Vol. 06, No. 2 (2024) 601-612, doi: 10.24874/PES06.02.017

Proceedings on Engineering Sciences

www.pesjournal.net

FEDEMB: A VERTICAL AND HYBRID FEDERATED LEARNING ALGORITHM USING NETWORK AND FEATURE EMBEDDING AGGREGATION

Fanfei Meng[1]
Lele Zhang
YuChen
Yuxin Wang

Received 14.06.2023.
Received in revised form 07.10.2023.
Accepted 22.11.2023.
UDC – 004.85

Keywords:

Federated Learning, Differential Privacy, Vertical and Hybrid Federated Learning

A B S T R A C T

Federated learning (FL) is an emerging paradigm for decentralized training of machine learning models on distributed clients, without revealing the data to the central server. The learning scheme may be horizontal, vertical or hybrid (both vertical and horizontal). Most existing research work with deep neural network (DNN) modeling is focused on horizontal data distributions, while vertical and hybrid schemes are much less studied. In this paper, we propose a generalized algorithm FedEmb, for modeling vertical and hybrid DNN-based learning. The idea of our algorithm is characterized by higher inference accuracy, stronger privacy-preserving properties, and lower client-server communication bandwidth demands as compared with existing work. The experimental results show that FedEmb is an effective method to tackle both split feature & subject space decentralized problems. To be specific, there are 0.3% to 4.2% improvement on inference accuracy and 88.9 % time complexity reduction over baseline method.

1. INTRODUCTION

Federated learning stands in contrast to traditional centralized machine learning techniques where all the client datasets are uploaded to a central server (Su et al, 2023; Kalra et al., 20203) using privacy-preserving methods. In terms of another driver for federated learning, besides confidentiality considerations, is the abundance of datasets - the mandatory size of sample sets, or dense representations to ensure learning effects - then focus on implementing the advanced and efficient small set machine learning approaches for Federated Learning systems & paradigms. In Horizontal Federated Learning (HFL) clients shares the same feature space but differ in the samples selected. FedAvg was introduced (McMahn et al., 2017; Wu et al., 2023) by as the first HFL algorithm to train DNN models with decentralized and confidential datasets. Following this work (Wang et al., 2023), developed FedMat with weight matching average, and (Li et al., 2020) developed FedProx for heterogeneous networks when the data are non-iid.

[1] Corresponding author: Fanfei Meng
Email: fanfeimeng2023@u.northwestern.edu

Vertical Federated Learning (VFL), introduced in (Hardy et al., 2017), aggregates different features to build models using samples with the same IDs. In daily life, VFL is widely investigated in collaborative financial and hospitality systems, where different banks & hospitals own unique information for same users & patients. Whereas in HFL DNN weights are averaged, in VFL gradients and feature embedding vectors are normally exchanged among all parties. Existing work in VFL leaves the following challenges. Encryption on gradient and feature embedding vectors is required for privacy-preserving on the client side. At the same time, decryption in server is necessary for aggregated computing; this process is inherently of high time complexity. On the other hand, the classical encrypting methods such as homomorphic encryption (Zhang et al., 2018; Zhang and Zhu, 2020) come with high computation costs when data is converted to a paillier tensor (Zhao and Geng, 2019; Acar et al., 2018).

Hybrid Federated Learning (HBFL) combines both configurations of HFL and VFL (Figure 1), where data do not share the same feature spaces or the same sample IDs. HBFL has more extensive application scenarios, when distributed clients have overlapping or non-overlapping user ID or features. For instance, the mobile payment institute and traditional branch bank have partial common customer and partial similar user features. Meanwhile, their tracking databases also store variant users and unique features due to different business models and target groups. In this case, HBFL is a very good algorithms to tackle complex data spaces. For HBFL most work is based on linear models (Gao et al., 2019; Sharma et al., 2019; Yang et al., 2021) and we are not aware of any work on DNN modeling in this space. A typical problem in the HBFL scheme for DNN is that a client may possess only a subset of sample IDs and a subset of features, that are not totally disjoint with data held by other clients. Furthermore, the data is likely to be incomplete with respect to the centralized setting. In the VFL scheme all of the clients need to formulate a batch of samples with the same IDs; this problem is exacerbated in the HBFL scheme because not all the clients have all samples. An ideal algorithm shall work without requiring the clients to synchronize their sample draws.

