Privacy-Preserving and Robust Federated Deep Metric Learning

Yulong Tian*, Xiaopeng Ke*, Zeyi Tao†, Shaohua Ding*, Fengyuan Xu*, Qun Li†, Hao Han‡, Sheng Zhong*, Xinyi Fu§
*State Key Laboratory for Novel Software Technology, Nanjing University, China
†Department of Computer Science, College of William & Mary, USA
‡College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
§Ant Group, China

***Abstract*—Federated learning, in contrast to traditional learning paradigms, has demonstrated its unique advantages in providing intelligence at the edge. However, existing federated learning approaches focus on the end-to-end classification tasks requiring a simple collaboration procedure where each participant can perform its local training independently. Unfortunately, there are still many tasks relying on learning the distinguishable feature metrics with respect to all the data, which is a different collaboration procedure across training participants. For example, the model for people identification has to ensure the feature representing a person is dissimilar to those representing others. To enable such federated learning for deep metrics (a.k.a federated deep metric learning) is challenging due to the data privacy and procedure robustness issues. With the consideration of these two challenges, this work proposes a novel computing framework for federated deep metric learning. This framework leverages the system-algorithm co-design to address privacy concerns via the Trusted Execution Environment (SGX enclave) and Differential Privacy mechanism. It also introduces a large-scale federated protocol which can robustly and efficiently deal with practical factors like the network fluctuation. We implement and evaluate our computing framework with two settings. One is a real-world implementation with a large number of mobile devices, while the other one is in our controllable environment for conducting experiments in various tasks. Our evaluation results show that our computing framework is able to train federated deep metric learning models with excellent scalability, data privacy preserving, and considerable accuracy even in exception conditions.**

I. Introduction

Federated learning (FL) is a powerful collaborative learning framework of deep learning (DL) models for distributed data sources like end devices [1], [2]. The key feature of FL is to keep original learning data on the device possessing them during the whole training procedure. Upon this paradigm, many privacy computing techniques [3]–[5] have been proposed to further strengthen the protection of data on participant devices. FL is naturally suitable to the deep learning in mobile scenarios and enables many popular applications [6]–[9].

Existing FL frameworks mainly focus on one collaborative learning paradigm. This paradigm enforces each device to only receive aggregated model information, horizontally isolating these participants. It is fine when FL frameworks are used to train traditional DL classification models. However, along with the rapid development of the deep learning technology, newly emerged model categories are not supported in existing frameworks due to different collaborative learning paradigm.

One representative model category of such newly emerged ones is the deep metric learning (DML) [10]–[12]. DML leverages deep neural networks to extract high-quality end-to-end feature embeddings from input data. The extracted embeddings from the same data class have smaller distances, while those from different data classes have larger distances. This property makes DML increasingly popular as the data representation foundation for many other DL tasks like the user authentication, clustering, and so on [10], [13]–[17]. Given that usually the data of these DL tasks are distributed among many sources, supporting DML in FL frameworks (FDML) is important to comply with privacy regulations like GDPR [18] and applications of both DML and FL.

Unfortunately, existing FL frameworks cannot well guarantee the privacy and robustness of the new collaborative learning paradigm in FDML. This is because, in FL scenarios, the DML model training requires extra information exchanges across collaboration participants at the individual sample level. For example, when a group of smartphones conduct a FDML training for a model to distinguish phone owners, each smartphone needs "contrastive" user behavior samples from others to improve the distinguishability. Such extra interactions among participants break down the horizontal isolation leveraged by existing FL frameworks, i.e. only aggregated model updates are exchanged, leading to potential privacy leaking and communication difficulty.

In this work, we therefore propose an architecture design for the federated deep metric learning at scale. Our FDML addresses design challenges of the data privacy and communication robustness in massive training collaboration of DML models. It can preserve in a system-algorithm co-design manner the privacy of exchanged data for both similarity measure and weight update. It also can handle unexpected participant offline or long interaction responses in a large participant setting. Our FDML extends current collaborative learning paradigms with practical considerations and theoret-

Fengyuan Xu is the corresponding author.

978-1-6654-6824-4/22/$31.00 © 2022 IEEE

ical guarantee.

With our FDML, a DML model is split into two parts, the embedding network and the loss component. The former part is deployed on each participant, while the latter part is placed on the central server used in the FL scenario. Participants are directly connected to the central server in the same way as in existing FL frameworks, but the communication protocol is different. In the model forward computation, participants send embeddings (data features) to the server; in the model backward propagation, partial derivatives of the network loss w.r.t. the embeddings are sent to participants. Moreover, to protect the data privacy for participants, our FDML migrates sensitive server-side computations into the Trust Execution Environment (TEE) in case the server controller is malicious. It also introduces a differential privacy algorithm to protect participants' sensitive information from the participants who may be malicious. Additionally, we also analyze the factors that influence the robustness of our FDML and propose countermeasures with theoretical convergence analysis.

Our design has been evaluated on commercial deep metric learning tasks inside Alipay[1], with a large scale number of volunteer employees' smartphones as participants. We also build a research emulation platform and test FDML applications on different datasets, including a real-world screen-touch dataset collected from 1517 volunteer employees from Alipay. The experimental results show that our design is practical and can be robust in extreme conditions.

To summarize, our main contributions are:

- To the best of our knowledge, we are the first to investigate FDML and provide a practical, privacy-preserving, and robust system design for large-scale participants.
- We propose a system-algorithm co-design to protect participants' privacy. We design TEE based isolation and differentially private partial derivative releasing mechanism to deal with the privacy leakage from participants to cloud and cloud to participants, respectively.
- We consider practical factors like network fluctuation and propose designs for ensuring training procedure robustness. We study the impact of possible exceptions in FDML, propose a robust training design that works even when lots of participants are offline, and also provide theoretical convergence proof.

The rest of this paper is organized as follows. We first introduce some background knowledge and the related work in Section II. Then we state our problem in Section III and present the architecture design overview in Section IV. We present the privacy protection design in Section V and study the system robustness in Section VI. Furthermore, we describe the experiment platforms & applications and results of experiments in Section VII and Section VIII, respectively. Finally, we conclude this paper (Section IX).

[1]https://www.alipay.com/

II. Background and Related Work

In this section, we first introduce some background knowledge and related works for DML and FL in Section II-A and Section II-B, respectively. Then, we present some popular general-purpose privacy protection methods and describes existing privacy-preserving FL designs in Section II-C.

A. Deep Metric Learning (DML)

DML leverages deep neural networks to learn the similarity between objects. DML usually consists of a neural network for embedding extraction (we refer to it as embedding network) and a loss component for model training. The embedding network can be any neural network (like CNN, RNN, and so on) and generates embeddings for inputs. The loss component is the core part of DML. It encourages the embeddings of inputs that belong to different categories to have larger distances, but that belong to the same category to have smaller distances. The distances here are usually the euclidean distances or the cosine similarity. We take the triplet loss [10] which is a popular DML loss as an example. First, we need to build embedding triplets in the form of $< A, P, N >$, where A is the anchor embedding, P means positive and is in the same category as A, and N means negative and is in a category other than A. Then, we can calculate the loss using

$$loss = \frac{1}{k} \sum \max\left(||A - P|| - ||A - N|| + m,\ 0\right), \qquad (1)$$

where k is the number of triplets, $||A - P||$ is the distance between the two embedding of the same category (intra-class distance), $||A - N||$ is the distance between the two embedding of different categories (inter-class distance), and m is a hyper-parameter. The loss means the inter-class distances should be larger than the intra-class distances at least by m.

