data partitioning, or being federated transfer learning. Horizontal data partitioning (sample-based federated learning) indicates that the feature space is shared across clients, but the sample space is not, best represented when all clients have the same numbers of rows (same features across clients) but there are no guarantees about the values present. Horizontal data partitioning is the

most commonly encountered, and encompasses this work. Vertical data partitioning (feature-based federated learning) is the opposite, where clients have the same sample space but not the same feature space and thus clients may have vastly differing features to train on. Federated transfer learning is when neither the feature nor the sample space are shared between clients, and thus this is rarely encountered.

D. Personalized FL

Personalized federated learning offers a promising option for training over heterogeneous client data. In scenarios where all clients sample data from the same distribution (e.g. there is no statistical heterogeneity), federated learning will always outperform purely local models (e.g. models training only on a given client's local data), as the federated implementation gets to effectively train over all the data in the network. Personalization is important because it is often the case in federated applications that clients do not sample the same data distribution, but rather sample multiple similar data distributions. Thus, a key factor is just how similar the underlying data distributions between clients are: clearly, if there is little or no relation, a collaboratively trained global model will perform arbitrarily poorly and likely will not converge [20].

The most prevalent approach for personalized federated learning is local fine-tuning, which includes the popular MAML-based approaches, where a global model is collaboratively trained and afterwards the client takes additional steps (typically gradient-based) to reach a better performance than the shared global model could achieve [1]. Other works have shown promising approaches via multi-task learning, adding contextualization, regularizing the difference between the global and local models, or by interpolating the global and local models in various ways. In addition to the methods discussed above, other exciting approaches include shared representation [9] and split learning [16], both of which center around neural network architectures where a portion of the network is trained individually by the client and then the remaining portion is collaboratively trained and shared by all clients: as these approaches are only applicable for neural network-based approaches, they will not be covered in this work.

1) MAML and Per-FedAvg: Model-agnostic meta-learning (MAML) can be described by altering the optimization problem from Equation 5 to focus on finding a good shared starting point for all clients in the network. As shown in Equation Equation 1, MAML collaboratively finds a meta-model which performs well after a few steps of some form of gradient descent performed by the client, as opposed to finding a model which performs well on all samples in expectation as in Equation 7.

$$\min_{w\in\mathbb{R}^d} F(w) = \frac{1}{K}\sum_{i=1}^{K} f_i(w - \alpha\nabla f_i(w)) \tag{1}$$

2) Adaptive Personalized Federated Learning (APFL): Another popular method for personalization is model interpolation [20], with one example being the APFL algorithm which consists of training a selected group of clients each round, calculating hyperparameters which control the learning rate based on the μ-strong convexity and L-smoothness, and then calculating global ($w_i^{(t)}$), local ($v_i^{(t)}$), and mixed (or personalized) models ($\bar{v}_i^{(t)}$), respectively:

$$\bar{v}_i^{(t)} = \alpha_i v_i^{(t)} - (1-\alpha_i) w_i^{(t)} \tag{2}$$

[20] also recommended implementing an adaptive α such that it dynamically scales based on the calculated values of μ and L which can be derived from the Hessian:

$$\begin{aligned}\alpha_i^{(t)} &= \alpha_i^{(t-1)} - \eta_t \nabla_\alpha f_i(\bar{v}_i^{(t-1)}; \zeta_i^t) \\ &= \alpha_i^{(t-1)} - \eta_t \langle v_i^{(t-1)} - w_i^{(t-1)}, \nabla f_i(\bar{v}_i^{(t-1)}; \zeta_i^t)\rangle\end{aligned} \tag{3}$$

III. Methods

A. Task Description

This study is a secondary data analysis of a larger study investigating co-adaptive learning between users and EMG-controlled interface for a trajectory-tracking task [12]. The goal of this work is to establish the privacy risk (adversarial linkage capabilities) between personalization parameters and unique participant identifiers, and to subsequently introduce machine learning methods for maintaining or improving performance while limiting said linkability.

In the original experiment, 14 participants were tasked with using their forearm muscles to control a cursor to follow a target trajectory displayed on a computer monitor [12]. In order to accomplish this, a linear regression model was trained in order to convert the incoming EMG data into a velocity and ultimately an onscreen position, as described in Equation 4, where v is the user's velocity, F is the EMG input signal matrix, y is the user's position, and D is the decoder matrix (trained model).

$$v(t) = D \cdot s(t), y(t) = y(0) + \int_0^t v(\sigma) d\sigma, \tag{4}$$

There are two main areas of this experiment we hope to extend, particularly in the context of scaling it up for a wider consumer or clinical audience. First, there is no calibration and the recording armband has no guarantee that the orientation is the same as previous trials (e.g. that channels are over the same muscle groups). Thus, with each fresh use, the user has to work through about 1 minute where the interface is not usable. Second, there is no privacy protection for the user: raw EMG data is transmitted unencrypted via a Bluetooth

connection, which is known for being exploitable by common wireless attacks [22]. Thus, our goal is to find a shared model that users can initialize their devices to, circumventing the discussed initial dead period, and subsequently personalize to a high performing interface for each user. In this application, the users do not want to upload their personal biosignal data to a centralized server due to privacy concerns. To simulate the real-life environment, we deliver the collected data in batches to our algorithm, mimicking the incoming batches of data that the real-time trials feature.

