Communicational and Computational Efficient Federated Domain Adaptation

Hua Kang, Zhiyang Li, and Qian Zhang, *Fellow, IEEE*

Abstract—The emerging paradigm of Federated Learning enables mobile users to collaboratively train a model without disclosing their privacy-sensitive data. Nevertheless, data collected from different mobile users may not be independent and identically distributed. Thus directly applying the trained model to a new mobile user usually leads to performance degradation due to the so-called domain shift. Unsupervised Domain Adaptation is an effective technique to mitigate domain shift and transfer knowledge from labeled source domains to the unlabeled target domain. In this article, we design a Federated Domain Adaptation framework that extends Domain Adaptation with the constraints of Federated Learning to train a model for the target domain and preserve the data privacy of all the source and target domains. As mobile devices usually have limited computation and communication capabilities, we design a set of optimization methods that significantly enhance our framework's computation and communication efficiency, making it more friendly to resource-constrained edge devices. Evaluation results on three datasets show that our framework has comparable performance with the standard centralized training approach, and the optimization methods can reduce the computation and communication overheads by up to two orders of magnitude.

Index Terms—Federated learning, domain adaptation, communicational efficient

1 Introduction

With the proliferation of intelligent devices, data have multiplied at the network edge and enabled many applications including health care, smart home, and behavior analysis. Deep learning tools capable of handling complicated data patterns have been widely adopted for data processing and analysis [1], [2], [3]. Traditional model training takes a centralized approach where all training data need to be gathered in a single place. However, due to the limited network bandwidth as well as privacy concerns, it is impractical to take such a centralized learning approach. As a result, data will be stored and processed locally with the emerging technology of mobile edge computing (MEC). Recently, Federated Learning (FL) [4] has been proposed to keep all participants' data local during the training process. However, different participants may have data collected from different distributions and the target participant where the model to be applied may only have limited unlabeled data. Therefore, a Transfer Learning technique named Unsupervised Domain Adaptation (UDA) [5], [6] is typically used for the training, which can transfer the knowledge from the labeled source domain to the unlabeled target domain. Fig. 1 shows some common scenarios where different domain information including different rooms, persons, devices and so on, have an impact on the corresponding data distributions. In addition, the collected data are stored in different places, and these scenarios call for a secure federated domain adaptation method. However, currently there is no secure FL framework suitable for UDA. One possible candidate is the Federated Transfer Learning framework proposed in [7], [8]. Nevertheless, it does not fit the scenario either. First, it requires some labeled data in the target domain, but the target domain may have no labeled data because it is burdensome to require the target user to label their data. Second, its alignment loss definition is based on co-occurrence pairs between source domain samples and target domain samples but such a pairing may not exist. Another approach [9] integrates FL with adversarial DA. However, the work mainly focuses on the model architecture design without a detailed discussion of privacy-preserving methods and communication costs.

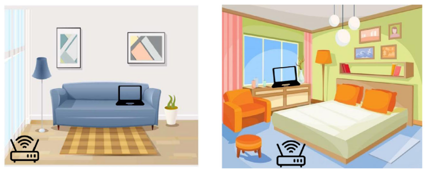
In this paper, we propose an FL framework for UDA. We use Maximum Mean Discrepancy (MMD) [10] loss as the alignment loss in our framework, which is widely used in UDA. Moreover, for the necessary information exchange during training, we use the well-known Homomorphic Encryption scheme named Paillier [11] to encrypt the information to prevent indirect data leakage and also make computation on ciphertext viable. For limited computation and communication capabilities of edge devices, we propose two optimization methods to further improve the efficiency. The model in our framework is composed of two parts: feature extractor and classifier (or regressor for regression tasks).

There are three challenges in our framework design:

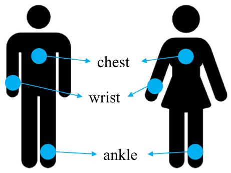
1) *How to compute MMD loss and gradients in a federated setting?* MMD loss calculation involves data from both

- *Hua Kang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. E-mail: hkangae@cse.ust.hk.*
- *Zhiyang Li is with ByteDance Inc., Hangzhou, Zhejiang 310030, China. E-mail: lizhiyang.zy@bytedance.com.*
- *Qian Zhang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. E-mail: qianzh@cse.ust.hk.*

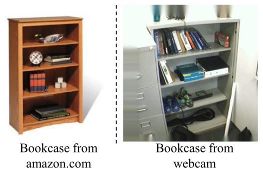
Manuscript received 8 June 2021; revised 1 Apr. 2022; accepted 6 Apr. 2022. Date of publication 14 Apr. 2022; date of current version 11 July 2022.
This work was supported by the RGC under Grants CERG 16204418, 16203719, 16204820, and R8015; and in part by the Guangdong Natural Science Foundation under Grant 2017A030312008.
(Corresponding author: Qian Zhang.)
Recommended for acceptance by B. DiMartino.
Digital Object Identifier no. 10.1109/TPDS.2022.3167457



(a) Different rooms' receivers collect WiFi signal data for tasks like gesture recognition, and the received data are influenced by different rooms.



(b) Different sensors attached to different persons collect IMU data for human activity recognition, and the collected data are influenced by different persons.



(c) Different cameras capture images for classification, and the images are influenced by the environments where they are collected and the cameras' quality.

Fig. 1. Common tasks that collect data separately and are influenced by different domain information

source and target domains. In a federated setting, data of different domains are located on different devices, and we need to guarantee the data privacy of all participants. By analyzing the structure of MMD loss and dividing it into three parts, we find that each part has a unique relationship with the source and target domain data. We design a scheme for both domains to securely exchange necessary information and collaboratively compute the MMD loss and gradient values.

2) *How to incorporate the Paillier encryption scheme into MMD loss and gradient calculation?*MMD loss is composed of kernel functions, typically Radial basis function (RBF) kernels. The RBF kernel involves the evaluation of the exponential function, but Paillier only supports ciphertext additions. To cope with this challenge, we substitute the exponential function with the first few terms in its Taylor Series, transforming the RBF kernel into a polynomial. Next, we specify the items to be encrypted and sent by both domains to compute this polynomial supported by Paillier.

3) *How to minimize the computation and communication overheads during model training?*It is essential to improve the efficiency of data located on edge devices with limited computation and communication capabilities. To this end, we propose two optimization methods. First, based on the mathematical characteristic of MMD loss, we conduct a transformation showing that both domains can calculate and send less information than the trivial method. Second, by applying the chain rule of derivative to the normal gradient calculation process, we observe that we only need to care about the intermediate derivatives, which are far fewer than the final gradients.

We build a prototype implementation and conduct experiments on three different datasets with data formats varying from image and WiFi sensing to wearable sensor data. The results show that our Federated Domain Adaptation framework has comparable performance with the standard centralized training approach. Moreover, further evaluation shows that our optimization methods can reduce our framework's computation and communication overheads by up to two orders of magnitude. In addition, the more complex the model, the more significant the reduction of computation and communication overheads.

Our contribution can be summarized as follows:

- We design a Federated Domain Adaptation framework that combines FL with UDA.
- We propose two optimization methods that significantly reduce the computation and communication overheads during training.
- We build a prototype implementation and conduct extensive experiments that show our framework's effectiveness.