Since DML models can extract high-quality embeddings even for the data categories that are not in the training dataset, they are widely used in many real-world critical scenarios [10], [13]–[17].

B. Federated Learning (FL)

FL shows a new training paradigm where end devices and a central cloud train deep learning models in a privacy-friendly manner [1], [19]–[21]. In a traditional FL training round, end devices first download a global model from the cloud. Then each participant uses its training data to perform local training using the downloaded model as the starting point. Next, the participants send the updated model to the cloud. Finally, the cloud aggregates the received local models and replaces the global model with the newly aggregated one. It is worth noting that the participants in traditional FL should be able to conduct the local training independently. For DML, this assumption is not practical because the loss of DML may need to use data from different participants (details in Section III), so we need to design new federated training methods for DML.

FL has shown its success in many applications like next word prediction for smartphone keyboard [19], input query suggestion for mobile phone [20], product recommendation

[21], healthcare [7], and so on. Apart from training classical neural networks for classification, FL also has been used for other learning tasks. Liu et al. [22] introduce the design of federated transfer learning. Zhuo et al. [23] propose federated reinforcement learning. Cheng et al. [24] show a federated boosting algorithm.

There are some variants of traditional FL. Vepakomma et al. [7] introduce distributed split learning for classification tasks. In their design, the cloud will take part in the model forward computing and back propagation, but not just aggregates the uploaded local models as the traditional FL does. Vertical FL [25] is also different from traditional FL. In vertical FL, the features of a single training sample are held by different participants, thus the cloud also needs to be deeply involved in the training details. In this paper, the cloud will also undertake more computing tasks in our design.

C. Privacy Protection Methods

1) Differential Privacy: Differential Privacy [26] provides a theoretical metric for evaluating the privacy protection level. Definition II.1 shows the formal definition of differential privacy.

Definition II.1. Differential Privacy [26]. A randomized mechanism $\mathcal{M}$ is ϵ-differential private, if for all $S \subseteq Range(\mathcal{M})$ and all adjacent inputs D and $\hat{D}$, $\Pr[\mathcal{M}(\mathrm{D}) \in \mathrm{S}] \leq \exp(\epsilon)\Pr[\mathcal{M}(\hat{\mathrm{D}}) \in \mathrm{S}]$ holds ($\epsilon > 0$).

The adjacent D and $\hat{D}$ only differ by one item. ϵ is the privacy budget. The smaller the ϵ, the better the privacy protection level. Laplace mechanism is one of the mechanisms that satisfy definition II.1 and is shown in Definition II.2. It satisfies ϵ-differential privacy following Lemma II.1.

Definition II.2. The Laplace Mechanism [26]. Given any bounded function f, the Laplace mechanism is defined as $\mathcal{M} = f(D) + (Y_i, .., Y_k)$, where $D \in Domain(f)$ and Y_i are i.i.d. variables sampled from $Lap(b)$.

Lemma II.1. *Let* $\Delta f = \max_{||D-\hat{D}||_1=1} (||f(D) - f(\hat{D}||)$ *be the sensitivity of* f *on any adjacent inputs, the Laplace mechanism is differentially private,* $\epsilon = \Delta f/b$ *[26].*

2) Trusted Execution Environment (TEE): TEE is a hardware-enabled secure area in the central processor. It preserves the confidentiality and integrity of the data and code inside it. A process can securely run in the TEE without being peeked at or tampered with by the adversary outside. Attestation mechanisms are usually also provided by the TEE for users to verify whether their programs are running in a secure TEE. Intel SGX Enclave [27] is a kind of TEE and is integrated into the commercial CPU produced by Intel.

3) Privacy Protection Methods for Traditional FL: To solve the data privacy issues in FL, researchers propose many approaches. Hao et al. [28] combine additively homomorphic encryption with differential privacy to protect participants' data privacy. For the same purpose, Bonawitz et al. [29] propose secure aggregation preventing the cloud from seeing the uploaded data, and Passerat et al. [30] describe a design of placing the cloud aggregator in the TEE. In addition, frameworks like PySyft [31] and tf-encrypted [32] provide easy-to-use APIs for implementing FL with privacy protection methods.

III. Problem Statement

We consider the scenario of cloud-aided collaborative learning of numerous participants. The participants, who use mobile devices like smartphones, belong to different parties and want to train DML models while keeping their training data private. We focus on the scenario where a single participant cannot train a satisfactory model without referring to the data of others. This scenario is common because the training data of one participant is usually either too little or from too few categories to train a DML model. For example, in the scenario of user authentication for smartphones, one smartphone usually only has the data of its owner.

The DML training in such scenarios requires a suitable data exchange mechanism to enable cross-participant feature comparisons, which traditional FL cannot provide. Traditional FL designs assume that participants can perform the local training by only using their own data (Recall Section II-B) and do not need data exchanges for loss calculation. However, the DML loss requires data exchanges across different participants to perform feature comparisons. Therefore, we need new enhanced FL designs that can provide the desired data exchanges for DML.

The additional data exchanges in the new framework will inevitably increase the risk of privacy leakage. For example, a malicious participant might try to extract private information from the content obtained in the data exchanges. And the cloud that coordinates the training might also be able to exploit those exchanges and obtain participants' privacy. Thus, we need to carefully examine possible privacy leakages and solve the challenging problem that requires minimizing the leakage while keeping the system efficient and usable. In this paper, we assume the cloud controller and any participant could be the attacker and assume that the hardware infrastructure including the TEE is secure.

Other practical factors like procedure robustness also need to be considered in the design of the new framework. In such distributed systems, common exceptions like network fluctuation and the instability of end devices will result in unexpected participant offline. If the offline happens when the data exchanges are not finished, the model training may be affected. Thus, we need to analyze the impact of the unexpected offline on the model accuracy and design practical methods to make the training procedure robust.

IV. FDML Framework Overview

As discussed in Section III, the end devices cannot handle the loss calculation when following the existing FL design. Hence, we plan to leverage the cloud server to undertake the main work of the loss calculation.

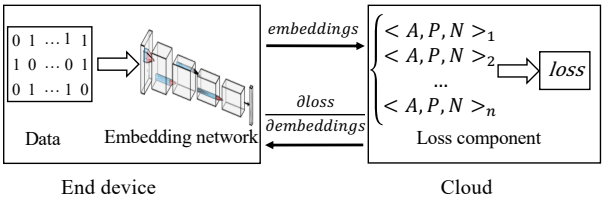

Fig. 1. The DML Model Split Scheme. In the model forward execution and backward propagation, the end device is responsible for the computation related to the embedding network, and the cloud processes the loss component.