To address the potential co-adaptivity, prior work has observed that most learning (measured as improvement in performance) occurs in the first half of each batch sent to the server and then plateaus: thus we only run the federated algorithm on the second half of each update, where the user is considered to have already fully adapted with little to no further co-adaptation taking place for the rest of the update batch. Given that co-adaptation from the human thought to be expressed by changing the strategy employed, we have also opted to focus only on the second half of the entire dataset, at which point the human should have chosen a strategy.

B. Problem Formulation

The canonical problem formulation for federated learning is concerned with a distributed setting in which K clients collaborate to train a shared global model by sending the results of training on their local data to a trusted third-party server for aggregation. Thus, the goal is to find a model w which minimizes the cost associated with the the summed cost functions of each client i.

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{K} \sum_{i=1}^{K} f_i(w) \tag{5}$$

Translating to an empirical realization of $f(w)$, we can solve the above problem by computing the loss associated with training data pairs x and y, which are the samples and labels, respectively.

$$f_i(w) = \mathbb{E}_{(x,y)\sim p_i}[l_i(w; x, y)] \tag{6}$$

Note that a key factor here is that the underlying data distribution of each client, p_i, is unknown, and thus to solve the above we typically combine gradient descent with stochastic gradients utilizing the realized sample data (ξ_i):

$$w_i^{(t)} = w_i^{(t-1)} - \alpha \nabla f_i(w_i^{(t-1)}; \xi_i^t) \tag{7}$$

Following the original work [12], the objective is to train a linear regression model that minimizes the performance error, represented between the difference between the translated velocity (from the human EMG data) and the target velocity (on-screen cursor).

$$c_{perf} = ||V_{user} - V_{target}||_2^2 \tag{8}$$

The translated velocity, V_{user}, is a matrix resulting from the matrix product of the decoder, which has a shape of (2, number of channels), and the current batch of user EMG signals, which is a matrix with size (number of channels, number of time points). Note that the EMG recording device has 64 channels, but this can effectively be reduced via PCA.

$$V_{user} = DF \tag{9}$$

V_{target} is the difference between the velocity of the cursor and of the user. The velocity of the cursor ($\dot{\tau}$) is computed as the derivative of the position of the cursor with respect to time ($\frac{d}{dt}p_{ref}$), and the velocity of the user is the matrix product of the current (fixed) decoder and the incoming EMG input.

$$V_{target} = \dot{\tau} - \dot{y} \tag{10}$$

This can alternatively be framed by comparing the reference and user (so-called actual) positions, which is helpful since the label available for training is the position of the target, not the velocity.

$$p_{actual} = \int_0^t v_{actual} dt = \int_0^t D_{prev} F dt \tag{11}$$

$$c_{perf} = ||DF - \frac{d}{dt}(p_{ref} - \int_0^t D_{prev} F dt)||_2^2 \tag{12}$$

The cost function is a weighted sum of the performance (Equation 12), the norm of the given input data batch, and the norm of the decoder. The norm of the decoder acts as a regularizer, helping to ensure that the resulting cost function satisfies μ-strong convexity. The input data batch is the subset of the total local dataset corresponding to which update the given client is on. Specifically, since the experiment was run in real-time, EMG data was processed and sent to the computer in about 30 second batches, resulting in 19 updates per 5 minute trial. Given that the EMG data is non-IID, this implementation has taken steps to preserve the streaming nature of the data collection. Note that the variable of optimization for the cost function is D, and thus the norm of the input data batch is essentially a constant associated with any given data batch, coming from the primary analysis which viewed this interaction as a two-player game, whereas for this work, we assume there is no co-adaptivity (the user does not learn and adapt to the decoder, only the decoder is being updated). If we assumed the human was also learning, we would then have a second minimization problem for the decoder agent, namely minimizing the cost function with respect to F. Explicitly defining the cost function in terms of the decoder D and the reference position (the label):

$$\begin{aligned} c_{L2} &= \min_D f(D) \\ &= \lambda_E ||DF - \frac{d}{dt}(p_{ref} - \int_0^t D_{prev} F dt)||_2^2 \\ &+ \lambda_D ||D||_2^2 + \lambda_F ||F||_2^2 \end{aligned} \tag{12}$$