2 Related Work

2.1 Federated Learning

The concept of Federated Learning (FL) was first proposed by McMahan *et al.* [4]. They targeted the Horizontal FL scenario and proposed a Federated Averaging algorithm for model training. In each round of the algorithm, each participant trains the current model for multiple epochs and sends the updated weights to the central server. The server computes the weighted average over the uploaded weights as the new weights. After that, many improvements or extensions to Horizontal FL have been proposed. Smith *et al.* [12] incorporated Multi-Task Learning into FL to solve the statistical and system challenges faced by Horizontal FL. The statistical challenge refers to the non-IID distribution of data of different participants, and the system challenge refers to the distinct capabilities of different training nodes, which may result in problems like stragglers. Zhao *et al.* [13] focused on the statistical challenges and proposed to use a globally shared dataset to improve the accuracy. Briggs *et al.* [14] also targeted this challenge and proposed Hierarchical Clustering, which separates participants into groups according to the similarity of their updates, and trains an independent model for each group. Chen *et al.* [15] combined Meta-Learning with FL and shared a parameterized algorithm among participants rather than a model. In terms of security, Bonawitz *et al.* [16] proposed a Secure Aggregation protocol in which the exact update values uploaded by each participant are hidden from the central server. However, the server can still calculate their weighted average values. As for communication cost optimization in Horizontal FL, Konecný *et al.* [17] proposed two methods namely structured updates and sketched updates to reduce the communication overhead. Wang *et al.* [18] proposed a Communication Mitigation method in which a participant will not upload its update if its update tendency is different from that of the global model. Note that these communication cost optimization methods are tailored for Horizontal FL and are not applicable to Federated Transfer Learning frameworks.

Apart from Horizontal FL, Vertical FL and Federated Transfer Learning are also gradually becoming the research

TABLE 1
List of Symbols

Symbol	Description
N	Batch size in the pretraining phase
n_s	Source batch size in the fine-tuning phase
n_t	Target batch size in the fine-tuning phase
C	Number of classes
M	Number of monomials of the approximated kernel function
S	Number of source domains
L	Length of the feature vector
p	Ciphertext size of Paillier encryption
v_i^s	Source sample i's feature vector
v_j^t	Target sample j's feature vector
y_i^j	Ground truth of sample j's class i
s_i^j	Prediction score of sample j's class i
γ	Loss weight of the regularization loss
λ	Loss weight of the MMD loss
γ_1	Loss weight of source regularization loss in the fine-tuning phase
γ_2	Loss weight of target regularization loss in the fine-tuning phase
Θ^s	Parameters of the source model
Θ^t	Parameters of the target model
L_C	Classification loss
L_Θ	Regularization loss
L_{MMD}	MMD loss

focus. For Vertical FL, Hardy *et al.* [19] used Entity Resolution to map the samples with the same identity and build a linear model from both parties' data. As for Federated Transfer Learning, Liu *et al.* [7] proposed a framework suitable for semi-supervised transfer learning. Sharma *et al.* [8] further incorporated Multi-Party Computation and Secret Sharing into this framework. Still, basic assumptions, including the semi-supervised scenario and alignment loss format, are not changed. This framework can not apply to UDA due to the reasons mentioned in the introduction.

2.2 MMD Based Domain Adaptation

Ghifary *et al.* are one of the earliest researchers that propose to use Maximum Mean Discrepancy (MMD) as alignment loss for Deep Domain Adaptation (DA) [10]. They showed that MMD based DA leads to higher accuracy than other state-of-the-art DA approaches. After that, MMD loss becomes widely used in DA. Tzeng *et al.* [20] proposed a method to find the best place and size of the adaptation layer, whose output is regarded as feature vectors and used to compute MMD loss. Long *et al.* [21] regarded the last few layers of the model as adaptation layers and applied MMD to each of them. They further extended this method and proposed Joint MMD [22] to minimize the distance between the joint distribution of these layers. Zhang *et al.* [23] applied MMD to both the feature vectors and the final output scores. Yan *et al.* [24] pointed out that different class weights between the source and target domain will affect the effectiveness of DA and proposed Weighted MMD to cope with this problem. Rozantsev *et al.* [25] proposed to loosen the weight sharing restriction between feature extractors of the source and target domain and achieved better performance. We build on top of these works and integrate FL with MMD based DA.

3 List of Symbols

We list the symbols used in our framework in Table 1.

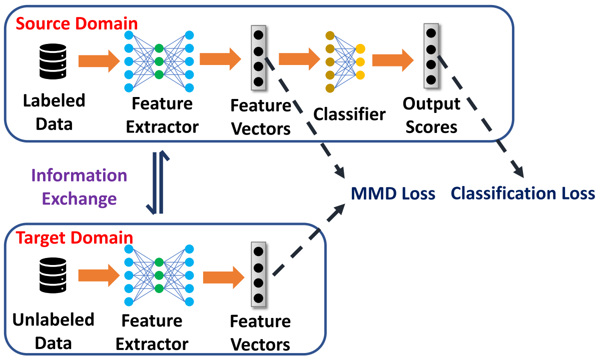


Fig. 2. A sketch of our federated domain adaptation framework for one source and one target user.

4 Federated Domain Adaptation

In many applications, data collected from the target domain are typically unlabeled since it is time-consuming and not user-friendly to label the data. Therefore, UDA is used to train the target model with the help of labeled data from other participants or existing datasets.

To protect the privacy of the data holders, we design a Federated Domain Adaptation framework that keeps the data local during the model training process. A sketch of our framework is shown in Fig. 2. For simplicity, we first discuss the situation with only one source and one target. Extension to support multiple participants with labeled data and one participant with unlabeled data is shown in Section 4.4. The source domain will train both the feature extractor and the classifier, whereas the target domain will only train the feature extractor due to lack of labels. The feature extractors of the source and target domains adopt the same architecture, but their weights in our framework are not the same to have a better performance [25]. There are two main loss terms during the training procedure: classification loss and MMD loss. We can easily replace the classifier with a regressor and the classification loss with the regression loss for regression tasks. Classification loss is calculated on the output score of the classifier in the source domain, and MMD loss is calculated on the feature vectors of both domains. Optional regularization terms such as weight decay can also be added to the loss function. Information needs to be exchanged between the two domains for loss and gradient calculation during the training process. When the whole training process finishes, we concatenate the feature extractor in the target domain and the classifier in the source domain to get the final model for the target domain.

Our framework adopts the typical training procedure for Domain Adaptation, containing a pretraining phase and a fine-tuning phase. Next, we describe these two phases in detail in Sections 4.1 and 4.2 respectively. Then in Section 4.3, we introduce two optimization methods to reduce communication and computation costs further. After that, the framework for one source domain and one target domain is complete. In Section 4.4, we extend the framework to fit the multi-source domains and one target domain situation. Finally, we analyze the communication cost in Section 4.5.

4.1 Pretraining Phase

The source domain pretrains the feature extractor and classifier to prepare for the fine-tuning phase during the pretraining phase. Before pretraining begins, the source and target domain must first agree on the detailed model structure for the feature extractor and the classifier. Then the source domain starts to train its feature extractor and classifier using its own data. The loss function during pretraining can be defined as follows:

$$L_{pretrain} = L_C + \gamma L_\Theta, \tag{1}$$

where L_C denotes the classification loss, and L_Θ denotes an optional regularization term. Typically, Cross Entropy loss is used as classification loss. That is, $L_C = \frac{1}{N}\sum_{j=1}^{N}[-\sum_{i=1}^{C} y_i^j \log(s_i^j)]$, where N represents batch size, C represents the number of classes, y_i^j and s_i^j represent the ground truth and output score for each class of sample j.

When pretraining finishes, the source domain sends the weights of the feature extractor to the target domain. The target domain uses it to initialize its own feature extractor.

4.2 Fine-Tuning Phase

After the target domain initializes its feature extractor with pretrained weights from the source domain, they can start the fine-tuning phase and train the whole model collaboratively. For each batch, the source and target domain need to exchange the necessary information to calculate the current loss value and gradients for their own model part. At this phase, the total loss is defined as follows:

$$L_{finetune} = L_C + \lambda L_{MMD} + \gamma_1 L_{\Theta^s} + \gamma_2 L_{\Theta^t}, \tag{2}$$

where L_C still represents the classification loss on the labeled data of the source domain. L_{Θ^s} and L_{Θ^t} are optional regularization terms for the source and target domains, respectively. L_{MMD} represents the MMD loss. Let n_s and n_t denote the batch size of the source and target domains respectively, $\{v_i^s\}_{i=1}^{n_s}$ and $\{v_j^t\}_{j=1}^{n_t}$ denote the feature vectors of one batch of the source and target domains respectively. The L_{MMD} of one batch can be formulated as:

$$L_{MMD} = L_{p1} + L_{p2} + L_{p3}, \tag{3}$$

$$L_{p1} = \frac{1}{n_s(n_s-1)}\sum_{i=1}^{n_s}\sum_{i'=1,i'\neq i}^{n_s} k(v_i^s, v_{i'}^s), \tag{4}$$

$$L_{p2} = \frac{1}{n_t(n_t-1)}\sum_{j=1}^{n_t}\sum_{j'=1,j'\neq j}^{n_t} k(v_j^t, v_{j'}^t), \tag{5}$$

$$L_{p3} = -\frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t} k(v_i^s, v_j^t), \tag{6}$$

where $k(v, v')$ represents the RBF kernel function, i.e., $k(v, v') = \exp(-\alpha\|v - v'\|^2)$ where α is the kernel function parameter.