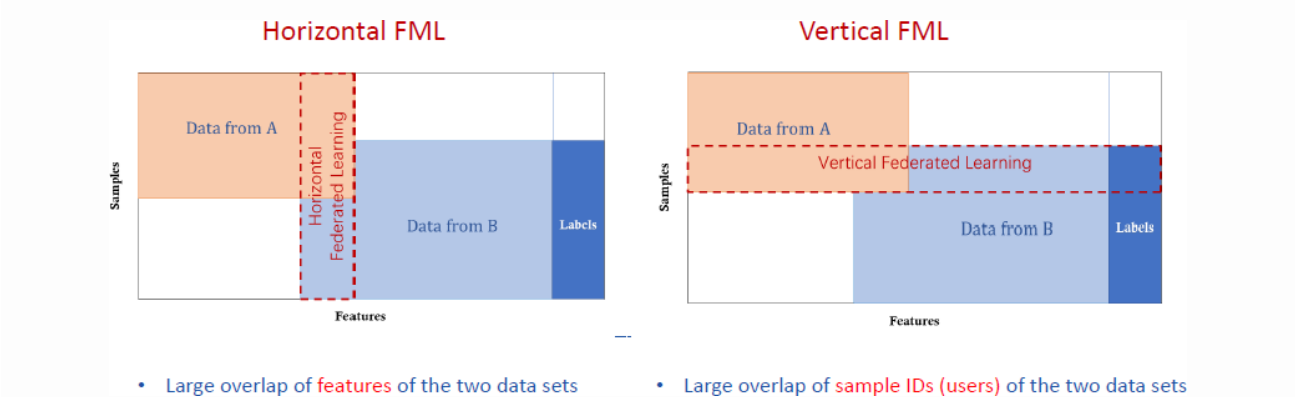

Figure 1. Horizontal and Vertical Data Distribution Visualization: HFL owns different samples, but VFL owns different features (Yang et al., 2019).

Motivated by incremental learning (Wu et al., 2019), we propose the use of partial network embedding and tuning to address both VFL and HBFL schemes in our algorithm, FedEmb. Generally, partial networks from different clients possibly stores disjoint or overlapping features of their own data, which cannot be directly averaged. To address this, we make the server vertically combine all partial networks to enable communication between different feature spaces using concatenated feature embedding vectors, thus acquiring knowledge of the whole data distribution incrementally during the global rounds. The feature embedding vectors are intermediate representations in partial networks at clients with privacy-preserving, so confidentiality of the datasets is guaranteed. Our contributions are as follows:

- To the best of our knowledge, this is the first investigation addressing HBFL for both i.i.d and non-i.i.d cases with DNN modelling. There is no need for sample ID synchronization and alignment for server processing.
- FedEmb is completely free of gradient exchange that most works normally deploy for training DNN in VFL or HBFL, which is highly questionable due to deep leakage (Zhu et al., 2019) and of high time complexity to complete a round of network updates of all clients. We only pass weights of partial networks of clients and aggregation-level feature embeddings, and clients are able to conduct model training independently, which significantly lowers communication costs.
- FedEmb has a higher level of privacy protection. In clients, we protect feature embedding vectors in two ways: vector aggregation and differential privacy, thereby precluding the need for decryption in the server.
- Our method has low bandwidth cost for communications between the server and clients. Instead of a full model or complete feature embedding vector exchange, we

aggregate vectors via clustering and the partial network is distributed.

In the paper, we organize our investigations in the following orders: Section 1 introduces most recent work about horizontal federated learning, vertical federated learning, hybrid federated learning (federated transfer learning) and differential privacy for privacy-preserving manners. Then Section 2 illustrates data partition setting in vertical federated learning and hybrid federated learning and their learning paradigms. Concrete differential privacy protection applied to feature embedding vectors are demonstrated in the end. Finally, a series of experiments, including two baseline methods (centralized learning and FATE for vertical federated learning) are presented to validate the performances and benefits of FedEmb. Conclusion Sections centralizes on summarizing the whole article and proposes possible future works.