Given that D_{prev} is fixed (i.e. separate from our optimization variable D), we can condense all the constant terms and equivalently rewrite the cost function as:

$$\begin{aligned} c_{L2} &= \min_{D} f(D) \\ &= \lambda_E ||DF - V_+||_2^2 + \lambda_D ||D||_2^2 + \lambda_F ||F||_2^2 \end{aligned} \tag{13}$$

Given that the cost function is known analytically, we have also analytically derived the gradient and the Hessian (thus a second order oracle available), which are shown below.

$$\nabla f(D) = 2\lambda_E (DF - V_+)F^T + 2\lambda_D D \tag{14}$$

$$\nabla^2 f(D) = 2\lambda_E (FF^T \otimes I) + 2\lambda_D (I \otimes I) \tag{15}$$

The cost function is convex, as it it is the sum of convex (quadratic) terms, and its convexity exhibits a high dependence on the penalty terms (λ_E, λ_D and λ_F) such that when the penalty terms are larger, the cost function is more convex, but likewise is more likely to minimize the norm of D than improving the performance $DF - V_+$. Staying with the original parameters used in the primary analysis, λ_E, λ_D and λ_F are set to 1E-6, either 1E-3 or 1E-4, and 1E-7, respectively. The cost function is continuous, differentiable, and has been shown to be L-smooth. We rely on a decentralized stopping criterion based of 500 global iterations. Given the federated setup, the associated network formed between the participants (clients) and the server is directed, as clients have a bi-directional connection with the server but have no connection to other clients, and fixed in time for the current deployment. In the lab, we will only be able to collect one person's data at a time, although such real world networks would be time-varying (clients using their devices at different times), which makes asynchrony a necessary future avenue to pursue.

C. Privacy Analysis

The initial focus of this paper was to quantify the amount of personalization (if any) occurring in the data collected by [12]. To this end, we chose to focus on the linkability between the decoder and the participant ID code, which represents the ability of an adversary to link a set of weights (i.e. a decoder matrix) to a specific individual. The decoders from each update were used as training data for a number of machine learning models (logistic regression, k-nearest neighbor, support vector classifier, decision tree, gradient-boosted classifier, and stochastic gradient descent), which we will refer to as the adversarial models. Originally, the Frobenius norm of each decoder was used as training data (paired with its participant ID code), and further testing found more success using flattened versions of the decoder matrices as input. After establishing this capability, the training was further discretized, training models with only decoders from a given update, hypothesizing that the later the update, the more the decoder would be personalized to the user which should correspond to a higher adversarial accuracy. A similar evaluation was conducted with respect to the different condition numbers (related to different parameters tested during the primary study). The adversarial model accuracy was then compared against the user error over each trial.

D. Simulations and Data Processing

1) Simulation: A simulation instantiates client nodes and a server node, with configurable parameters based on what federated and/or local training algorithm and what other hyperparameters are desired. In addition to fully minimizing the current data batch before advancing to the next update as in [12], we tested two other streaming settings: one where the entire dataset of each client is used for training (thus simulating no streaming), and one where the client's current update is advanced after running a set number of iterations, which most closely mirrors the real-time setting of the experiment. For the non-federated experiments, the simulation mimics the optimization approach from [12], running a full minimization for each update for each client

Algorithm 1: Server-Side FederatedAveragingEMG. K clients each represented by k, α is the learning rate, η is the number of local gradient steps, p is the number of principal components, u is the client's current update, r is the client's current round, and is the transition threshold (local round threshold) of each client.

Server executes:
$A_t \leftarrow$ set of all available clients;
for *each round* $t = 1, 2, ...$ **do**
  $S_t \leftarrow$ randomly sample set of $C * K$ clients from A_t;
  foreach *client* $k \in S_t$ **do**
    $r \leftarrow r + 1$;
    if $r > \tau$ **then**
      $u \leftarrow u + 1$;
      $D_u^k \leftarrow$ stream next data update from the full client dataset D^k;
      $\zeta^k \leftarrow \frac{D_u^k}{||D_u^k||_2}$;
      $\zeta^k \leftarrow$ PCA(ζ^k, p) ;
    end
    $w_t^k \leftarrow$ current global model w_t;
    for $n \in \eta$ **do**
      $w_{t,n}^k \leftarrow w_{t,n}^k - \alpha \nabla l(w_{t,n}^k; \zeta^k)$
    end
  end
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end