For each batch, we need to calculate the loss value to monitor the training process and the gradients to update the model parameters. Each batch of the fine-tuning phase can be divided into the following five steps:

- *Feed forward*: Source domain and target domain forward a batch of samples through the model.
- *Classification loss and gradients calculation*: Source domain calculates the classification loss L_C and its gradients on its own. Since the target domain only has unlabeled data, it is not involved in this step.
- *Regularization loss and gradients calculation*: L_{Θ^s} and its gradients are only related to the model parameters in the source domain. Similarly, L_{Θ^t} and its gradients are only associated with the model parameters in the target domain. Thus, the source domain is responsible for the calculation of L_{Θ^s} and its gradients, whereas the target domain is responsible for L_{Θ^t} and its gradients.
- *MMD loss and gradients calculation*: Source domain and target domain work together to calculate the MMD loss and gradient values.
- *Parameter update*: Source domain and target domain update their model parameters respectively with the calculated gradients.

Except for *MMD loss and gradients calculation*, all other steps are performed on either domain separately. Since MMD loss comprises feature vectors of both domains, we need collaboration between the source and target domains to calculate its loss value and gradients. Thus the procedure is much more complex than other steps. Next, we elaborate on the MMD loss and gradients calculation process.

4.2.1 MMD Loss Calculation

In the MMD loss definition, we notice that the first part L_{p1} is only related to feature vectors in the source domain. Therefore, the source domain can calculate L_{p1} on its own. Similarly, the target domain can calculate L_{p2} on its own. However, for L_{p3}, each kernel function $k(v_i^s, v_j^t)$ contains a feature vector v_i^s from the source domain and a feature vector v_j^t from the target domain. In our framework, we let the source domain send encrypted information about each feature vector v_i^s to the target domain, and the target domain is responsible for calculating L_{p3}. To enable computation on ciphertexts, we use Paillier [11], an Additive PHE system. However, Paillier only allows additive operations and does not support the exponential function in the kernel. To tackle this problem, we leverage the Taylor series of the exponential functions, and use its first few terms for approximation [7], [19]. Specifically, for exponential function we have $\exp(x) = \sum_{k=0}^{\infty}\frac{x^k}{k!}$. We can keep the first n terms $\sum_{k=0}^{n-1}\frac{x^k}{k!}$ in this Taylor series and use it to substitute the exponential function in the kernel. After this step, the kernel function $k(v_i^s, v_j^t)$ becomes a polynomial of elements in v_i^s and v_j^t.

For the approximated kernel $k(v_i^s, v_j^t)$, suppose that vector $v_i^s = (a_1, a_2, \ldots, a_L)$, and vector $v_j^t = (b_1, b_2, \ldots, b_L)$. Let $f_m(v_i^s)$ represent a constant 1 or a monomial composed of elements in v_i^s, which appears in the approximated kernel. Similarly, let $g_m(v_j^t)$ represent 1 or a monomial composed of elements in v_j^t. Let M be the total number of monomials. Then the approximated kernel can be written as:

$$k(v_i^s, v_j^t) = \sum_{m=1}^{M} c_m f_m(v_i^s) g_m(v_j^t), \tag{7}$$

where c_m is the constant coefficient for each term in the approximated kernel. For example, if there is a term $3a_1a_2b_1^2$

in the approximated kernel, then for this term we have $c_m = 3$, $f_m(v_i^s) = a_1 a_2$, and $g_m(v_j^t) = b_1^2$. From this kernel function representation, we can see that in order to calculate the value of this kernel $k(v_i^s, v_j^t)$, the source domain needs to send the ciphertexts for all $\{f_m(v_i^s)\}_{m=1}^M$ to the target domain. Since the target domain can calculate the ciphertexts for all $\{g_m(v_j^t)\}_{m=1}^M$, it can now calculate the ciphertexts of $k(v_i^s, v_j^t)$ based on the additive property of the Paillier crypto system.

For the target domain to calculate L_{p3}, the source domain needs to compute $\{\{f_m(v_i^s)\}_{m=1}^M\}_{i=1}^{n_s}$ for all the feature vectors in the batch, encrypts them and sends them to the target domain. The target domain then computes the ciphertext of $k(v_i^s, v_j^t)$ for each vector pair (v_i^s, v_j^t), adds them together and finally multiplies the sum with the coefficient in L_{p3} to get the ciphertext of L_{p3}.

4.2.2 MMD Gradients Calculation

MMD gradients are the partial derivatives of MMD loss with respect to model parameters. Since MMD loss is based on the feature vectors of the two domains, it is not relevant to the parameters in the classifier of the source domain. For a model parameter θ^s in the feature extractor of the source domain, the partial derivative can be written as:

$$\frac{\partial L_{MMD}}{\partial \theta^s} = \frac{\partial L_{p1}}{\partial \theta^s} + \frac{\partial L_{p2}}{\partial \theta^s} + \frac{\partial L_{p3}}{\partial \theta^s} = \frac{\partial L_{p1}}{\partial \theta^s} + \frac{\partial L_{p3}}{\partial \theta^s}. \quad (8)$$

The partial derivative of L_{p2} can be omitted because it is only related to feature vectors in the target domain, thus is not relevant to θ^s. Similarly, for a model parameter θ^t in the feature extractor of the target domain, we have:

$$\frac{\partial L_{MMD}}{\partial \theta^t} = \frac{\partial L_{p2}}{\partial \theta^t} + \frac{\partial L_{p3}}{\partial \theta^t}, \quad (9)$$

where the partial derivative of L_{p1} is omitted.

For the source domain, L_{p1} only contains its own feature vectors, thus it can compute $\frac{\partial L_{p1}}{\partial \theta^s}$ on its own. In contrast, L_{p3} contains feature vectors from the target domain, so the source domain needs information about these feature vectors to calculate $\frac{\partial L_{p3}}{\partial \theta^s}$. Since L_{p3} is basically a sum of $k(v_i^s, v_j^t)$, we next focus on the partial derivative of $k(v_i^s, v_j^t)$ with respect to θ^s.

Based on Equation (7) of the approximated kernel, this derivative can be written as:

$$\frac{\partial k(v_i^s, v_j^t)}{\partial \theta^s} = \sum_{m=1}^{M} c_m g_m(v_j^t) \frac{\partial f_m(v_i^s)}{\partial \theta^s}, \quad (10)$$

where $g_m(v_j^t)$ only contains feature vector elements of the target domain, thus it is not relevant to θ^s. In the above equation, $g_m(v_j^t)$ is unknown to the source domain, therefore the target domain needs to send the ciphertexts of all $g_m(v_j^t)$ to the source domain, so that the source domain can calculate the encrypted value of $\frac{\partial k(v_i^s, v_j^t)}{\partial \theta^s}$ and $\frac{\partial L_{p3}}{\partial \theta^s}$.

A similar analysis also applies to the target domain. For $\frac{\partial L_{p2}}{\partial \theta^t}$, the target domain can compute it on its own. As for $\frac{\partial L_{p3}}{\partial \theta^t}$, the target domain needs encrypted $f_m(v_i^s)$ of each feature vector from the source domain to compute it. Note that during MMD loss calculation, the source domain has already sent this information to the target domain, so the target domain does not need extra data from the source domain to compute this derivative.