2. RELATED WORK

With the improvement of computer computing power, machine learning, as an analysis and processing technology for massive data, has widely served human society. However, the development of machine learning technology faces two major challenges: firstly, data security is difficult to guarantee, and the problem of privacy data leakage needs to be solved urgently; secondly, network security isolation and industry privacy, there are data barriers between different industries and departments, resulting in data The formation of "isolated islands" cannot be safely shared, and the performance of the machine learning model trained only on the independent data of each department cannot achieve global optimization. In the following paragraphs, we demonstrate three main federated learning paradigm to address the data security and decentralization.

2.1 Horizontal Federated Learning (HFL)

With horizontal data, subjects & IDs are available with a consistent set of features. This is exactly the type of data fee into a supervised machine learning task. Horizontal federated learning is suitable for the situation where feature space of each ID is almost or completely overlapped, but the sample ID overlaps less. For example, the customer data of two banks in different regions. The word "horizontal" comes from the "horizontal partitioning, a.k.a. sharding" of data. FedAvg is the classical method where the server averages well-trained local weights and distributes back global models to clients in every round. However, the method is less competitive when the data in the clients is non-i.i.d. To address this issue, demonstrate FedMat to conduct neuron matching and alignment using Bayesian optimization to improve inference accuracy. In terms of the limitation that local models should be well-trained before passing in FedAvg, FedProx (Li et al., 2020) is proposed where weights can be exchanged with the serve without fine-tuning. In (Yang et al., 2021; Sidahmed et al., 2021) the authors discuss the learning effects on averaging partial networks of clients to reduce the computational complexity in the server. However, their algorithms can only be applied for HFL, and not VFL or HBFL, with the same architectures for exchanging weights.

2.2 Vertical Federated Learning (VFL)

FL with data split by features, is a unique learning paradigm in relative to the scheme that data is partitioned by IDs. It is optional to introduce a central server as a the third-party for processing, as a result, it requires clients to share presentations with privacy-preserving manners instead of network & algorithm parameters as exchanges, to enable full gradient calculations. Most current work is centralized on the linear model, especially linear regression and logistic regression. The concept and algorithms for VFL were first proposed by (Hardy et al., 2020), where a federated logistic regression scheme is employed using homomorphic encryption. Subsequent to this work, (Hu et al., 2019a; Kang et al., 2020; Hu et al., 2019b; Wang et al., 2020) introduce the gradient-free VFL paradigm for logistic regression models. In the context of DNN modelling, Heterogeneous Federated Learning using Homomorphic Encryption developed by (Zhang et al., 2018; Zhang and Zhu, 2020) is the most classical DNN-based VFL. The method exchanges encrypted feature embedding vectors from the client output layer, then back-propagate gradients from the server to update the models in the clients. However, this approach has high time complexity for communication and computing, the latter due to the encryption-decryption process. Secondly, the linear summation operation conducted by the server on feature embedding vectors from different clients is unable to fully represent disjoint feature spaces, thus impacting the training. On the other hands, there are some literature discuss the potential of utilizing differential privacy to vertical federated learning as well (Errounda et al., 2023). discusses the feasibility and effectiveness of dynamic differential privacy implemented on model parameters and feature presentations (Li et al., 2023). put their focus on feature selections, which are most representative in each client and then combine together for further learning and communications.

Table 1. Comparison between paradigms: ✓means the paradigm shares the named space among clients & has existing work. Otherwise, X represents no named space sharing among clients & no previous work

Paradigm	Subject	Feature	DNN Modelling
Centralized	✓	✓	✓
HFL	X	✓	✓
VFL	✓	X	✓
HBFL	X	X	X