2) Data Processing Pipeline: For each communication round during the federated learning process, the chosen

client(s) check to see how many rounds they have trained on the current data update. If it exceeds a predetermined limit, the update number is advanced and the client's training dataset likewise advanced to the second half of the next update batch's data, at which point the new client training dataset is then normalized. This normalization is necessary as the norm of the EMG data appears in the cost function, and without this normalization, the cost function is dominated by EMG norm, which the model cannot affect. In order to run larger models or use more training data, it was deemed necessary to implement PCA in order to reduce the number of channels to just the minimum number of meta-channels that can explain 95% of the variance across all 64 original channels. Additionally, PCA may have the added benefit of removing the dependence of specific channels numbers, since the EMG recording device is not calibrated and more is worn without any alignment guarantees. Namely, the recording channels between users and between trials may not necessarily cover the same muscle groups, thus PCA may allow better aggregation between decoders, combining the information contained in the meta-channels instead of being constrained to actual orientation the recording device is in. PCA is applied to reduce the number of channels from 64 to 7. The entire data processing pipeline for each communication round is shown in Figure 1, which shows both the server and client processing steps as well as the federated aggregation.

E. Federated Learning Simulations

The federated algorithm employed in this work is a extension of the basic FedAvg algorithm [4], incorporating necessary processing steps for accommodating EMG data. FedAvg conducts a limited number of gradient steps (Line 14-16) which are shared with the server, and a simple weighted-averaging scheme (based on amount of data each client trained on) is implemented to aggregate all the trained local models into the new resulting global model (see Line 18).

IV. Results

A. Federated Learning

1) Performance Benefits of Federated Learning: Federated learning has the potential for increased performance, as each client effectively gets to train over all the network data as opposed to being constrained to just their local data. As seen when comparing Figures 2 and 3, there is a stark difference in performance between the non-federated and federated implementations: even with random initializations, the federated implementation greatly outperforms the non-federated case, with costs that are orders of magnitude lower. Similarly, using the final client decoders from each of their respective trials, it is again clear that the federated model vastly outperforms the non-federated model. Specifically, the local models from the federated case have nearly zero error whereas the non-federated case essentially retreads the same cost curve as the previous trials. Figure 3 shows that using the collaboratively trained global model in place of the trained global model (the labeled local model) can achieve a very similar performance, generalized across all clients. Notably, using the previously trained global model also beats the federated case with a random initialization.

2) Cost Function Breakdown: From Figure 4, it is clear that the norm of the decoder is of much higher magnitude than the performance and thus experiences the lion's share of the minimization. Since the input data F was normalized before being trained on, its norm over time is essentially flat and is thus excluded from this analysis. Figure 5, it is clear that the gradient monotonically decreases, following a quadratic shape. The small blips that appear in the figure are the result of advancing to the next update, which causes a one-time spike in the gradient (since it is stochastically calculated and thus depends on the data). By changing the parameters involved, the jump caused by switching updates can be changed as well.

3) Parameter Study: From the conducted parameter testing, it is clear that as the learning rate is increased, the cost function (averaged across all clients) decreases faster initially, but ultimately reaches the same final error and does some in the same total number of local iterations. A higher learning rate may be desirable for more quickly getting through the initial dead phase, and it appears to come at no cost to the final number of iterations required or final accuracy achieved.

As discussed earlier, the number of principal components required to explain the desired variance of each client varies substantially. Thus, we investigated the effects of using different numbers of principal components on the cost function. Generally, more principal components (less compression) results in worse performance, until about 10 or so principal components, at which point fewer principal components does not show any substantial benefit.

A final set of experiments looked at the number of local rounds before advancing to the next update, the fraction of clients chosen each communication round, and the number of gradient steps taken each communication round, all of which did not show any effect on the mean cost function. This is unintuitive as we would expect the cost to drop at least with more local computation (number of gradient steps) and with more parallel computation (fraction of clients per round). This may imply that the cost function has many local minimums, which need additional computation to pass.

B. Personalized Federated Learning

1) APFL: A current issue with the APFL implementation is that the gradient does not monotonically decrease as we would expect for our convex cost function, but rather, it explodes when learning rates are too high, which so far the only solution found has been to use arbitrarily low learning rates (e.g. 10^{-12}), although it would appear that this is because the gradient simply cannot make enough progress to explode in the first place. Gradient clipping and other learning rate setting schemes have been implemented without producing the desired effect. A future possibility would be to change the penalty terms (which would likely require a whole parametrization study since there are many hyperparameters that undoubtedly