To summarize, the source domain needs to send encrypted $\{f_m(v_i^s)\}_{m=1}^M$ of each feature vector v_i^s to the target domain, and the target domain needs to send encrypted $\{g_m(v_j^t)\}_{m=1}^M$ of each feature vector v_j^t to the source domain. Therefore, both domains can calculate their own MMD gradients, and the target domain can calculate the MMD loss.

4.2.3 Loss Monitoring and MMD Gradient Decryption

In our framework, we let the source domain monitor the total loss during training. Since the loss terms L_{Θ^t}, L_{p2} and L_{p3} are computed by the target domain, we let the target domain compute $\lambda(L_{p2} + L_{p3}) + \gamma_2 L_{\Theta^t}$ and send the result to the source domain. Note that this result is encrypted with the source domain's public key because the L_{p3} computed by the target domain is encrypted. Thus, the source domain needs to decrypt this result first and then add it to the remaining part of the total loss.

For parameter updates, note that the MMD gradients computed by both domains are encrypted with the other party's public key. Therefore, they need to send the MMD gradients to each other for decryption. However, they cannot directly send the raw gradients due to the potential security risk. Sending raw gradients means that one party will know the real gradients of the other party after decryption. However, some works have shown that information about training data can be inferred from gradients [26], [27], [28]. Therefore, in our framework, we let one party add a random mask to its gradients before sending them to the other party [7], [29]. Since the other party does not know the random mask, it cannot get the actual gradients. After decryption, the plaintexts are sent back, and both parties can subtract the mask to get gradient values for parameter updates.

When the fine-tuning phase finishes, the source domain will send the classifier weights to the target domain. The target domain can concatenate its feature extractor and the received classifier to get the final model.

4.3 Further Optimization

In our framework, the training participants may be edge devices or even mobile devices with limited computation and communication capabilities. Thus, we design two methods to further reduce the computation and communication costs during the training process.

4.3.1 Compression of Feature Vector Information

In the MMD loss and gradients calculation method described above, both domains need to send information about feature vectors to the other domain so that the source domain can calculate $\frac{\partial L_{p3}}{\partial \theta^s}$, and the target domain can calculate L_{p3} and $\frac{\partial L_{p3}}{\partial \theta^t}$. Specifically, for each feature vector v_i^s in the batch, the source domain needs to calculate the values of all monomials $f_m(v_i^s)$ and send their encrypted results to the target domain. Similarly, for each feature vector v_j^t, the target domain needs to send the encrypted results of all monomials $g_m(v_j^t)$ to the source domain. Suppose the number of monomials $f_m(v_i^s)$ and $g_m(v_j^t)$ is m_s and m_t respectively, and the batch size of the source and target domain is n_s and n_t respectively. Then for each batch, the source domain needs to compute and send $m_s \cdot n_s$ values, and the target domain needs to send $m_t \cdot n_t$ values.

In order to compress the feature vector information sent by both domains, we can leverage the mathematical characteristics of L_{p3}. Instead of calculating each kernel value $k(v_i^s, v_j^t)$ or its derivative $\frac{\partial k(v_i^s, v_j^t)}{\partial \theta}$ separately, we can consider L_{p3} as a whole, and compute its value and derivative using the following transformation:

$$\begin{aligned} L_{p3} &= -\frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t} k(v_i^s, v_j^t) \\ &= -\frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\sum_{m=1}^{M} c_m f_m(v_i^s) g_m(v_j^t) \\ &= -\frac{2}{n_s n_t}\sum_{m=1}^{M} c_m \cdot \left(\sum_{i=1}^{n_s} f_m(v_i^s)\right) \cdot \left(\sum_{j=1}^{n_t} g_m(v_j^t)\right), \end{aligned} \tag{11}$$

$$\frac{\partial L_{p3}}{\partial \theta^s} = -\frac{2}{n_s n_t}\sum_{m=1}^{M} c_m \cdot \left(\sum_{i=1}^{n_s} \frac{\partial f_m(v_i^s)}{\partial \theta^s}\right) \cdot \left(\sum_{j=1}^{n_t} g_m(v_j^t)\right), \tag{12}$$

$$\frac{\partial L_{p3}}{\partial \theta^t} = -\frac{2}{n_s n_t}\sum_{m=1}^{M} c_m \cdot \left(\sum_{i=1}^{n_s} f_m(v_i^s)\right) \cdot \left(\sum_{j=1}^{n_t} \frac{\partial g_m(v_j^t)}{\partial \theta^t}\right). \tag{13}$$

From the above transformation, we can see that for the target domain to calculate L_{p3} and $\frac{\partial L_{p3}}{\partial \theta^t}$, the source domain only needs to send $\sum_{i=1}^{n_s} f_m(v_i^s)$ for each monomial. For each monomial, the source domain can calculate the sum of all feature vectors and send the sum to the target domain. In this way, the number of values sent by the source domain is reduced to m_s. Similarly, for the target domain, it only needs to send $\sum_{j=1}^{n_t} g_m(v_j^t)$ for each monomial, and the number of values sent is reduced to m_t. Thus, it dramatically reduces the amount of feature vector information sent and encrypted by both domains in which way the computational overhead can be reduced.

4.3.2 Optimization Based on the Chain Rule

Currently, both domains directly calculate the gradient $\frac{\partial L_{p3}}{\partial \theta}$ for each parameter θ, using the feature vector information from the other party. Due to time-consuming homomorphic operations on ciphertext gradient calculation and large numbers of parameters, the current method will incur high computation and communication cost for both domains.

We leverage the chain rule in the derivative calculation to optimize this procedure. For the source domain, the gradient $\frac{\partial L_{p3}}{\partial \theta^s}$ can be written as follows, using the feature vectors as an intermediate stage:

$$\frac{\partial L_{p3}}{\partial \theta^s} = \sum_{i=1}^{n_s}\sum_{l=1}^{L} \frac{\partial L_{p3}}{\partial v_{il}^s} \cdot \frac{\partial v_{il}^s}{\partial \theta^s} + \sum_{j=1}^{n_t}\sum_{l=1}^{L} \frac{\partial L_{p3}}{\partial v_{jl}^t} \cdot \frac{\partial v_{jl}^t}{\partial \theta^s}, \tag{14}$$

where v_{il}^s denotes the lth element in the feature vector v_i^s. The second part in the sum can be omitted because v_{jl}^t is a feature vector element in the target domain and is not relevant to θ^s. For each parameter θ^s, the source domain can compute $\frac{\partial v_{il}^s}{\partial \theta^s}$ on its own. As for the calculation of $\frac{\partial L_{p3}}{\partial v_{il}^s}$, it needs information from the target domain. Now the source domain only needs to calculate $n_s \cdot L$ encrypted values because there are only $n_s \cdot L$ feature vector elements. This is typically much smaller than the number of source feature extractor's parameters $|\Theta_s|$. Moreover, now the source domain only needs to send $\frac{\partial L_{p3}}{\partial v_{il}^s}$ for each feature vector element v_{il}^s to the target domain for decryption, rather than $\frac{\partial L_{p3}}{\partial \theta^s}$ for each parameter θ^s. A similar analysis also applies to the target domain. Thus, this new chain rule-based method greatly reduces the computation and communication costs.

For a specific feature vector element $v_{i_0 l_0}^s$, the derivative $\frac{\partial L_{p3}}{\partial v_{i_0 l_0}^s}$ can be computed as follows:

$$\begin{aligned} \frac{\partial L_{p3}}{\partial v_{i_0 l_0}^s} &= -\frac{2}{n_s n_t}\sum_{m=1}^{M} c_m \cdot \left(\sum_{i=1}^{n_s} \frac{\partial f_m(v_i^s)}{\partial v_{i_0 l_0}^s}\right) \cdot \left(\sum_{j=1}^{n_t} g_m(v_j^t)\right) \\ &= -\frac{2}{n_s n_t}\sum_{m=1}^{M} c_m \cdot \frac{\partial f_m(v_{i_0}^s)}{\partial v_{i_0 l_0}^s} \cdot \left(\sum_{j=1}^{n_t} g_m(v_j^t)\right). \end{aligned} \tag{15}$$

From the above equation, we can see that the information it needs from the target domain is still $\sum_{j=1}^{n_t} g_m(v_j^t)$, which is the same as the direct gradient calculation. The same conclusion also holds for the target domain. That is, for the new chain rule-based method, the number and types of monomials sent from the source domain to the target domain and from the target domain to the source domain do not change compared to direct gradient calculation.