2.3 Hybrid Federated Learning (HBFL)

In hybrid federated learning, also referred to as federated transfer learning, the sample IDs and feature spaces are disjoint. A HBFL setting is similar to HFL in that clients do not share their local data or labels (Gao et al., 2019;). Current work in HBFL is primarily focused on linear models as VFL, for example (Gao et al., 2019; Zhang et al., 2022; Yang et al., 2020). Their methods are either constrained by specific scenarios or hard to be generalized for DNN with large learnable parameter sets. We did not find any work with the HBFL scheme using DNN. In edge-device learning network or resource-restricted systems (Guo et al., 2023; Zhang et al., 2023; Qi et al., 2023), HBFL is widely investigated due to the natural fact that neither ID nor features sharing in a set of small data collectors. To be specific, mobile devices are usually power-limited or storage-limited, which leads to constrained computing & data transferring in user & client communication with confidentiality. These works all discuss linear model applications such as logistic regression, tree-based learning or shallow vector machine algorithms, which is weak in representing higher-level information and data spaces. As mentioned previously, the scheme of HBFL naturally miss more information than HFL and VFL. Consequently, the shallow feature representation captured by simple models severely shorten the aggregation learning performances.

2.4 Differential Privacy (DP)

DP is a system for disclosing information about a dataset by maintaining feature distributions about entities in the dataset (Dwork and Roth, 2014; Friedman and Schuster, 2010; McSherry and Talwar, 2007) while preserving privacy (Waserman and Zhou, 2010; Dwork and Lei, 2009). The main objective is that if the effect of making an arbitrary single substitution in the database is small enough, the query result cannot be used to infer information about any single individual or unit, and therefore provides privacy. Achieving differential privacy typically requires some form of data transformation. The three main categories of data transformation methods are: generalization through altering multiple individual feature values to the same value (Nissim and Stemmer, 2015; Dwork et al., 2010), suppression by removing attributes (Terrorvitis et al., 2017; Chen et al., 2013) and perturbation using noise addition such as the Laplace method (Xiao and Xiong, 2015; Nozari et al., 2016). We apply generalization and perturbation to feature embedding vectors for privacy preservation, which can be directly utilized for server tuning without significant accuracy loss.

3. ALGORITHM

In high level, our algorithms aggregate encrypted feature embedding vectors from all clients and partial neural network weights in server to do fine-tuning. The split vectors are vertically concatenated to represent the whole data spaces, similarly, the split weights are vertically concatenated to demonstrate distributed learning outcomes. In this section, we first introduce the data partitions distributed across all clients for the VFL and HBFL schemes. We then present the learning paradigm of FedEmb. Finally, the methods on privacy enhancement are illustrated for protecting feature embedding vectors.

3.1 Data Partition

Assume that there are N samples denoted as $A = \{\mathbf{x}_n, y_n\}_n^N$ where each sample xn has D features: $\mathbf{x}[1]_n; \ldots; \mathbf{x}[\mathcal{D}]_n]$ and yn is the corresponding label. There are M clients in total. Index the features by $d = 1, \ldots, \mathcal{D}$ sample data by n=1,...,N, and clients by m=1,...,M $m = 1, \ldots, M$ If the scheme is VFL then each clients possess all N samples with partial feature space, and the m-th client dataset can be formulated as $A_m = \{[\mathbf{x}[d_1]; \ldots; \mathbf{x}[d_m]]_n^{d_1,\ldots,d_m \in \mathcal{D}_m}, y_n\}_n^N$. For HBFL, where each client has some samples and their partial features, and if the m-th client only has the Nm samples out of all N samples, we can denote the local set as $A_m = \{[\mathbf{x}[d_1]; \ldots; \mathbf{x}[d_m]]_{n_m}^{d_1,\ldots,d_m \in \mathcal{D}_m}, y_{n_m}\}_{n_m}^{N_m}$. Dm is the set of features available to the m-th client. In our learning setting, both xn, yn are **NOT** exchanged with the server. For HBFL, any dm can either repetitively exist in different clients (joint feature sets) or not (disjoint feature sets).