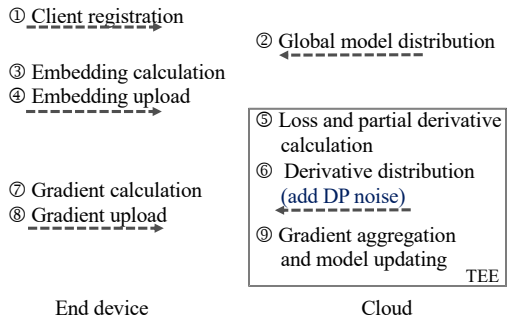

Fig. 2. The Communications between End Devices and the Cloud in One Training Round. At first, the end devices who are able and willing to participate in the training will make registrations (Step 1). Then, the cloud distributes the global embedding network to some selected registered end devices (Step 2). Next, the end devices use the received global embedding network to extract embeddings from their training data (Step 3) and send those embeddings to the cloud (Step 4). The cloud will use those uploaded embeddings to calculate the loss and the partial derivatives w.r.t. those embeddings $\{\frac{\partial loss}{\partial emb_i}\}$ (Step 5). $\frac{\partial loss}{\partial emb_i}$ will be sent back to the end device that uploaded emb_i (Step 6). After that, each end device computes its local gradients (Step 7) and uploads those gradients to the cloud (Step 8). Finally, the cloud aggregates the gradients uploaded by the end devices as the global gradients and then use the global gradients to update the global embedding network (Step 9). The noise adding and the TEE part are privacy-preserving designs, and they will be covered in Section V.

As shown in Figure 1, inspired by the split learning [7] (which is not designed for DML, we choose to split the DML model into two parts (the embedding network and the loss component) and place them in the cloud server and end devices, respectively. The two parts share necessary information by sending messages to each other during the model training. In the model forward computation, the end devices will extract embeddings from their training data leveraging the embedding network, and send the extracted embeddings to the cloud server. Then the cloud server will calculate the network loss based on the received embeddings. As to the backward propagation, we leverage the property of the chain rule and design to send necessary partial derivatives from the cloud-server to the end devices so that the end devices can calculate the gradients. We design a protocol based on the above split scheme. Figure 2 shows how the protocol works in a training round.

The major differences between our FDML design and the traditional FL design are: first, the loss calculation of FDML requires the data from different devices and is conducted on the cloud side, while the loss computing of traditional FL does not involve different devices and is performed at the end-device side; second, in traditional FL, the end devices only need to upload the local model (local gradient), but in our FDML design, extra data exchanges are needed for computing the loss and the gradient.

Client registration (Step 1) and participation strategy. The client registration can be asynchronous with the model training, and end devices do not need to make registration at each training round. A participant's registration is considered to be valid within a predefined time range.

As to the participation strategy, since the end devices can be battery-based devices using wireless networks, the participants could be sensitive to battery and data usage or system fluency. For this case, we can reuse the strategy mentioned in [33] that an end device could participate in the training only when the end device is charging, connected to WIFI, and idle. Participants will suspend their participation once their devices are not in the condition mentioned above.

Gradient aggregation and model updating (Step 9). We define the gradient aggregation of a training round as

$$G = \sum_{i}^{k} \frac{|d_i|}{|D|} \frac{\partial loss}{\partial parameters}_{p_i}, \tag{2}$$

where k is the number of the participants in the training round, $|d_i|$ denotes the number of training sample of participant p_i, $|D| = \sum |d_i|$, and $\frac{\partial loss}{\partial parameters}_{p_i}$ represents the local gradient of participant p_i.

V. Privacy Protection

The communications in the proposed protocol may leak the participants' private information if no privacy protection measures are taken. Specifically, the cloud might recover private information from the data embeddings and local gradients uploaded by the participants [34], and a malicious participant also has the chance to obtain the private information of other participants by analyzing the global model and the partial derivatives distributed by the cloud [35], [36]. Hence, we need to design methods to reduce the possible private information leakage from the communications. For the privacy leakage from the participants to the cloud, we propose a TEE-based method in Section V-A. And for the information leakage from the cloud to participants, we will show our differential-privacy-based method in Section V-B.

A. Protect the Content Uploaded by Participants with TEE

Since the cloud-side loss computation is complicated, privacy protection methods, like MPC and homomorphic encryption, are ill-suitable. Moreover, TEE shows excellent performance and is free from utility degradation, so we plan to leverage TEE to protect the data uploaded by the participants.

Although TEE shows many advantages compared to other privacy protection methods, in our design, we do not plan to perform the entire DML training inside TEE. This is because that choice requires implementing a DL framework inside TEE which would significantly increase the Trusted Computing

Base (TCB) of the TEE. Instead, we choose to only put the loss computing & partial derivatives calculation (Step 5) and gradient aggregation (Step 9) of the proposed protocol inside TEE. This design is lightweight and does not require implementing complicated forward and backward model computing. And the core part of the TEE-side implementation only requires less than 300 lines of code in C++ language.

The participants will attest the integrity of the TEE at the beginning and then directly send their embeddings and local gradients into the TEE during the model training. And the partial derivatives will be directly sent back to the participants from the TEE. Using this design, the cloud controller cannot access the participants' embeddings and local gradients and thus preventing the cloud controller from launching attacks. In this paper, we choose the Intel SGX enclave as the TEE because of its good availability and compatibility.

Since TEEs usually have limited computing resources and do not support GPU, they could be a performance bottleneck. To address this issue, we design optimized solutions for the computationally intensive loss and partial derivative calculation which is conducted inside TEEs. Our basic idea is to derive the mathematical expression of the partial derivatives directly, instead of introducing automatic gradient calculation tools. This method minimizes memory usage, avoids introducing automatic partial derivation, and reduces the amount of code in TEE and the difficulty of the implementation. We use triplet loss as an example to show the design in Algorithm 1.

Algorithm 1: Partial Derivatives Calculation inside TEE

Inputs : Embeddings from different participants: $D_{emb} = \{emb_i\}$; labels of the embeddings: $\{label_i\}$
Output: Partial derivatives: $\{\frac{\partial loss}{\partial emb_i}\}$

1 Calculate the Euclidean distances between any two embeddings in D_{emb}, and only save the upper triangle of the distance matrix;
2 Allocate memory for each $\frac{\partial loss}{\partial emb_i}$, and initialize them with 0;
3 $counter \leftarrow 0$;
4 **while** *build up an embedding triplet* $< A, P, N >$ **do**
5 $\quad \frac{\partial loss}{\partial emb_A} \leftarrow \frac{\partial loss}{\partial emb_A} + \frac{emb_A - emb_P}{||emb_A - emb_P||} - \frac{emb_A - emb_N}{||emb_A - emb_N||}$;
6 $\quad \frac{\partial loss}{\partial emb_P} \leftarrow \frac{\partial loss}{\partial emb_P} + \frac{emb_P - emb_A}{||emb_A - emb_P||}$;
7 $\quad \frac{\partial loss}{\partial emb_N} \leftarrow \frac{\partial loss}{\partial emb_N} + \frac{emb_A - emb_N}{||emb_A - emb_N||}$;
8 $\quad counter \leftarrow counter + 1$;
9 **foreach** $\frac{\partial loss}{\partial emb_i}$ **do**
10 $\quad \frac{\partial loss}{\partial emb_i} \leftarrow \frac{\partial loss}{\partial emb_i} \frac{1}{counter}$

B. Differentially Private Partial Derivative Releasing

We cannot simply put the participant-side calculation into participant-side TEEs because there are many participants and not every participant's device supports TEE. Furthermore, participants' devices usually have limited computing resources, thus using MPC or homomorphic encryption is also infeasible. Hence, we choose to design a differentially private mechanism to reduce the private information leakage from the cloud to the (malicious) participants.