4.4 Extension to Multiple Users

Till now, we have only discussed the case where there is one participant with labeled data and one participant with unlabeled data. In reality, there may be multiple participants with labeled data, such that we need to extend the method to accommodate multiple source domain situations.

One approach is to regard all participants with labeled data as a single source domain. A possible solution for this scenario is incorporating Horizontal FL techniques into our framework. We can use a central server to coordinate all participants in the source domain and securely aggregate their information during training. For classification and regularization terms during the pretraining and fine-tuning, each participant in the source domain can compute their loss and gradients on their own data and securely send the results to the central server for aggregation. As for MMD loss and gradients, feature vector information exchange is needed before computation and aggregation. Here for L_{p_1} loss and gradients calculation, the participants in the source domain need to exchange information with each other. For L_{p_3} loss and gradients calculation, we need information exchange between the target user and each participant in the source domain. L_{p_2} is still handled by the target user alone. Suppose there are S labeled participants, each with batch size $n_{s_i}, i \in \{1, 2, ..., S\}$. The total number of samples of source labeled participants for one batch is $\sum_{i=1}^{S} n_{s_i} = N_s$. One unlabeled participant has a batch size n_t. The MMD loss is composed of three parts as shown in Equation (3), but L_{p_1} and L_{p_3} are recalculated as shown in Equations (16) and (17) respectively.

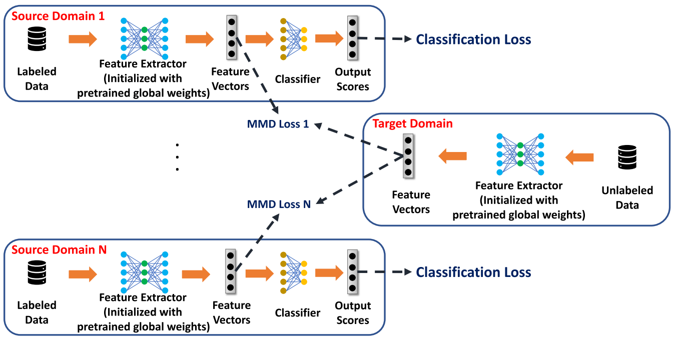


Fig. 3. A sketch of our federated multi-source domain adaptation framework (fine-tuning phase).

$$\begin{aligned} L_{p1} &= \frac{1}{N_s(N_s-1)} \sum_{i=1}^{N_s} \sum_{i'=1, i' \neq i}^{S} k(v_i^s, v_{i'}^s) \\ &= \frac{1}{N_s(N_s-1)} \left[\sum_{i=1}^{S} \sum_{j=1, j\neq i}^{S} \sum_{k=1}^{n_{s_i}} \sum_{k'=1}^{n_{s_j}} k(v_k^{s_i}, v_{k'}^{s_j}) \right. \\ &\left. + \sum_{i=1}^{S} \sum_{k=1}^{n_{s_i}} \sum_{k'=1, k'\neq k}^{n_{s_i}} k(v_k^{s_i}, v_{k'}^{s_i}) \right]. \end{aligned} \tag{16}$$

As shown in Equation (16), exchanges between different labeled participants bring high communication costs, and discrepancies exist between labeled data distributions. Thus we treat each labeled participant as a source domain and apply multi-source domain adaptation, which calculates MMD loss between each source and the target domain. Fig. 3 shows a sketch of our proposed federated multi-source domain adaptation.

The MMD loss is formulated as follows:

$$L_{MMD} = \frac{1}{S} \sum_{i=1}^{S} L_{p1}^{s_i} + L_{p2} + \frac{1}{S} \sum_{i=1}^{S} L_{p3}^{s_i}, \tag{18}$$

$$L_{p1}^{s_i} = \frac{1}{n_{s_i}(n_{s_i}-1)} \sum_{j=1}^{n_{s_i}} \sum_{j'=1, j'\neq j}^{n_{s_i}} k(v_j^s, v_{j'}^s), \tag{19}$$

$$L_{p3}^{s_i} = -\frac{2}{n_{s_i} n_t} \sum_{j=1}^{n_{s_i}} \sum_{k=1}^{n_t} k(v_j^{s_i}, v_k^t). \tag{20}$$

In this way, we relieve the communication costs of exchanging values among source domains. We show later that multi-source domain adaptation methods achieve even better accuracy than the single source domain adaptation method. Note that we also relieve the communication cost needed for source domains to aggregate their weights update resulting from classification and regularization loss in the fine-tuning phase. Since at the beginning of the fine-tuning phase, all the source domains' feature extractors adopt the same model weights they trained together in the pretraining phase. Then they can fine-tune their own feature extractors without sharing weights among each other. As each source domain classifier will output a classification result for the target data, we take the average as the final prediction result for the target domain.

4.5 Communication Cost Analysis

We summarize the relationship between the communication cost and corresponding parameters including source batch size n_s, target batch size n_t, number of Taylor expansion monomials M, number of source labeled users S and ciphertext size of Paillier encryption p. M is a function of latent feature length L and the highest degree of Taylor expansion n. Each labeled user needs to send $\sum_{i=1}^{n_s} f_m(v_i^s)$ of M monomials to the target domain, and the total bytes needed are $O(M \cdot p)$. To calculate gradients, each labeled user also needs to send $\frac{\partial f_m(v_i^s)}{\partial v_{il}^s}$ for each feature element of each sample thus the total bytes needed are $O(n_s \cdot L \cdot p)$. The labeled user receives encrypted gradients (with random mask) from the target domain and needs to send the decrypt values back to the target. Therefore, the labeled user also needs to send $O(n_t \cdot L)$ bytes. The total number of bytes a source labeled user sends for a batch is $O(M \cdot p + n_s \cdot L \cdot p + n_t \cdot L)$. Similarly, the targeted unlabeled user needs to send $O(M \cdot p + S \cdot n_t \cdot L \cdot p + S \cdot n_s \cdot L)$. The first term $M \cdot p$ is the encrypted sum monomials over all the feature vectors in the batch. As it is encrypted by the target domain's public key, it only needs to be sent once. The second term is for $\frac{\partial g_m(v_j^t)}{\partial v_{jl}^t}$. As they are encrypted by corresponding sources' public keys, they need to be sent S times. The last term is to send back the decrypted gradients (with random masks).

5 Experiment

We conduct experiments on three datasets, namely Widar 3.0 [30], Office [31] and PAMAP2 [32] to test the performance of our Federated Domain Adaptation framework. On the one hand, we want to compare the prediction performance of our framework with a centralized training approach, where the same UDA method is applied, but data from all the domains are known to the model trainer. On the other hand, we want to measure the enhancement made by our optimization methods in terms of training speed and communication cost.

5.1 Deployment

All experiments are emulated on the Amazon EC2 cluster to prototype the client-side model training as well as the cloud coordination using socket programming. We do not use real edge devices like mobile phones or tablets since the slow network connection as well as the limited computation capacity at the real edge-side devices will significantly slow down the training process. Also, as all the experiments are conducted under the same network conditions, and our statistics are multiples of improvements, the result should reflect the truth to a large extent. For each node, we used an m4.xlarge EC2 instance with 4 cores and 16 GB RAM.

5.2 Datasets

In this section, we briefly introduce the datasets used in our experiment.