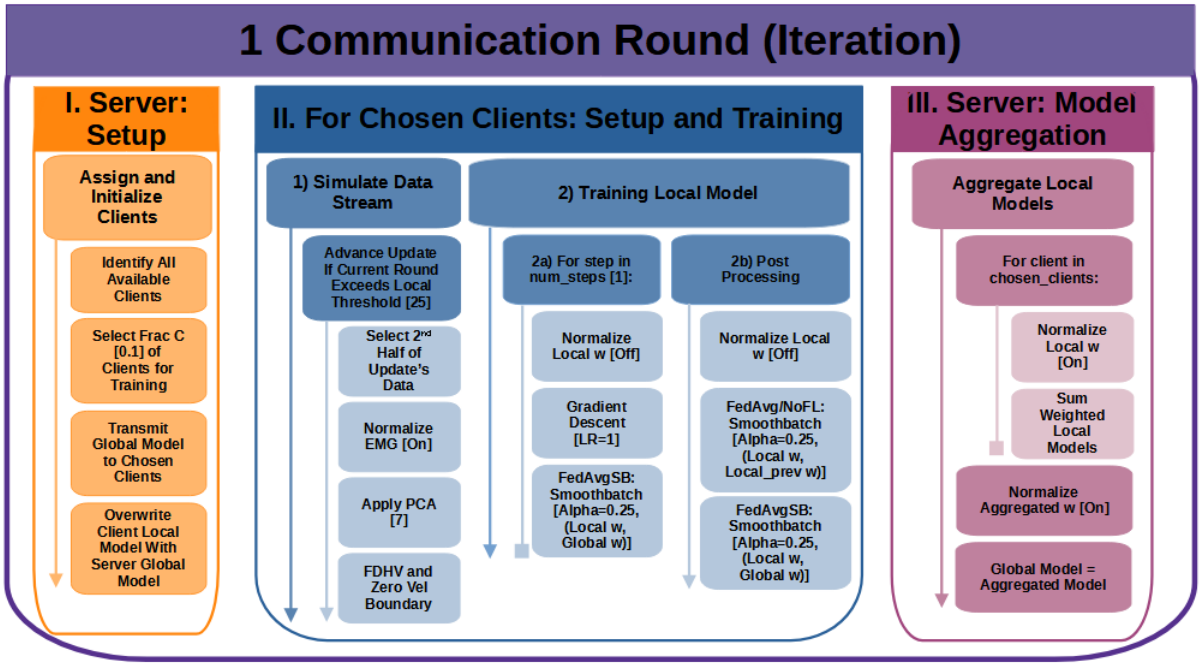

Fig. 1. Full data processing pipeline taking place each communication round between the server and client in the federated setting.

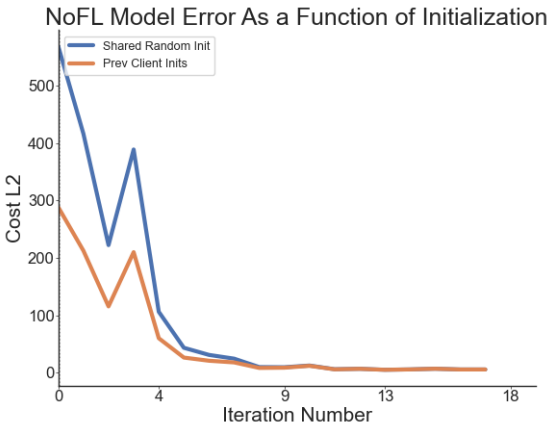

Fig. 2. Mean cost curves from the non-federated model.

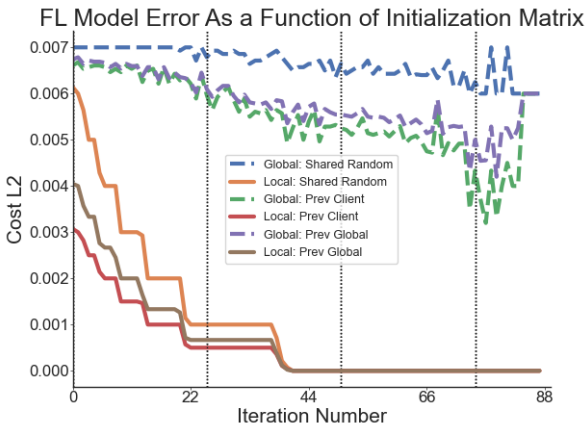

Fig. 3. Mean global and local cost curves from the federated learning model with different initializations.

depend on each other), as well as to assess client heterogeneity and potentially clustering similar clients during training or excluding problematic clients that cause the global model to diverge.

2) FedAvgSB: FedAvgSB takes the general idea from APFL and implements a simpler version, namely by simply interpolating the local and global models at the end of each communication round. Given the linear regime of this application, the results are fairly intuitive: interpolating the models directly results in the same interpolation of their accuracies, as displayed in Figure 6.

3) Per-FedAvg: Another approach was taken, following [1] to create a personalized model, and this is still under development.