We clip the partial derivatives and add Laplace noise on them before sending them to the participants. To further protect privacy, we randomly drop some embeddings before calculating the loss. Hence even when the attacker knows that some embeddings are uploaded in one training round, the advantage of this background knowledge will be reduced because the attacker will never know which uploaded embeddings are actually used. Algorithm 2 shows our method, which adjusts the Step 5 of the proposed protocol (Figure 2). Inspired by [37], we prove that the algorithm is differentially private and gives a privacy bound ($\epsilon = ln[(1-r)e^{2T/b} + r]$) in Theorem V.1.

Algorithm 2: Differentially Private Training – Cloud side loss and partial derivatives calculation

Inputs : Embeddings from different participants: $D_{emb} = \{emb_i\}$; loss fuction: f_{loss}; embedding drop rate: r; bound: $T > 0$; noise scale: b
Output: Processed partial derivatives: $\{\frac{\partial loss}{\partial emb_i}\}$

1 Randomly choose some embeddings with a rate of r, and remove those embeddings from D_{emb};
2 Perform loss calculation based on the new D_{emb}, and calculate $\frac{\partial loss}{\partial emb}$ for each embedding;
3 Clip each $\frac{\partial loss}{\partial emb}$ using $-T$ as the lower bound and T as the upper bound;
4 Update each $\frac{\partial loss}{\partial emb}$ by adding $Lap(b)$;

Theorem V.1. *A FDML training round following Algorithm 2 is differentially private, i.e.* $\epsilon = ln[(1-r)e^{2T/b} + r]$.

Proof. **(a)** Let $D = \{data_i\}$ be the set of the training samples from different participants in one training round and $F_{derivate}$ represent the computing logic of Algorithm 2. For any two adjacent training dataset D_a and D_b, we have $||F_{derivate}\Big|_{D_a} - F_{derivate}\Big|_{D_b}|| \leq 2T$.

According to Lemma II.1, we have

$$P[F_{derivate}\Big|_{D_a} = S] \leq e^{2T/b} P[F_{derivate}\Big|_{D_b} = S]. \quad (3)$$

(b) Let F_{emb} be the embedding network. The embeddings uploaded by the participants can be written as $Emb = \{F_{emb}(data_i)\}$. Randomly dropping the embeddings in Emb is the same as dropping the corresponding training sample in D in the sense of probability.

Suppose there are two adjacent training dataset D_x and D_y that differ by one training sample $data_k$, there are two cases for the random embedding drop:

(1) The embedding extracted from $data_k$ is dropped, $P[F_{derivate}\Big|_{D_x} = S] = P[F_{derivate}\Big|_{D_y} = S]$.

(2) The embedding extracted from $data_k$ is not dropped. According to Equation 3, we have $P[F_{derivate}\Big|_{D_x} = S] \leq e^{2T/b} P[F_{derivate}\Big|_{D_y} = S]$.

Combine above two cases:

$$P[F_{derivate}\Big|_{D_x} = S]$$
$$\leq rP[F_{derivate}\Big|_{D_y} = S] + (1-r)e^{2T/b}P[F_{derivate}\Big|_{D_y} = S]$$
$$= e^{ln([r+(1-r)e^{2T/b}])}P[F_{derivate}\Big|_{D_y} = S].$$

Thus, $\epsilon = ln[r + (1-r)e^{2T/b}]$.

□

VI. Training Procedure Robustness

In this section, we consider the unexpected situations in such distributed systems. In Section VI-A, we give a design that aims to make the training procedure robust to unexpected participant dropping, and in Section VI-B, we provide the convergence analysis for the proposed robustness design.

A. Strategy for Training Procedure Robustness

Since the cloud-server is more controllable than the end devices, we focus on the exceptions related to mass participants' devices and assume the cloud-server will always perform its work as expected.

The participants' devices may not perform their work on schedule. The network fluctuation may affect the communication between the cloud-server and the end devices which are probably using wireless networks. Furthermore, the end devices could suspend the participation due to issues like battery usage, data plan, or system load. These exceptions may obstruct the expected communications: (Case A) participants who have received the global embeddings network cannot upload their embeddings; (Case B) participants who have already successfully committed their embeddings cannot provide their local gradients to the cloud.

We design Algorithm 3 to cope with the above mentioned cases. For Case A, when distributing the global embedding network, the cloud-server can select more participants in case some of them are offline. When Case B occurs, the gradient collected by the cloud is not intact. The cloud-server has two choices: (1) give up the current round and start a new round; (2) continue the training with the local gradients collected from the remaining participants. The second choice could be better. For one thing, the local gradients of the remanent participants still contain plenty of information for model training. The local gradients of one participant contain not only its information but also other participants' information, which provides information redundancy (see the equations in Algorithm 1). For another, the strategy of continuing training also saves overall training time, because waiting for an exception-free training round is difficult when there are large-scale participants, even the exception rate is low.

B. Convergence Analysis

In this section, we provide the convergence analysis of FDML with incomplete gradient information in Algorithm 3. We show that the FDML achieves $O(1/T)$ convergence rate using the stochastic gradient descent (SGD) for training when training iteration T is sufficiently large.

Algorithm 3: System Robustness Design – Cloud Side

Inputs: Participant number: $num_{required}$; expected participant response rate: $rate_p$; multiplier: $k \geq 1$

```
Init: num_p1 ← 0, num_p2 ← 0;
Distribute global embedding network to num_required * k
  participants ;                                   // Step 2
while num_p1 < num_required do
    if waiting time is up then
        Reset timer and go to line 2;
    Collect participants' embeddings and increase participant
      counter num_p1;
Calculate loss and partial derivatives ;           // Step 5
Distribute partial derivatives ;                   // Step 6
while num_p2 < num_required * rate_p do
    Collect local gradients and increase participant counter num_p2;
    if waiting time is up then
        break;
Aggregate gradients and update the model ;         // Step 9
```

The objective function can be defined as:

$$\min_{x \in \mathcal{F}} f(x) \coloneqq E_{\xi \sim \mathcal{D}_i}[F(x;\xi)] \tag{4}$$

where $F(\cdot;\cdot)$ is the predefined loss function, x is model parameters from some feasible parameter space $\mathcal{F}$. The $\mathcal{D}_i$ is a dataset on the i^{th} participant. ξ randomly samples from $\mathcal{D}_i$ for each training iteration.