Widar3.0. Widar3.0 [30] is a gesture recognition dataset based on Wi-Fi signals. It contains data for 22 types of gestures and 16 volunteers (not all gestures are performed by each volunteer). Body-coordinate Velocity Profile (BVP) is a novel representation proposed to reduce domain discrepancy. It represents the intensity of each velocity vector at the moment with a 20 by 20 matrix.

TABLE 2
Performance Comparison on Widar3.0 Dataset (%) (Values in Brackets Show the Difference Compared With the Multi-Source Centralized Method)

Cases	Centralized (multi-source)		Centralized (single-source)		Proposed (multi-source)	
	Balanced accuracy	Weighted F1	Balanced accuracy	Weighted F1	Balanced accuracy	Weighted F1
Room1 → Room2	39.29	40.05	39.86 (+0.57)	38.14 (-1.91)	38.95 (-0.34)	37.47 (-2.58)
Room2 → Room3	39.74	39.83	39.44 (-0.50)	39.24 (-0.59)	38.50 (-1.24)	39.67 (-0.16)
Room3 → Room1	43.14	39.73	45.04 (+1.90)	44.64 (+4.91)	44.83 (+1.69)	40.36 (+0.63)
Average	40.72	39.87	41.45 (+0.73)	40.67 (+0.80)	40.76 (+0.04)	39.17 (-0.70)

Office. Office [31] is a benchmark of domain adaptation. It contains images from three distinct domains: Amazon, DSLR, and Webcam. The 31 categories in the dataset consist of objects commonly encountered in office settings, such as keyboards, laptops, and the like.

PAMAP2. PAMAP2 [32] is an activity recognition dataset based on inertial and heart rate sensors. It collects data from 3 inertial sensors at the wrist, chest, and ankle. There are 9 volunteers following a given protocol containing 12 physical activities. Some of the volunteers also follow an optional protocol which includes another 6 activities. It provides raw sensor readings for each volunteer at 100Hz and the labels at each timestamp.

5.3 Prediction Performance

In this section, we evaluate the prediction performance of our framework compared with the centralized training approach. We use two metrics in our evaluation: *balanced accuracy* and *weighted F1 score*. Balanced accuracy is calculated as the average of recalls obtained in each class. Compared with regular accuracy, it better compensates for the class imbalance in the test set. Weighted F1 score is the weighted average of F1 score on each class, which is also used in the evaluation of other Federated Transfer Learning frameworks [7], [8]. We run all experiments 30 times and show the average results of the above metrics.

5.3.1 Performance on Widar3.0 Dataset

We pick the same six gestures: *Push & Pull*, *Sweep*, *Clap*, *Slide*, *Draw Circle*, and *Draw Zigzag*, as that of the original paper of the Widar3.0 dataset [30]. We use the same architecture as the original paper, except that we reduce the dimension of feature vectors from 128 to 4 to speed up our experiment, which comes with some sacrifice on the prediction accuracy. Note that it does not affect the comparison between our framework and the centralized training approach because they both use the same simplified network. In our experiments, we use three users in one room as the source domain and the other three users in another room as the target domain. We evaluate the performance of our framework and centralized training approach on three across room cases. We run the centralized training approach in two ways for each across room case: treat the three labeled users as three source domains and treat the three labeled users as a single domain. Our proposed FL method treats three labeled users as three source domains. The highest Taylor expansion degree is set to 2, the target batch size is set to 64, Paillier encryption ciphertext size is 1024 (we set this value as small as possible to speed up the training), and the weight for MMD loss is set to 0.25.

The results in Table 2 show that our Federated DA framework only has minor performance differences compared with the standard centralized learning approach on this dataset. Also, there is little difference between the centralized training approach with one single source domain and the centralized training approach with multiple source domains.

5.3.2 Performance on Office Dataset

For the Office dataset, we use the resnet18 network architecture with a bottleneck layer to extract 4-dimensional feature vectors. We evaluate our method across three transfer tasks commonly used for evaluation (Amazon → Webcam, DSLR → Webcam, and Webcam → DSLR). As there is only one source domain, we do not need to extend this to multiple source domain situations. The results are shown in Table 3.

From the results, we can see that our proposed method has similar results to the centralized method. Note that the reason our accuracy is much lower than that of the original paper is that the extracted feature length is only 4, which is much lower than the normal feature length (e.g., 256).

5.3.3 Performance on PAMAP2 Dataset

For the PAMAP2 dataset, we use the same data preprocessing pipeline as in [33]. We group the 12 activities into three classes as in [32] representing different activity intensities. As for the model architecture, we use two fully connected layers with 32 and 4 output units respectively, and a dropout layer with 0.5 dropout rate in between as the feature extractor, and one dropout layer with 0.5 drop out rate plus one fully connected layer with 3 output classes as the classifier. We use five users for evaluation on this dataset while the other four of the nine users in the dataset are not used due to incomplete data. We run experiments in the leave one user out manner. The results are summarized in Table 4.

TABLE 3
Performance Comparison on Office Dataset (%) (Values in Brackets Show the Difference Compared With the Centralized Method)

Case	Centralized (single-source)		Proposed (single-source)	
	Balanced accuracy	Weighted F1	Balanced accuracy	Weighted F1
A → W	27.42	24.08	25.53 (-1.89)	21.67 (-3.13)
W → D	48.59	37.85	46.39 (-2.20)	37.92 (+0.07)
D → W	32.92	26.61	31.48 (-1.44)	24.08 (-2.53)
Average	36.31	29.51	34.47 (-1.84)	27.89 (-1.62)

TABLE 4
Performance Comparison on PAMAP2 Dataset (%) (Values in Brackets Show the Differences Compared With the Multi-Source Centralized Method)

Target user	Centralized (multi-source)		Centralized (single-source)		Proposed (multi-source)	
	Balanced accuracy	Weighted F1	Balanced accuracy	Weighted F1	Balanced accuracy	Weighted F1
101	67.70	72.81	66.44 (-1.26)	70.52 (-2.29)	66.55 (-1.15)	71.51 (-1.30)
102	68.65	78.23	66.94 (-1.71)	73.25 (-4.98)	66.48 (-2.17)	71.44 (-6.79)
105	65.22	73.59	62.87 (-2.35)	69.75 (-3.84)	67.00 (+1.78)	75.20 (+1.61)
106	69.99	80.71	67.91 (-2.08)	75.02 (-5.69)	70.60 (+0.61)	81.38 (+0.67)
108	71.47	82.59	68.31 (-3.16)	76.32 (-6.27)	70.83 (-0.64)	81.93 (-0.66)
Average	68.61	77.59	66.49 (-2.12)	72.95 (-4.63)	68.68 (-0.32)	76.30 (-1.29)

From the result, we can see that the multi-source centralized training approach has higher performance compared with single-source centralized training. Thus it is better to treat multi-users as multi-source domains in our federated approach. Our Federated Domain Adaptation framework has similar performance as centralized training with multi-source domains.

5.4 Effect of Optimization Methods

In this section, we evaluate the enhancement of running speed and communication cost resulting from our optimization methods. We run two versions of our framework, one with the optimization and one without, on all three datasets. As seen in the last section, treating all the labeled users as one source and treating them as multiple sources achieves similar balanced accuracy and weighted F1 score. In this section, we also compare the communication cost and running speed of these two versions on PAMAP2 and Widar 3.0 datasets which have multiple source labeled users. Note that we only compare communication costs related to MMD loss and gradients computation without considering aggregation cost when we treat source domains as a single domain. For Widar 3.0 and the PAMAP2 dataset, we use the same model architecture described in previous experiments for both versions. For the Office dataset, we simplify the Resnet18 model and only keep the first layer block to reduce the running time of the naive method. Tables 5 and 6 show the running time and the amount of data sent for each batch averaged over multiple batches. The values in the table reflect the average of all the cases on each dataset. All experiments are conducted with a batch size of 64 for the target domain. The highest degree for Taylor expansion is 2 for the three datasets. The Paillier ciphertext length is 1024 for the Widar3.0 dataset and 512 for the other 2 datasets.