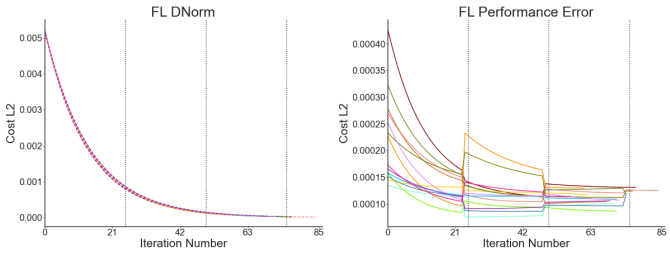

Fig. 4. The two main components of the cost function, shown using the FedAvgEMG algorithm.

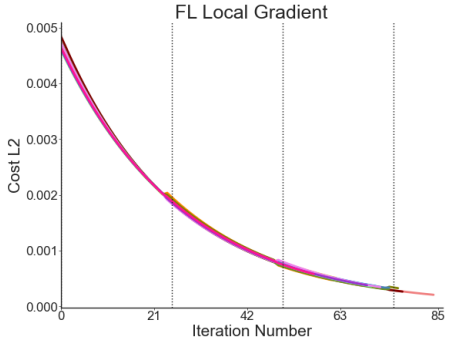

Fig. 5. Gradient of each client collected during the training of the federated model.

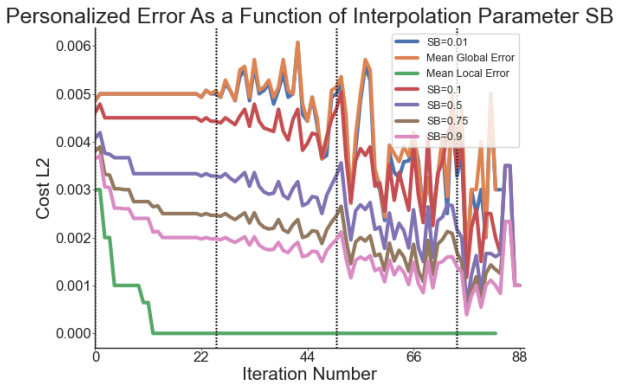

Fig. 6. FedAvgSB: Mean personalized model error for various model interpolation weighting parameters.

C. Privacy Analysis

1) Initial Linkage Attacks: Initially, work focused on evaluating the privacy risk associated with the EMG data as well as the decoders. Figure 7 below shows the initial adversarial model accuracy, representing the ability of an adversary to link a given decoder back to the participant ID code, specifically against decoders coming directly from [12] (e.g. trained without federated learning).

An addition result of interest is showing the linkability of EMG data back to the participant ID codes decreases as more post-processing is applied. Notably, the EMG input data has already been filtered on device, so this second step of processing is not required but can serve to reduce the dimensionality substantially, which is its goal here. In line with the data processing inequality, the less processing that is applied results in higher adversarial linking capabilities, and notably the EMG data linkability is much higher than just the decoder, which can be thought of representing a compressed or simply more processed form of this data.

2) Post-Federated Learning Linkage Attacks: Finally, Figure 9 shows the adversarial model accuracy is highest for the trained global models (e.g. our so-called local models), and lowest for the shared global model, which is in line with expectations since all clients share this initialization each communication round.

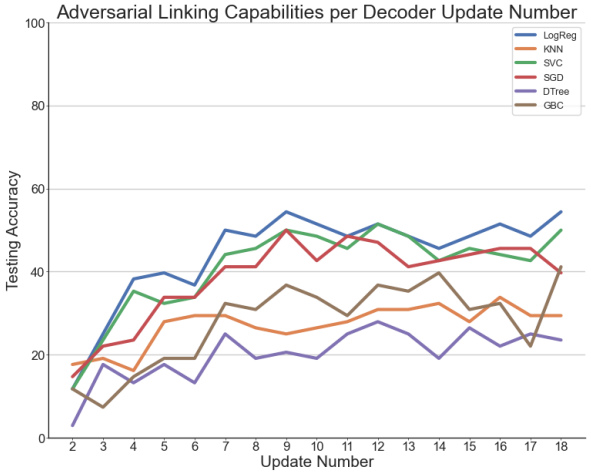

Fig. 7. Initial adversarial model accuracy for linking decoders to each participant.

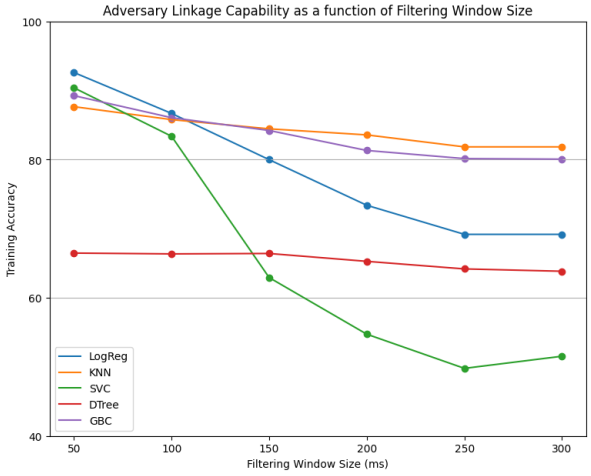

Fig. 8. EMG data is more difficult to link the more the data is processed.