We use the following assumptions for the non-convex optimization problem.

Assumption VI.1. *(Smoothness) We assume the Lipschitz continuous for all model copies $\nabla f_i(x)$, i.e., $||\nabla f_i(x) - \nabla f_i(y)|| \leq L||x-y||$ for all x, y and $i \in [n]$.*

Assumption VI.2. *(Unbiasedness) The stochastic gradients $\nabla F_i(x)$ is unbiased estimator of objective function, i.e., $E_{\xi \sim \mathcal{D}_i}[\nabla \mathcal{F}(\boldsymbol{x};\xi)] = \nabla f_i(x)$.*

Lemma VI.1. *(Lipschitz smoothness) Suppose model copy $f_i(x)$ in Assumption. VI.1 holds is L-Lipschitz continuous and suppose the model $f_i(x)$ is strongly-convex. For any $x, y \in R^k$, the objective function $f(x)$ satisfies $f(x) - f(y) - \nabla f(y)^\top (x-y) \leq \frac{L}{2}||x-y||^2$.*

The proof of Lemma VI.1 can be found in the previous work [38]. These assumptions have been widely used in the ML community for analyzing the ML model convergence, particularly for large-scaled distributed machine learning. In addition, we provide the following auxiliary inequality, which are used in our proof.

$$\text{Inequality 1: } ||\sum_{i=1}^{n}(x_i - y_i)||_2^2 \leq n\sum_{i=1}^{n}||x_i - y_i||_2^2 \tag{5}$$

We also define the collection matrix $\mathcal{C}_t \in R^{n \times n}$ as diagonal matrix where only $k(k \leq n)$ diagonal entries are 1 and the rest entries are 0. We assume that the diagonal entry $c_{i,t}$ (the i^{th} collected participant at iteration t) has probability $p(c_{i,t} = 1|\text{Exceptions}_t) = \rho_t$ for all participants in t round. Now, we have the following convergence guarantee for FDML.

Theorem VI.2. *(FDML Convergence Analysis) We assume that the optimal feasible set $\mathcal{F}$ of our objective function $f(x)$ is nonempty and bounded. And $f(x_t)$ has bounded gradients $\mathbb{E}\|\nabla f(x)\| \leq G$ and $\mathbb{E}\|\nabla f(x)\|_2^2 \leq \sigma^2$. Let the sequence of learning rate $\{\gamma_t\}$ is $\sum_{t=1}^{\infty}\gamma_t^2 < \infty$ and $\sum_{t=1}^{\infty}\gamma_t = \infty$. Under the assumptions above, Federated deep metric learning converges to a stationary point in expectation as:*

$$\frac{1}{T}\Big(\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x_t)\|_2^2\Big) \leq \frac{2(f(x_0)-f(x^*))}{\gamma T} + \frac{(\gamma+\gamma^2 L)\sigma^2}{2}\sum_{t=0}^{T-1}(1-\rho_t)$$

Proof. According to the Federated deep metric learning present in Equation 5, the gradient descent updating can be expressed as

$$x_{t+1} = x_t - \gamma_t(\mathcal{C}_t\partial\mathcal{F}(x_t;\xi_t))^\top\frac{\mathbf{1}_n}{n} \tag{6}$$

Here, $x_t \in R^d$ are the model parameters on each clients, $\mathcal{C}_t \in R^{n\times n}$ is the collection matrix representing collected gradients as defined in the Federated deep metric learning algorithm, and $\partial\mathcal{F}(x_t,\xi_j) = [\nabla F_1(x_{t,1};\xi_{t,1}), \nabla F_2(x_{t,2};\xi_{t,2}), \cdots, \nabla F_n(x_{t,n};\xi_{t,n})]^\top$ are gradients. According to Equation 6, we know

$$\mathbb{E}[f(x_{t+1})|\mathcal{C}_t,\mathcal{D}_t] = \mathbb{E}[f(x_t - \gamma_t(\mathcal{C}_t\partial\mathcal{F}(x_t;\xi_t))^\top\frac{\mathbf{1}_n}{n})] \tag{7}$$

Then, we apply the above equation to the Lipschizan gradient assumption and obtain

$$\mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(x_t)] - \underbrace{\mathbb{E}[\langle\nabla f(x_t), \gamma_t(\mathcal{C}_t\partial\mathcal{F}(x_t;\xi_t))^\top\frac{\mathbf{1}_n}{n}\rangle]}_{T_1} + \frac{L}{2}\underbrace{\mathbb{E}\left\|\gamma_t(\mathcal{C}_t\partial\mathcal{F}(x_t;\xi_t))^\top\frac{\mathbf{1}_n}{n}\right\|_2^2}_{T_2} \tag{8}$$

For T_1, we have:

$$\begin{aligned}
T_1 &= \mathbb{E}[\langle\nabla f(x_t), \gamma_t(\mathcal{C}_t\partial\mathcal{F}(x_t;\xi_t))^\top\frac{\mathbf{1}_n}{n}\rangle] \\
&= \gamma_t\mathbb{E}[\langle\nabla f(x_t), (\mathcal{C}_t\partial f(x_t))^\top\frac{\mathbf{1}_n}{n}\rangle] \quad \textit{(Assumption VI.2)} \\
&= \frac{\gamma_t}{2}\mathbb{E}\|\nabla f(x_t)\|_2^2 + \frac{\gamma_t}{2}\mathbb{E}\left\|(\mathcal{C}_t\partial f(x_t))^\top\frac{\mathbf{1}_n}{n}\right\|_2^2 \\
&\quad - \frac{\gamma_t}{2}\underbrace{\mathbb{E}\left\|\nabla f(x_t) - (\mathcal{C}_t\partial f(x_t))^\top\frac{\mathbf{1}_n}{n}\right\|_2^2}_{T_3}
\end{aligned}$$

And for T_3,

$$\begin{aligned}
T_3 &= \mathbb{E}\left\|\nabla f(x_t) - \mathcal{C}_t\partial f(x_t)\frac{\mathbf{1}_n}{n}\right\|_2^2 \\
&= \mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^n\nabla f_i(x_t) - \frac{1}{n}\sum_{i=1}^n\mathbf{I}_{c_i}\nabla f_i(x_t)\right\|_2^2 \\
&\leq \frac{1}{n}\sum_{i=1}^n\mathbb{E}\|\nabla f_i(x_t) - \mathbf{I}_{c_i}\nabla f_i(x_t)\|_2^2 \\
&= \frac{\sigma^2}{n}\sum_{i=1}^n\mathbb{E}[(\mathbf{I}-\mathbf{I}_{c_i})^2]
\end{aligned}$$

where $\mathbf{I}_{c_i} = 1$ if entry $c_{i,t} = 1$ for $i \in [k]$. Now we have:

$$\mathbb{E}[(\mathbf{I}-\mathbf{I}_{c_i})^2] = (1-\rho_{i,t})\cdot 1^2 = (1-\rho_t)$$