The running time in our framework is mainly determined by the model complexity, the number of cryptographic operations including encryptions, decryptions, and homomorphic additions, and also the key size for the Paillier scheme. We choose a relatively small key size to speed up our experiments. From the results in Table 5, we can see that the optimization methods result in an improvement of $115.38\times$, $292.71\times$ and $9.45\times$ for running time on the three datasets, respectively. This is due to the fact that our optimization methods significantly reduce the number of cryptographic operations. Note that for more complex model architecture, greater improvements can be achieved.

As for the communication cost during a batch, we can see from Table 6 that the amount of data sent by the target domain is smaller than the source domain if there is only one source domain (on Office dataset). The first case is where the source domain has more data and thus has a larger batch size. The other reason is that in our framework, the target domain needs to calculate the loss value L_{p3}, but the source domain only needs to calculate the gradient of L_{p3}. In the approximated kernel function, the monomial $f_m(v_i^s)$ and $g_m(v_j^t)$ can be a constant number 1. When $g_{m_0}(v_j^t)$ is 1, the source domain still needs to send the corresponding monomial $f_{m_0}(v_i^s)$ to the target domain for loss calculation. However, when $f_{m_0}(v_i^s)$ is 1, its derivative is 0 when calculating gradients, so there is no need for the target domain to send corresponding $g_{m_0}(v_j^t)$ to the source domain. Thus the target needs to send less data than the source.

In terms of the total amount of communication, the optimized version shows an improvement of $160.46\times$, $442.75\times$ and $24.65\times$ on three datasets, respectively. Note that the degree of improvement on PAMAP2 is not as significant as the other two due to fewer parameters in its model.

TABLE 5
The Running Time for One Batch on Different Datasets (In Seconds)

	w/ optimization (multi-source)	w / optimization (single-source)	w/o optimization
Widar3.0	10.86	14.63 (1.35×)	1253 (115.38×)
Office	13.23	/	3873.51 (292.71×)
PAMAP2	9.45	11.0 (1.17×)	89.24 (9.45×)

The numbers in the brackets show the improvement.

TABLE 6
The Amount of Data Sent for One Batch on Different Datasets (In KB)

Method		Widar 3.0	Office	PAMAP2
w/ optimization (multi-source)	source domain	253.8	129097.2	100.4
	target domain	527.1	111934.7	361.1
	total	780.9	241031.9	461.5
w/ optimization (single-source)	source domain	1261.4	/	1151.2
	target domain	529.2	/	367.4
	total	1790.6 (2.29×)	/	1518.6 (3.29×)
w/o optimization	source domain	62663.5	53528178.0	6897.0
	target domain	62638.7	53188133.6	4478.4
	total	125302.2 (160.46×)	106716311.7 (442.75×)	11375.4 (24.65×)

The numbers in the brackets show the improvement.

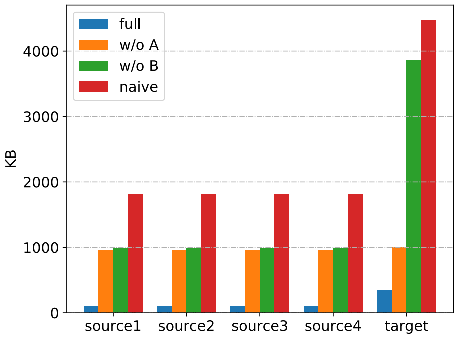


Fig. 4. Ablation study for optimization methods.

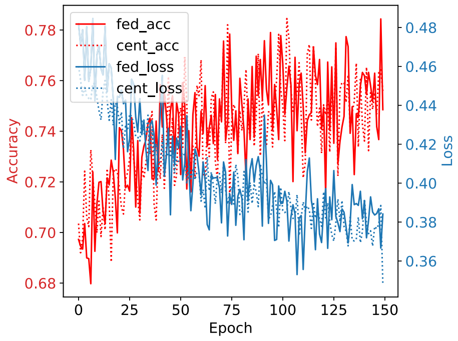


Fig. 5. Convergence.

In summary, our optimization methods can significantly improve the running speed and communication cost during the model training by up to two orders of magnitude.

5.5 Ablation Study

In this section, we show the contribution of our two proposed optimization methods, namely compression of feature vector information and optimization based on the chain rule. We evaluate on the PAMAP2 dataset where user 108 is the target and the other four users are the sources. The result is shown in Fig. 4. We can see that both optimization methods have a significant impact on the communication cost. Optimization method B is more effective for the target domain since the target needs to encrypt the gradients with the corresponding labeled user's public key and send it to the server. As there are multiple labeled users, this step saves more communication resources with optimization method B.

5.6 Convergence Analysis

We compare the loss-epoch curve and accuracy-epoch curve of the centralized training method and federated training method. As shown in Fig. 5, the blue lines represent the loss of the federated method and centralized method, respectively, and the red lines represent the accuracy of the federated method and centralized method, respectively. We can observe that our federated training method finally converges, and the convergence speed is similar to the centralized method as the lines overlap.

5.7 Parameter Study

In this section, we evaluate the influence of different parameters, including the number of labeled users, Taylor expansion highest degree, and Paillier encryption ciphertext size.

5.7.1 Number of Labeled Users

To evaluate the impact of the number of labeled users, we gradually increase the number of labeled users from 1 to 4, with user 108 as the target. We draw the balanced accuracies, weighted F1 scores, communication and computation cost improvements which are the quotients of corresponding values of methods without the optimization and proposed method under 4 cases in Fig. 6a. We can see that the improvements are stable with different numbers of users, and the balanced accuracies and weighted F1 scores increase with the number of labeled users.

5.7.2 Taylor Expansion Highest Degree

Taylor expansion highest degree is an important factor that may affect communication cost and accuracy. This value is k in $exp(x) = \sum_{k=0}^{K} \frac{x^k}{k!}$. We evaluate the value of Taylor expansion's highest degree ranging from 1 to 3 and calculate the accuracy and time for one batch. We show the result in Fig. 6b. We test in two cases, case 1: the source domains are user 101, 102, 105, and 106, the target domain is user 108 in the PAMAP2 dataset and case 2: the source domains are user 14, 15, and 16 in room 1, the target domain is users in room 2 in the Widar 3.0 dataset. The results are consistent. The performances are similar when we take different values of the highest degree, but the time used increases sharply. Thus, we take it that the Taylor degree as 1 is enough for a good performance. The reason is that our kernel function is $exp(-\alpha\|v_i - v_j\|^2)$, when we substitute x in $exp(x) = \sum_{k=0}^{K} \frac{x^k}{k!}$ with $-\alpha\|v_i - v_j\|^2$ and take $k = 1$, the highest order of element of v_i is actually 2. The higher orders may not contribute much.

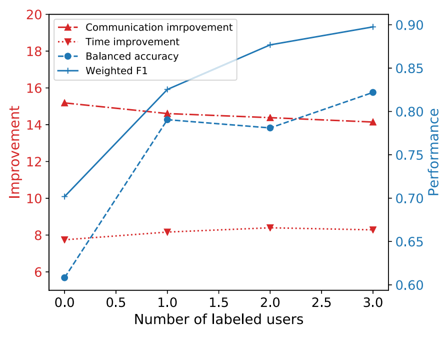
5.7.3 Paillier Encryption Ciphertext Size

Paillier encryption ciphertext size determines the security level and influences the communication cost. In the experiments in previous sections, we set this value as small as possible to speed up training. Here we set different Paillier ciphertext sizes as values in $\{512, 1024, 2048\}$ and compare their communication costs. As we can see from Fig. 6c, the communication cost is proportional to the ciphertext size. Therefore, if we enlarge the ciphertext size, our method can save more communication consumption compared with the naive method and becomes more secure.