3.2 Learning Paradigm

For the m-th client $m = 1, \ldots, M$, we denote $g_m(\mathbf{x}_{F_m})$ as the network function for local DNN, where $\mathbf{x}_{F_m} \in \mathbb{R}^D$ is an input sample from a local dataset A_m, $F_m \subseteq D$ There are NM samples in total ($N_M = N$ if the scheme is VFL). We define two parts of the whole network: the private partial network is $f_m^L(\mathbf{x}_{F_m}; \tau_m^L) \in \mathbb{R}^{u_m}$, and the public partial network is $f_m^S(l_m; \gamma_m^S) \in \mathbb{R}^{k_m}$, where u_m, k_m are the respective layer dimensions for the two networks. L denotes the first network and S denotes the second network, and τ_m^L and γ_m^S are the respective output embedding feature vectors of the two networks. The entire network is given by:

$$g_m(\mathbf{x}_{F_m}) = f_m^S(f_m^L(\mathbf{x}_{F_m}; \tau_m^L), \gamma_m^S).$$

We denote $h(\mathbf{x}_{F_m}^S) = f_m^S$ as the shareable network, and the network of the server can be represented as:

$$u(\mathbf{x}^M) = f([h(\mathbf{x}_{F_1}^S); \ldots; h(\mathbf{x}_{F_M}^S); \alpha_j^S]),$$

where j is the index of the global round, α_j^S is the connecting weight at round j for mitigating overfitting, and $h(\mathbf{x}_{F_1}^S); \ldots; h(\mathbf{x}_{F_M}^S)$ are trainable weights in u. After the local training is completed, we pass $h(\mathbf{x}_{F_M}^S) \in \mathbb{R}^{k_m}$ and the aggregation-level feature embedding vector set

$l_m^L = \{f_m^L(\mathbf{x}_{F_m}; \tau_m^L) | \mathbf{x}_{F_m} \in A_m\}$ and $\theta_m^S = \{g_m(\mathbf{x}_{F_m})) | \mathbf{x}_{F_m} \in A_m\}$ to the server. For vector set pairs (l_m^L, γ_m^L) from all M clients, they formulate the new embedding vector set $(l^L \in \mathbb{R}^{u_1+\ldots+u_M}, \theta^S \in \mathbb{R}^{k_m})$, where $l^L = [l_1^L; \ldots; l_M^L], \theta^S = \sum_{m=1}^{M} (\theta_1^S + \ldots + \theta_M^S)$.

l^L works as the input set for server network fine-tuning, and θ^S is the supervising ground truth set. Similarly, the networks passed from the clients are also vertically embedded correspondingly. We do the fine-tuning on using two cohesive sets of vectors l^L and θ^S as shown in Equation and then distribute back updated f_m^S to the m-th client correspondingly.

local training in parallel: Train local network g_m using local dataset $A_m \backslash l_m^L = \{f_m^L(\mathbf{x}_{F_m}; \tau_i^L) | \mathbf{x}_{F_m} \in A_m\}$ with privacy protections. Pass l_m^L θ_m^S and f_m^S to the server, **server tuning**: Formulate θ^S, l^L, Let the updated weights be notated by NEW: $\alpha_j^S = \alpha^{S,NEW}$ Distribute back $h(\mathbf{x}_{F_m}^S)^{NEW}$ to corresponding client m.

The learning paradigm is summarized in Algorithm 1 and visualized in Figure 2. It is obvious that the local training could be in parallel and server tuning is in one timestamp; as a consequence, the time complexity $\mathcal{O}(m) = 2$ holds. l_m^L are intermediate feature representations and θ_m^S are the predicting representations of DNN, where the linear summed outputs θ^S approximate predictions of concatenated inputs l^L. These vectors represent the data space owned by the corresponding clients, which could be incrementally communicated and learnt by the fused network in every round of server interactions, thereby preventing the network f_m^S from catastrophically forgetting after subsequent local training. To this end, the local data and labels are non-revealing after applying privacy protection measures. The reason for conveying f_m^S without the last few layers is due to the fact that the bottom layers are usually over-parameterized relative to the top layers, where parameter spaces are shallow (Du et al., 2018; Li and Liang, 2018; Allen-Zhu et al., 2019).