Then $T_3 \leq \sigma^2(1-\rho_t)$. Similarly we have (the proof is omitted due to the page limit)

$$T_2 \leq \frac{\gamma_t^2\sigma^2}{b}\sum_{s=1}^b(1-\rho_t) + \gamma_t^2\mathbb{E}\left\|\partial f(x_t)\frac{\mathbf{1}_n}{n}\right\|_2^2$$

After applying two inequalities T_1 and T_2 to Equation 8, we have

$$\begin{aligned}
\mathbb{E}[f(x_{t+1})] &\leq \mathbb{E}[f(x_t)] - \frac{\gamma_t}{2}\mathbb{E}\|\nabla f(x_t)\|_2^2 \\
&\quad - \frac{\gamma_t}{2}\mathbb{E}\left\|(\mathcal{C}_t\partial f(x_t))^\top\frac{\mathbf{1}_n}{n}\right\|_2^2 + \frac{\gamma_t\sigma^2}{2}(1-\rho_t) \\
&\quad + \frac{\gamma_t^2\sigma^2 L}{2b}\sum_{s=1}^b(1-\rho_t) + \frac{\gamma_t^2 L}{2}\mathbb{E}\left\|\partial f(x_t)\frac{\mathbf{1}_n}{n}\right\|_2^2
\end{aligned}$$

Suppose $\frac{\gamma_t}{2} \leq \frac{\gamma_t^2 L}{2}$, we then have,

$$\begin{aligned}
&\mathbb{E}[f(x_{t+1})] \\
&\leq \mathbb{E}[f(x_t)] - \frac{\gamma_t}{2}\mathbb{E}\|\nabla f(x_t)\|_2^2 + \frac{1}{2}(\gamma_t+\gamma_t^2 L)\sigma^2(1-\rho_t).
\end{aligned}$$

Summing up from $t=0$ to $t=T$, we obtain

$$\begin{aligned}
\mathbb{E}[f(x_T)] &\leq \mathbb{E}[f(x_0)] - \frac{1}{2}\sum_{t=0}^{T-1}\gamma_t\mathbb{E}\|\nabla f(x_t)\|_2^2 \\
&\quad + \frac{1}{2}\sum_{t=0}^{T-1}(\gamma_t+\gamma_t^2 L)\sigma^2(1-\rho_t)
\end{aligned}$$

If we let $\gamma_t = \gamma$ for all $t \in T$, we have

$$\frac{1}{T}\Big(\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(x_t)\|_2^2\Big) \leq \frac{2(f(x_0)-f(x^*))}{\gamma T} + \frac{(\gamma+\gamma^2 L)\sigma^2}{2}\sum_{t=0}^{T-1}(1-\rho_t)$$

□

We see, Algorithm 3 can guarantee the convergence of FDML and it achieves the $O(1/T)$ convergence rate when T is large enough.

VII. Platforms and Applications

A. Evaluation Platforms

We have two evaluation platforms for our FDML design. The first one is a real-world implementation inside Alipay, while the other one is a research platform described below. FDML models can be well trained on both two platforms. However, due to business interest concerns, in this paper, the details of the first platform are omitted. We will only show the results of the experiments conducted on the second platform. And the applications and the results of the research platform described in this paper do not represent those of the commercial implementation inside Alipay.

Research platform. We build an emulation platform according to our architecture design to perform FDML training. The platform uses PyTorch as the deep learning engine and supports a large scale number of virtual participants. Triplet metric learning loss is used as the loss function of all the FDML training experiments, and we use the semi-hard triplet loss implementation in [39]. The platform is deployed on a server equipped with an i7-6850K CPU, 128GB memory, and 4 NVIDIA TitanXp GPU cards. In this paper, our FDML training experiments are all conducted on this platform.

The research platform also contains an implementation of the TEE-part partial derivative calculation (Algorithm 1). The implementation is on a T-470P Laptop that runs Ubuntu 18.04 and is equipped with an i7-7700HQ CPU, 16 GB memory, and 256 GB SSD storage. We will use this implementation to study TEE-part performance.

B. FDML Applications

We design two applications. The first application is the user authentication based on user touch patterns on smartphone screens. We refer to it as *TouchAuth*. While the second one is the user authentication based on photos captured by cameras. We refer to it as *CameraAuth*. The training data of TouchAuth are the series of smartphone screen touch events, while CameraAuth uses human photos as the training data.

To perform training, the end device will capture the data (touch patterns or photos) of its owner and label them with its ID. Note that the cloud does not need to perform additional labeling. The captured data will be used as the training data. When the DML model is trained, to use the model, a smart device with the model deployed will first use its sensors to collect data of its current user. Then, embeddings will be extracted from the collected data by the trained DML model. At last, the smart device computes the distance between the embeddings and its owner's embedding template. If the distance is less than a given threshold, the model predicts that the legitimate user is using that smart device, otherwise not.

TouchAuth Emulation. We use the real-world screen touch data of the volunteer employees of Alipay. We collected screen touch events of the mobile payment app of that company from the volunteers' Android phones. Each touch event is represented by a 10-dimensional vector (details in the Appendix), and 30 consecutive touch events of one user that occur within 30 minutes constitute a training/test sample. The screen touch data about password typing are not collected. The data collection lasted a month, and during the process, the volunteers were asked to use the mobile payment app as usual. This data collection passed Alipay's internal review, and we comply with the volunteer data usage policy[2]

We select active app users from the volunteers and save the data from 4 phone models with the most users. There are 1397 users in the training set, and the number of users of the four smartphone models is 507, 333, 287, and 270, respectively. 120 users are in the test set, and each phone model has 30 users. The users in the training set and test set have no overlap.

For each training round, the cloud selects a phone model with equal probability and randomly chooses a predefined number of users who use this phone model as the participants of this round. Each chosen participant extracts training samples from its touch events using a data augmentation method which introduces randomness in deciding the first touch event of a sample. (In most cases, each participant has more than 25 samples.) The participant then randomly select a predetermined number of samples from the extracted samples as its training data for this round. For accuracy testing, for each user in the test set, we extract 40 samples from the touch data of that user to form a fixed test dataset. The embedding network used for this application contains four convolutional layers and one fully connected layer (Details are in the Appendix).

CameraAuth Emulation. We randomly select the photos of 400 people from the VGGFace2 dataset [40] to start the study. We use 300 of those people as the participants of the FDML training, and the rest 100 are used for testing. Each person in the training dataset has 70 different pictures of its own, while in the testing dataset, that number is 40. ResNet-34 [41] is used as the embedding network with the output size set as 64.