V. Discussion

A. Federated Learning

From Figure 2, it is clear that the previous client models used an initialization do outperform random initializations, but even though the final cost from the random initialization was close to zero, reusing the trained models still results in a high cost. This may indicate that there was little real learning retention when using the non-federated method.

From Figure 3, it is clear that the best performance comes when reusing a trained local model for the same client. While this is promising for cases where the same clients are reusing their devices (say, for personal wearables), for experimentally-focused applications where many participants come in and use the device only for their few trials before handing it off to the next participant, initializing everyone to one of their previous models will not be possible (at least

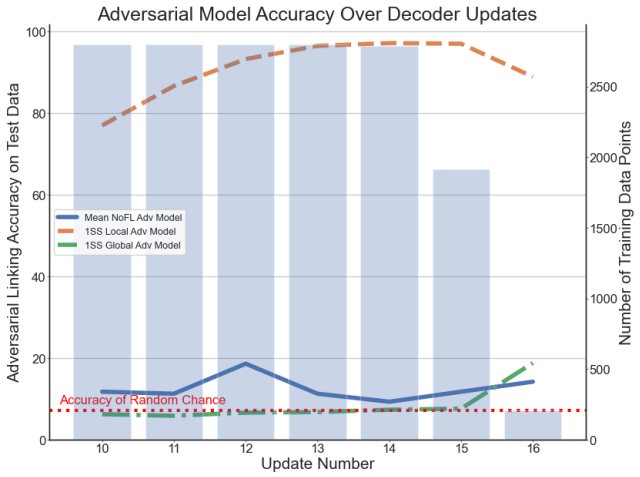

Fig. 9. Adversarial model accuracy for the non-federated (NoFL), federated local, and federated global models.

not on their first run). Thus, we can instead provide the new clients directly with the shared global model trained from previous trials, and this has been show to generate similar results.

B. Privacy Analysis

1) Post-Federated Learning Linkage Attacks: The final privacy question is whether federated learning, without being paired with other privacy mechanisms, is effective or sufficient in protecting user privacy, again measured by linkability from the decoders to participant ID codes. The main risk comes from the global model, as any device can join the network and request a copy of the current global model. Thus, the ideal scenario would be that the adversary can easily link the local updates back to participant ID codes (as they will never have access to this information) but struggle to make any progress given the global models (which they will be able to query). While it is clear from Figure 9 that the global model serves to thwart adversarial linkage models, the local model is still high risk, which indicates at the very least it is necessary to add some form of encryption between the client and the server so adversaries cannot recover the updated local models. Further work should investigate the ability to recover training data or infer membership. Another logical extension to this analysis would be to investigate the effects of many sybils, or alternatively the effects of said sybils attempting to poison the model/data and observing what information they may be able to recover.

C. Limitations

The major limitations stem from the fact that this analysis was conducted in simulations as opposed to real-time deployment for real-world subjects. Specifically, the main limitations are that 1) any effects of co-adaptivity were ignored (starting at update 10 after learning appears to have subsided and only using the second half of all the data collected in each update) and 2) it remains unclear how the chosen cost function and performance metric translate to real-world usability. Namely, once the cost gets arbitrarily close to 0, does going from 1E-4 to a cost of 1E-5 actual result in any meaningful difference, as experienced by the user? It remains unclear if the difference between these low order of magnitude performances can be felt by human operators and if there exists some minimum performance cost below which humans cannot tell a difference. Clearly, the chosen penalty terms affect the cost function but it is unclear how they affect usability.

D. Future Work

The most important future questions are around the effects and magnitude of client heterogeneity present, as well as to run live studies to validate that the cost function does correspond to actual usability. For client heterogeneity, the ideal scenario is that even before training begins, it is possible to cluster clients based on their (estimated) underlying data distributions, potentially implementing anomaly detection for withholding dissimilar clients or some form of clustering for training similar clients together and limited global model divergence. For the live experiments, this will require a number of additions to the current codebase, namely the ability to interface with the EMG sensors (accept wirelessly transmitted data), a transition to fully asynchronous or online ML since each trial will be limited to a single participant at a time, and ensuring that the model (both training and inference) can be run in real-time. For real-time considerations, further work needs to be done in order to determine what the minimization speed is in the federated set up to see if it can run in real-time, and, if so, then the ideal update size (or amount of time to collect data before sending back to the recording device) will likely need to be empirically determined based on how fast the federated simulation can minimize each batch's cost function.