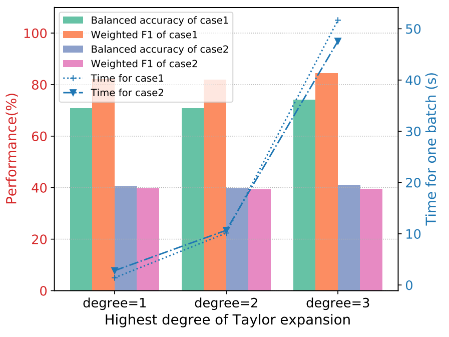
6 Discussion

6.1 Security Analysis

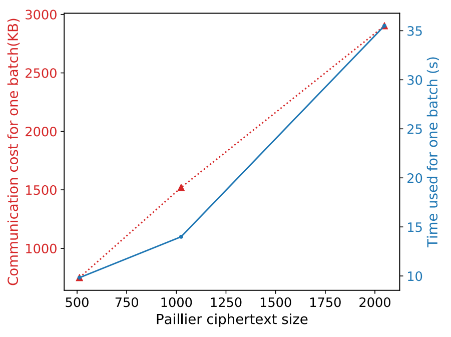
The same security definition and analysis for the Federated Transfer Learning framework in [7] applies to our framework. The only difference is that we additionally require the source domain to send pretrained feature extractor weights and final classifier weights to the target domain. Some research works show that exposing model weights can leak information about training data. However, these attacks [34], [35], [36], [37] do not work given the weights disclosed in our system since they all require access to the full model. In our system, only part of the pretrained model and part of the final model of the source domain are disclosed. Thus these two attacks are ineffective.



(a) Number of labeled users



(b) Taylor order



(c) Paillier ciphertext size

Fig. 6. Parameter study.

Other types of attacks based on model weights also exist, but they are not effective in our system either. The method proposed in [38] targets the Horizontal FL scenario. The attacker participates in the training procedure and can observe the global weight update at each round. The weights are shared among participants. In contrast, our system is a Federated Transfer Learning framework. Suppose that the target user is an attacker. Then firstly, it does not participate in the pretraining phase. Although it participates in the fine-tuning phase, the model weights of all the domains are not shared. Thus, it cannot get information about the training data of the source domain using this method.

To summarize, our framework is as secure as previous Federated Transfer Learning frameworks.

7 Conclusion

In this paper, we design a Federated Domain Adaptation framework. We use the Taylor Series to substitute the exponential functions in MMD loss to make Homomorphic Encryption applicable and design a scheme for both domains to calculate the MMD loss and gradients collaboratively. Moreover, we propose two optimization methods to further improve our framework's computation and communication efficiency. We conduct extensive evaluations on three datasets. The results show that our framework achieves comparable performance with a standard centralized training approach, and the optimization methods can reduce the overhead by up to two orders of magnitude.

Acknowledgments

Hua Kang and Zhiyang Li contributed equally to this work.

References

[1] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. UbiComp Adjunct*, 2016, pp. 185–188, doi: 10.1145/2968219.2971461.

[2] H. Zhang, C. Song, A. Wang, C. Xu, D. Li, and W. Xu, "PDVocal: Towards privacy-preserving parkinson's disease detection using non-speech body sounds," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, pp. 16:1–16:16, doi: 10.1145/3300061.3300125.

[3] W. Jiang *et al.*, "Towards environment independent device free human activity recognition," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 289–304, doi: 10.1145/3241539.3241548.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[6] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018, doi: 10.1016/j.neucom.2018.05.083.

[7] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1812.03337

[8] S. Sharma, C. Xing, Y. Liu, and Y. Kang, "Secure and efficient federated transfer learning," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 2569–2576, doi: 10.1109/BigData47090.2019.9006280.

[9] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in *Proc. 8th Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJezF3VYPB

[10] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," in *Proc. 13th Pacific Rim Int. Conf. Artif. Intell.*, 2014, pp. 898–904, doi: 10.1007/978-3-319-13560-1_76.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238, doi: 10.1007/3–540-48910-X_16.

[12] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Annu. Conf. Neural Informat. Process. Syst.*, 2017, pp. 4424–4434. [Online]. Available: http://papers.nips.cc/paper/7029-federated-multi-task-learning

[13] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582

[14] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," *CoRR*, 2020. [Online]. Available: https://arxiv.org/abs/2004.11791

[15] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1802.07876

[16] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.

[17] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1610.05492

[18] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 954–964, doi: 10.1109/ICDCS.2019.00099.

[19] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, 2017. [Online]. Available: http://arxiv.org/abs/1711.10677

[20] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1412.3474

[21] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 97–105. [Online]. Available: http://proceedings.mlr.press/v37/long15.html

[22] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2208–2217. [Online]. Available: http://proceedings.mlr.press/v70/long17a.html

[23] X. Zhang, F. X. Yu, S. Chang, and S. Wang, "Deep transfer network: Unsupervised domain adaptation," *CoRR*, 2015, *arXiv:1503.00591*

[24] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo, "Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 945–954, doi: 10.1109/CVPR.2017.107.

[25] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 801–814, Apr. 2019, doi: 10.1109/TPAMI.2018.2814042.

[26] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Informat. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018. [Online]. Available: https://doi.org/10.1109/TIFS.2017.2787987

[27] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Annu. Conf. Neural Informat. Process. Syst.*, 2019, pp. 14 747–14 756. [Online]. Available: http://papers.nips.cc/paper/9617-deep-leakage-from-gradients

[28] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," *CoRR*, 2020, *arXiv:2001.02610*

[29] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019, doi: 10.1145/3298981.

[30] Y. Zheng *et al.*, "Zero-effort cross-domain gesture recognition with Wi-Fi," in *Proc. 17th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2019, pp. 313–325. doi: 10.1145/3307334.3326081.

[31] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Providence*, 2012, pp. 2066–2073, doi: 10.1109/CVPR.2012.6247911.

[32] A. Reiss and D. Stricker, "Creating and benchmarking a new dataset for physical activity monitoring," in *Proc. 5th Int. Conf. Pervasive Technol. Related Assistive Environments*, 2012, Art. no. 40, doi: 10.1145/2413097.2413148.

[33] R. Adaimi and E. Thomaz, "Leveraging active learning and conditional mutual information to minimize data annotation in human activity recognition," *Proc. ACM InterAct. Mobile Wearable Ubiquitous Technol.*, vol. 3, no. 3, pp. 70:1–70:23, 2019, doi: 10.1145/3351228.

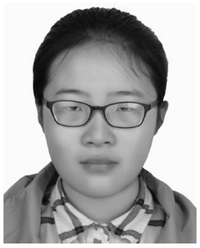
[34] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. 23rd USENIX Secur. Symp.*, 2014, pp. 17–32. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/fredrikson_matthew

[35] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333, doi: 10.1145/2810103.2813677.

[36] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18, doi: 10.1109/SP.2017.41.

[37] 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 *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753, doi: 10.1109/SP.2019.00065.

[38] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618, doi: 10.1145/3133956.3134012.



Hua Kang received the bachelor's degree in transportation engineering from Tongji University, Shanghai, China, in July 2018. She is currently working toward the PhD degree with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. Her research interests include Internet of Things, wireless sensing, and deep learning.

Zhiyang Li received the bachelor's degree in infomation security from Wuhan University, in 2017, and the MPhil degree in computer science and engineering from the Hong Kong University of Science and Technology, in 2020. He is currently working with ByteDance Inc. His research interests include wireless sensing and transfer learning.

Qian Zhang (Fellow, IEEE) received the BS, MS, and PhD degrees in computer science from Wuhan University, China, in 1994, 1996, and 1999, respectively. In 2005, she joined the Hong Kong University of Science and Technology, where she is currently a tencent professor in engineering and the chair professor with the Department of Computer Science and Engineering. She is also serving as the co-director of Huawei-HKUST Innovation Lab and the director of Digital Life Research Center, HKUST. Before that, she was with Microsoft Research Asia, from July 1999, where she was the research manager of the Wireless and Networking Group. Her current research interests include Internet of Things, smart health, mobile computing and sensing, wireless networking, as well as cyber security. She is currently serving as the editor-in-chief for *IEEE Transactions on Mobile Computing*. She was a members-at-Large of IEEE Communications Society from 2016 to 2018.