C. Evaluation Metrics

DML Model Accuracy. Given a trained DML model, for one user (or person) in the test set, we compute the embeddings of the samples of that user and divide the (40) embeddings into three parts: 20 for embedding template calculation ($\{emb\}_A$), 10 for distance threshold selection ($\{emb\}_B$), and 10 for accuracy evaluation ($\{emb\}_C$). The embedding template of one user is calculated by averaging all the embeddings in its $\{emb\}_A$. The threshold for authenticating one user is found by using Algorithm 4 with parameter $\{emb\}_{self}$ set as that user's $\{emb\}_B$ and with parameter $\{emb\}_{other}$ set as other ten users' $\{emb\}_B$. Algorithm 4 works by choosing the Equal Error Rate (EER) point as the threshold, and similar methods are widely used in biometric authentication [42], [43]. As to computing the accuracy of the authentication, for one user, we use the threshold found by Algorithm 4 to distinguish its $\{emb\}_C$ from the $\{emb\}_C$ of another ten users (those users

[2]Volunteer Data Usage Policy. First, volunteers in the research study fully understand the purpose and how their data are collected. Second, collected data are processed to make them anonymous and protected during the whole study. Last, they also will be deleted afterwards.

have no overlap with the users used in threshold selection). We use the averaged accuracy of all the users in the test set as the model accuracy.

Algorithm 4: Threshold Selection

Inputs : Embeddings of the target participant : $\{emb\}_{self}$; embedding template: $template$; embeddings for reference: $\{emb\}_{other}$

Output: $threshold$

1 Compute the distances between $template$ and all the embeddings in $\{emb\}_{self}$ as $\{dist\}_{self}$, and label them as the positive class;
2 Compute the distances between $template$ and all the embeddings in $\{emb\}_{other}$ as $\{dist\}_{other}$, and label them as the negative class;
3 Use a series of $threshold$s to predict the label of the items in $\{dist\}_{self}$ and $\{dist\}_{other}$, and return the $threshold$ that makes FPR $\approxeq$ FNR.

Privacy protection. We will use the privacy budget ϵ of differential privacy to show the privacy protection level.

System Performance. Since the TEE-part could be a performance bottleneck, we will report the TEE-part processing time of calculating the partial derivatives.

VIII. Experiments and results

A. Experiment Settings

We have these default settings for the FDML experiments:

(1) The participant number ($num_{required}$ in Algorithm 3) and the training sample number of each participant at each round are 64 and 8, respectively.

(2) The learning rate of FDML is 1e-4, and Adam optimizer is used on the cloud side.

(3) We set the maximum number of training rounds as 15K and 1.6K for experiments of TouchAuth and CameraAuth, respectively.

B. Experiment Results

*1) **FDML VS traditional FL**:* We conduct experiments to compare the accuracy gap between FDML and traditional FL when training user authentication models. We use the FL design with FederatedAveraging proposed in [44] to implement an FL training emulation platform. We refer to this platform as FL-traditional in this paper. Since existing FL does not support DML loss functions, to conduct training on FL-traditional, we replace the DML loss with the cross-entropy loss, and the last layer of the original embedding network is connected with a fully connected layer whose output has as many dimensions as the number of participants. The participant of FL-traditional will only perform one training iteration in each training round. The other settings of FL-traditional are the same as our FDML research platform. We train authentication models with TouchAuth and CameraAuth on the two platforms, and test the accuracy with the method mentioned in Section VII-C. To make fair comparisons, we disable the privacy-preserving design and robustness design and assume there will be no exceptions in the model training. The maximum numbers of training rounds for FL-traditional are 45K and 4.5K. We try different learning rates ($lr = 0.1, 0.01$) for the experiments on FL-traditional and report the results in Table I). The results show that FDML-based training achieves the highest accuracy and outperforms traditional FL training by a clear margin on both the two applications.

TABLE I
User Authentication Accuracy of FDML and FL

Training method	FDML	FL (lr=0.1)	FL (lr=0.01)
TouchAuth-I	**80.88**	71.4	63.24
CameraAuth	**79.98**	76.55	72.27

*2) **Roubstness design: To continue or to drop**:* As discussed in Section VI, when some participants cannot upload their local gradients as expected, the cloud can choose a strategy from abandoning the current round or continuing the training. We now conduct experiments to show continuing the training is better. We compare two strategies: (1) Abandon the current round and Start a new round (Strategy-AS). The cloud will abandon the current round if exceptions happen. (2) Unconditionally Continue (Strategy-UC). The cloud will always continue the current round if more than 2 participants are on schedule.

TABLE II
Accuracy of FDML Training with Different Exception Handling Strategies

Application	TouchAuth			CameraAuth		
Exception probability	0.2	0.4	0.6	0.2	0.4	0.6
Strategy-AS	57.86	58.61	58.61	55.92	55.38	54.45
Strategy-UC	81.11	80.94	80.94	77.46	80.54	79.53

We conduct experiments on TouchAuth and CameraAuth to evaluate the two strategies under different exception probabilities. Under each setting, the participants have the same probability of failing to upload their local gradients. Table II shows the results. We can find that strategy-AS achieves very low accuracy on both two applications ($< 59\%$), which indicates strategy-AS cannot work at all. Strategy-UC always has better accuracy ($> 77\%$) and can work well. In addition, the results under Strategy-UC across different exception probability are similar, having a difference smaller than 3.5%, and this indicates FDML can withstand high user dropout rates when Strategy-UC is chosen.

TABLE III
Accuracy of TouchAuth with Privacy and Robustness Designs Enabled

$rate_p$ (for robustness)	1	0.8	0.6
w/o DP	**80.88**	80.75	80.16
$\epsilon = 8$	80.55	**80.57**	71.40
$\epsilon = 4$	**74.56**	73.93	73.97

*3) **Impact of privacy protection and system robustness design on FDML training accuracy**:* We now show the training results with privacy and system robustness designs enabled. We conduct experiments for TouchAuth and CameraAuth. We set the bound T in Algorithm 2 as 2e-5 for all the experiments.

TABLE IV
ACCURACY OF CAMERAAUTH WITH PRIVACY AND ROBUSTNESS DESIGNS ENABLED

$rate_p$	1	0.8	0.6
w/o DP	**79.98**	79.86	79.1
$\epsilon = 1$	**79.94**	78.88	78.87
$\epsilon = 0.5$	**78.3**	77.7	75.51
$\epsilon = 0.25$	73.52	**73.93**	71.39

For TouchAuth, we set the embedding drop rate r of Algorithm 2 as 0.2, and for CameraAuth, we set that value as 0. Note that the embedding drop is conducted in an active manner and can be compensated by increasing the number of the participants and the number of each participant's training sample in a training round, so the choices on the embedding drop rate will not affect the model accuracy too much in practice. We set k and $num_{required}$ in Algorithm 3 as 1 and 64, respectively. And since it is difficult to simulate the timeout in Algorithm 3 in a meaningful way without leaking confidential business information, we assume that Line 12 of Algorithm 3 will not be executed. We then vary the ϵ (the privacy budget) and the $rate_p$ (the expected participant response rate) to conduct experiments.

Table III and Table IV shows the experimental results of TouchAuth and CameraAuth, respectively. The results show that the model accuracy decreases as the privacy protection level increases (the budget ϵ decreases). This shows a clear trade-off between the privacy protection and the model accuracy, and we can find the ϵ that achieves a reasonable balance point for most settings in Table III and Table IV (ϵ set as 8 for TouchAuth and set as 0.5 for CameraAuth). The results also suggest that, when preparing the training, the model trainer needs to carefully set the ϵ to find a balance point, and this process can be performed based on domain knowledge or based on the results of the training conducted on similar datasets.