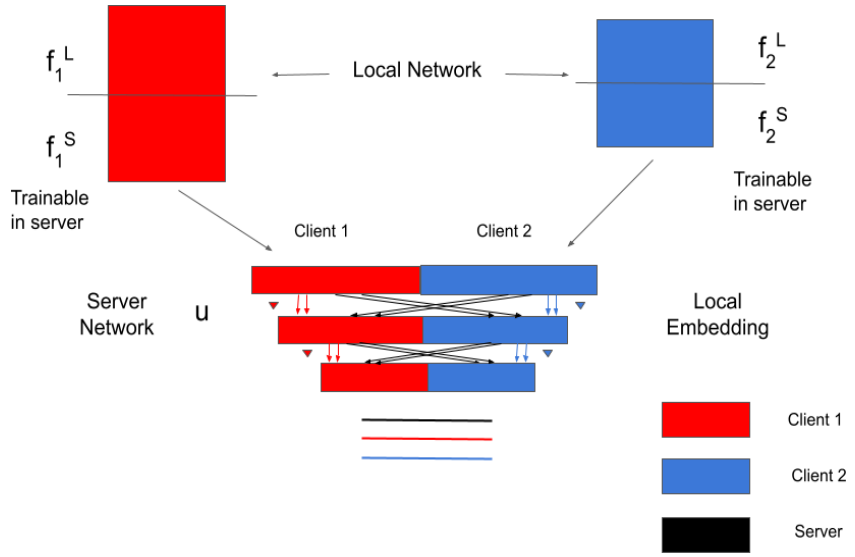

Figure 2. FedEmb learning paradigm (take 2 clients as an example): red and blue parts are heterogenous networks from different clients. We only pass partial networks to the server and embed them to formulate the server network. In server tuning, the network will be comprehensively communicated for learning all dataspaces

Algorithm 1 FebEmd Training

for $j = 1, 2, \ldots, J$ **do**
  for $m = 1, 2, ..., M$ **do local training** in parallel:
    Train local network g_m using local dataset A_m.
    $l_m^L = \{f_m^L(\mathbf{x}_{F_m}; \tau_i^L) | \mathbf{x}_{F_m} \in A_m\}$ and $\theta_m^L = \{g_m(\mathbf{x}_{F_m})) | \mathbf{x}_{F_m} \in A_m\}$ with privacy protections.
    Pass l_m^L ,θ_m^S and f_m^S to the server.
  end for
  for $m = 1, 2, \ldots, M$ **do server tuning**:
    Formulate θ^S, l^L as cohesive set of vectors.
    Train network u using set of vectors θ^S, l^L.
    Let the updated weights be notated by NEW: $\alpha_j^S = \alpha^{S,NEW}$.
    Distribute back $h(\mathbf{x}_{F_m}^S)^{NEW}$ to corresponding client m.
  end for
end for

Algorithm 1. FebEmd Training

3.3 Feature Alignment & Representation

VFL: In the VFL scheme, we assume that the sample IDs are synchronized and aligned in the servers with respect to the lL combination. For current research viewpoint, vertical training and inference is based on collections of all users' data covering all feature spaces without any missing information. Our models combine all underlying features from all local clients by aggregating processed feature embedding vectors and corresponding feature learning network. In this way, server networks are able to fully communicate and learn feature distribution from the other subspaces.

HBFL: For the HBFL scheme, it may not be possible to match the lmL from different users since the sample ID space is not fully disjoint. To address this issue, we randomly match and concatenate lmL in multiple combination formats, and the matching & concatenating order is fixed in every round. As we described before, feature alignment is naturally complex and computationally consuming in vertical-oriented setting. Inspired by transfer learning and presentation learning, our work sheds lights on feature space coupling, which does not impose demanding sample ID cohesion. The stacking structure of the deep neural networks enable feature learning through designing multiple layers of learning nodes. These cascading data down-streaming architectures drives from the assumption of densed representation: observed data is generated by the interactions of many different factors across low-to-high levels. For a deep neural networks, the embedding vectors & activation with regards to intermediate layers are viewed as a representation of the original data space. Each level uses the representation produced by topper layers as input data, then generated new feature embedding for input of next layers as higher-level information. The input at the bottom layer is raw data, and the output of the final layer is the final low-dimensional feature or representation. Following the principle, HBFL scheme is free of ID synchronization issues and the cohesive vector set fully represents the data space for all the data stored in the clients. When our algorithms employ distributed low-dimensional feature embedding vector combination in server, even if the vectors may be misaligned in ID-level, the aggregated ones still represent full data spaces in higher information level. The whole set is effective in being acquired by stacking structure of deep neural networks.