Another necessary extension to this work is a full parametrization study in order to re-calibrate the cost function to adjust for applying PCA and input normalization. Besides the penalty terms within the cost function, how many clients should be used each round, ideal learning rate, and other training constants where the recommended values from previous papers have been used as opposed to being empirically determined on our specific data should all be studied more in depth. Given that there are numerous differences between this dataset and most other papers that choose to focus on the standard MNIST/CIFAR, parameter values from the literature may hold little weight for novel biosignal applications.

From the privacy perspective, a complete product would require some form of encryption for sending biosignal updates: prior works have shown success reverse-engineering inputs based on intercepted weight updates [13] [21]. Popular options for this include secure multi-party computation, homomorphic encryption, or various lightweight, mobile-friendly encryption schemes; even so, many works still

consider an honest-but-curious server, often relying on differential privacy in combination with federated learning and some form of encryption in order to be the most privacy-robust.

A final step would be to refactor the codebase to work from PyTorch, as this is where nearly all state-of-the-art federated learning algorithms are implemented, and would make this work more useful for the community in terms of interoperability as well as validation.

VI. Conclusion

This work has shown that federated learning offers performance benefits in edge device scenarios, and furthermore that the increase in performance does not have to come at a higher risk to user privacy, given sufficient steps are taken to block malicious end users and that the weight updates are not intercept-able. Additionally, it shows that while federated learning is effective in ensuring user privacy, it is not sufficient, as some additional form of defense (e.g. encryption) is necessary in order to protect the high-risk local models.

Acknowledgment

Thanks to Dr.s Momona Yamagami and César Uribe for the guidance, and to Maneeshika Madduri for the data.

References

[1] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," Advances in Neural Information Processing Systems, vol. 33, pp. 3557–3568, 2020.

[2] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," Advances in Neural Information Processing Systems, vol. 30, 2017.

[3] B. Liu, Y. Guo, and X. Chen, "PFA: Privacy-Preserving Federated Adaptation for Effective Model Personalization," in Proceedings of the Web Conference 2021, 2021, pp. 923–934. doi: 10.1145/3442381.3449847.

[4] 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, 2017, pp. 1273–1282.

[5] F. Hamidi, K. Poneres, A. Massey, and A. Hurst, "Who should have access to my pointing data? privacy tradeoffs of adaptive assistive technologies," in Proceedings of the 20th international acm sigaccess conference on computers and accessibility, 2018, pp. 203–216.

[6] J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "The limits of differential privacy (and its misuse in data release and machine learning)," Communications of the ACM, vol. 64, no. 7, pp. 33–35, 2021.

[7] J. M. Abowd, "The US Census Bureau adopts differential privacy," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2867–2867.

[8] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in apple's implementation of differential privacy on macos 10.12," arXiv preprint arXiv:1709.02753, 2017.

[9] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in International Conference on Machine Learning, 2021, pp. 2089–2099.

[10] L. Sweeney, "Simple demographics often identify people uniquely," Health (San Francisco), vol. 671, no. 2000, pp. 1–34, 2000.

[11] L. Sweeney, "k-anonymity: A model for protecting privacy," International journal of uncertainty, fuzziness and knowledge-based systems, vol. 10, no. 5, pp. 557–570, 2002.

[12] M. M. Madduri et al., "Co-Adaptive Myoelectric Interface for Continuous Control," IFAC-PapersOnLine, vol. 55, no. 41, pp. 95–100, 2022.

[13] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," IEEE Security & Privacy, vol. 19, no. 2, pp. 20–28, 2020.

[14] M. T. Toghani, S. Lee, and C. A. Uribe, "Pars-push: Personalized, asynchronous and robust decentralized optimization," IEEE Control Systems Letters, vol. 7, pp. 361–366, 2022.

[15] M. T. Toghani, S. Lee, and C. A. Uribe, "PersA-FL: Personalized Asynchronous Federated Learning," arXiv preprint arXiv:2210.01176, 2022.

[16] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," arXiv preprint arXiv:1812.00564, 2018.

[17] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," IEEE Internet of Things Journal, vol. 7, no. 10, pp. 9530–9539, 2020.

[18] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, 2014, pp. 1054–1067.

[19] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," ACM Computing Surveys (CSUR), vol. 54, no. 6, pp. 1–36, 2021.

[20] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," arXiv preprint arXiv:2003.13461, 2020.

[21] Y. Wang, C. Si, and X. Wu, "Regression Model Fitting under Differential Privacy and Model Inversion Attack.," in IJCAI, 2015, pp. 1003–1009.

[22] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," Proceedings of the IEEE, vol. 104, no. 9, pp. 1727–1765, 2016.