From Table III and Table IV, we can also find that, under the same privacy protection level, the experiments with $rate_p$ set as 1 and 0.8 have similar accuracy. It indicates that when 20% of participants cannot (or don't need to upload) their local gradients, our design still works well. Also, we can see that even when $rate_p$ is 0.6, the accuracy is still considerable.

*4) **Performance of TEE**:* Since the TEE part could be a performance bottleneck, to test the TEE-side processing time of Algorithm 1, we choose the triplet loss with the "batch all" strategy [45] which has the highest computational cost among all well-known strategies. We vary the participant number in each round and report the results in Table V. We can find that it just takes about 2 s and 11 s to calculate the partial derivatives when there are 128 and 256 participants in one training round respectively. The results indicate that the performance of our design is satisfactory for reasonable choices on the number of required participants for each training round (e.g., < 256), and the TEE part will not be a bottleneck.

TABLE V
PERFORMANCE TEST ON TEE

Participant number ($num_{required}$)	64	128	256	512
Processing time (s)	0.48	2.19	11.07	63.65

IX. CONCLUSION

In this paper, we first introduce the concept of FDML, propose a practical, privacy-preserving, and robust FDML framework design that can support large scale participants. We also gives theoretical bounds of training convergence and privacy protection level of the differentially private mechanism. The proposed framework is shown to be practical with both privacy and robustness considerations throughout our experiments.

X. ACKNOWLEDGEMENT

This work was supported in part by the National Key R&D Program of China (#2021YFB3100300, and #2020YFB1005900), NSFC (#61872180, #61972200, and #61872179), Jiangsu "Shuang-Chuang" Program, Jiangsu "Six-Talent-Peaks" Program, Ant Financial through the Ant Financial Science Funds for Security Research, and the program B for outstanding Ph.D. candidate of Nanjing University.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv:1602.05629*, 2016.
[2] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, 2017.
[3] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015.
[4] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems*, vol. 74, pp. 76–85, 2017.
[5] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *TIFS*, vol. 13, no. 5, pp. 1333–1345, 2018.
[6] A. Imteaj and M. H. Amini, "Distributed sensing using smart end-user devices: pathway to federated learning for autonomous iot," in *CSCI*, 2019.
[7] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv:1812.00564*, 2018.
[8] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities and challenges," *arXiv:1908.06847*, 2019.
[9] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv:1912.04977*, 2019.
[10] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.
[11] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *SIMBAD*, 2015.
[12] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *CVPR*, 2016.
[13] K. Sheng, W. Dong, W. Li, J. Razik, F. Huang, and B. Hu, "Centroid-aware local discriminative metric learning in speaker verification," *Pattern Recognition*, vol. 72, pp. 176–185, 2017.
[14] J. Lu, J. Hu, and Y.-P. Tan, "Discriminative deep metric learning for face and kinship verification," *TIP*, vol. 26, no. 9, pp. 4269–4282, 2017.

[15] Z. Huang, R. Wang, S. Shan, and X. Chen, "Projection metric learning on grassmann manifold with application to video based face recognition," in *CVPR*, 2015.
[16] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W. Zheng, and S. Z. Li, "Embedding deep metric for person re-identification: A study against large variations," in *ECCV*, 2016.
[17] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai, "Hard-aware point-to-set deep metric for person re-identification," in *ECCV*, 2018.
[18] Council of European Union, "General data protection regulation," 2018, https://gdpr-info.eu/.
[19] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv:1811.03604*, 2018.
[20] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv:1812.02903*, 2018.
[21] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *arXiv:1802.07876*, 2018.
[22] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *arXiv:1812.03337*, 2018.
[23] H. H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated reinforcement learning," *arXiv:1901.08277*, 2019.
[24] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "Secureboost: A lossless federated learning framework," *arXiv:1901.08755*, 2019.
[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *TIST*, vol. 10, no. 2, pp. 1–19, 2019.
[26] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.
[27] Intel Corporation, "Intel software guard extensions for linux," https://01.org/intel-softwareguard-extensions, 2016.
[28] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *ICC*, 2019.
[29] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017.
[30] J. Passerat-Palmbach, T. Farnan, R. Miller, M. S. Gross, H. L. Flannery, and B. Gleim, "A blockchain-orchestrated federated learning architecture for healthcare consortia," *arXiv:1910.12603*, 2019.
[31] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv:1811.04017*, 2018.
[32] M. Dahl, J. Mancuso, Y. Dupis, B. Decoste, M. Giraud, I. Livingstone, J. Patriquin, and G. Uhma, "Private machine learning in tensorflow using secure computation," *arXiv:1810.08130*, 2018.
[33] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv:1902.01046*, 2019.
[34] S. A. Osia, A. S. Shamsabadi, A. Taheri, K. Katevas, S. Sajadmanesh, H. R. Rabiee, N. D. Lane, and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics," *arXiv:1703.02952*, 2017.
[35] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *CCS*, 2017.
[36] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *S&P*, 2019.
[37] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *KDD*, 2018.
[38] X. Zhou, "On the fenchel duality between strong convexity and lipschitz continuous gradient," *arXiv:1803.06573*, 2018.
[39] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. Cohen, "Revisiting training strategies and generalization performance in deep metric learning," in *ICML*, 2020.
[40] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *FG*, 2018.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[42] K. Mock, B. Hoanca, J. Weaver, and M. Milton, "Real-time continuous iris recognition for authentication using an eye tracker," in *CCS*, 2012.
[43] Y. N. Singh, S. K. Singh, and P. Gupta, "Fusion of electrocardiogram with unobtrusive biometrics: An efficient individual authentication system," *Pattern Recognition Letters*, vol. 33, no. 14, pp. 1932–1941, 2012.
[44] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.
[45] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv:1703.07737*, 2017.

APPENDIX

More details of the TouchAuth application. For each touch event collect from the volunteers, we use a 10-dimensional vector to represent it, i.e, $< x_s,\ y_s,\ x_e,\ y_e,\ x_d,\ y_d,\ dur,\ p,\ s,\ t >$. In this vector, x_s, y_s and x_e, y_e are the coordinates of the start and end points of one touch which are normalized by the screen size. x_d and y_d represent the touch direction, e.g. if $x_e - x_s > 0$, $x_d = 1$, otherwise $x_d = 0$. dur is the touch duration which is firstly clipped by an upper bound (1600 ms) and then normalized by the bound. p and s are the touch pressure and touch size, respectively. Their ranges are already between 0 and 1. t is the touch type, 0 for swipe, and 1 for tap. One sample for training or testing contains 30 touch events, and the size of the input of the embedding network is (1, 1, 30, 10).

The network structure of the DML model (embedding network) for this application is shown in the form of PyTorch code:

```
Conv2d(1, 64, (1, 10), 1))
Conv2d(64, 128, (1, 1), 1)
MaxPool2d((3, 1), (2,1))
Conv2d(128, 256, (1, 1), 1)
Conv2d(256, 256, (1, 1), 1)
Linear(64)
```

Each convoluntional layer is followed by a rectified linear unit.