3.4 Privacy Protection

3.4.1 Data Aggregation

In Section 3.2, we demonstrated the learning paradigm for FedEmb, where l^L and θ^S are the aggregation-level output feature embedding vectors of the respective partial layers. Data aggregation is an effective method for protecting the privacy of data owners (Bonawitz et al., 2017; He et al., 2017), and widely applied in deep learning model training (Abadi et al., 2016). In our aggregation implementation, we design two methods for feature embedding vector aggregation: random clustering and K-means clustering. To be specific, in the first global round, we use either of the two methods – random or K-means – to cluster all feature embedding vectors to formulate l^L and θ^S correspondingly. In subsequent global round, s the clustering order is maintained with respect to the sample IDs.

3.4.2 Differential Privacy

As introduced above, differential privacy enables original datasets to be modified with respect to sensitive information without impacting inference accuracy significantly (generally there is a trade-off between the extent of data modification and inference accuracy). For further protecting aggregation-level l^L and θ^S, we deploy either generalization or perturbation on l^L and θ^S.

$$(v_m[d]) = sort(vm[d]) : \text{All } \mathbf{x}[d]_{n_e} = (\{\mathbf{x}[d]_{n_e}\}_{n_e}^{N_e}).$$
$$\text{Recover the original order for } v_m[d].$$

Generalization. Given a specific column of feature dimension in aggregation-level vectors, we sort all observations in increasing order. Based on the generalization hyper-parameter , we generalize all individual feature values of every $N_e = e\% \times N_m$ to the same value, where we use the minimum value of these N_e observations for altering the N_e observations. After the generalization for all N_m observations is completed, we recover the original order of the observations then move forward to the next column of features. With this approach, we add anonymity to every feature value while maintaining the data distribution. The process is summarized in Algorithm 2.

Algorithm 2 Generalization

for $v_m = l_m^L, \theta_m^S$ **do**
  for $d = d_1, d_2, \ldots, d_m$ **do**
    $v_m[d] = \text{sort}(v_m[d])$
    for $\{\mathbf{x}[d]_{n_e}\}_1^{N_e}, \ldots, \{\mathbf{x}[d]_{n_e}\}_{\frac{N_m}{N_e}}^{N_e} \in v_m[d]$ **do**:
      All $\mathbf{x}[d]_{n_e} = \min(\{\mathbf{x}[d]_{n_e}\}_{n_e}^{N_e})$.
    end for
    Recover the original order for $v_m[d]$.
  end for
end for

Algorithm 2. Generalization

Perturbation. Previous studies have shown that Laplace noise has better performance for privacy protection as compared to Gaussian noise used by when layer normalization is applied. So we add Laplace noise into the aggregation-level vectors. The Laplace distribution is formulated as

$$\mathcal{L}(\mathbf{x}|\mu, b) = \frac{1}{2b} exp(\frac{-|\mathbf{x} - \mu|}{b}),$$

where μ and b are local and scale parameters, respectively. Finally,

$$l_m^L = l_m^L + \mathcal{L}(l_m^L|\mu, b), \theta_m^S = \theta_m^S + \mathcal{L}(\theta_m^S|\mu, b)$$

Sensitivity. Sensitivity is to quantify the significance of privacy-preserving. Given a DP query function q that is operating on feature embedding vectors l^L, θ^S and producing the average result difference for $q(l^L)$ and $q(\theta^S)$ will be $\|q(l^L) - l^L\|$ and $\|q(\theta^S) - \theta^S\|$, respectively. We define the minimum sensitivity relative to original data as privacy-preserving metrics (PPM), and select the minimum one out of two kinds of embedding vectors:

$$PPM = min(\frac{\|q(l^L) - l^L\|}{l^L}, \frac{\|q(\theta^S) - \theta^S\|}{\theta^S}),$$

Where $\|.\|$ is the L1-norm distance between data sets differing at most one element. The higher PPM means higher-level privacy-preserving.

4. EXPERIMENT AND DATASETS

4.1. Datasets

We separately test FedEmb for VFL and HBFL schemes using the same datasets, namely:

- **MNIST**: MNIST stands for Mixed National Institute of Standards and Technology, which has produced a handwritten digits dataset. This is one of the most researched datasets in machine learning, and is used to classify handwritten digits. It contains 60,000 training datasets and 10,000 testing datasets with 10 classes.
- **Fashion-MNIST, FASHION**: Fashion MNIST is an alternative to MNIST, and is intended to serve as a direct drop-in replacement for the original MNIST dataset. It contains 60,000 training datasets and 10,000 testing datasets with 10 classes.
- **KDD Network Intrusion, KDD**: The KDD data set is a standard data that has quite a lot of features namely 41 features that are continuous and discrete with normal or anomaly labels (Dos, Probe, R2L, U2R). It contains more than 560k samples with 2 classes.

Each client owns $\frac{1}{M}$ of the feature space and all sample IDs for the VFL scheme, and both $\frac{1}{M}$ feature spaces and $\frac{1}{M}$ sample IDs for the HBFL scheme. In our cases, there is no space overlapping across different decentralized clients.

4.2. Experiments

In this section we present our experimental results, including vertical and hybrid settings. In our case, M for vertical setting is 8 and M for hybrid setting is 4. The less number selection for HBFL is due to the fact that datasets in all users are just the subset of whole space instead of a complete set in relative to VFL, so the large number of clients may cause significant information deficiency. The DNN model is an 8-layer multiple layer perceptron (MLP) with 128 neuron for each hidden layer. The clustering size for data aggregation is set as 15,000, 1,200 and 600, respectively. We perform the experiments using multiple groups of differential privacy parameters: generalization parameter e% is set as 0.21%, 0.83% and 3.3%. The Laplace noise parameters μ and b are set as $(0,1.00), (0,2.50), (0,5.00)$ respectively.

FATE, an industrial-grade project build machine learning models collaboratively at large-scale in a distributed manner. The vertical learning methods in FATE mainly shed light on encrypting gradients and output activation, then send them to servers to conduct aggregated gradient computing. Local clients in FATE are unable to complete the self-training and gradients are exchanged for multiple times for privacy-preserving, which brings severe latency and complexity in system communication.

We compare the performance of FebEmd against two baseline methods: centralized training scheme with all samples and all features; and Heterogeneous Federated Learning using Homomorphic Encryption (FATE) developed by (Zhang et al., 2018; Zhang and Zhu, 2020), whose data distribution across 8 clients aligns with the VFL setting in FebEmd. The training results are shown in Table 2. In terms of HBFL, as we mentioned before, there is no exact investigation on HBFL using DNN modelling, as a consequence, only one centralized baseline setting is applied for reference in Table 3.

4.3. Result Analysis

From Table 2 and Table 3, we observe the following:

1. In terms of the extent of privacy-preserving, higher e in generalization and higher b in perturbation present higher PPM. Dive into different datasets, large absolute value of e and b provide stronger protections on FASHION and KDD than those on MNIST. This trends also align with the results in HBFL setting in Table [tb:

result2], where the size of samples in each client is smaller then VFL setting.

2. Compared with FATE for VFL in Table [tb: result1], FedEmb almostly outperforms FATE for inference accuracy for most cases. The best accuracy in MNIST is 0.890 (1.3 % higher than 0.878 FATE), the best in FASHION is 0.797 (4.2 % higher than 0.765 in FATE), and the best in KDD is 0.894 (0.3 % higher than 0.891 in FATE). For general trends, higher PPM may weaken the performance of FedEmb, but losses are incremental, and maintains the learning effects.
3. With regards to HBFL, the natural insufficient data space decides that the decentralized learning results are much worse than centralized paradigm. In Table [tb: result2], the trend for privacy-preserving VS inference accuracy is similar, where the two factors are negatively affected. 4 clients means all clients only own 25% whole datasets, however, we could still see FedEmb achieves descent learning abilities in global inference. The best HBFL accuracy for MNIST is 0.92, FASHION is 0.757, and KDD is 0.805.
4. For time complexity analysis, the complexity of FATE for single global round is $\mathcal{O}(m) = 2m + 2$, but the complexity of FedEmb is fixed at $\mathcal{O}(m) = 2$. When m = 2,4,8, we can reduce 66.7%, 80.0%, 88.9 %, respectively.

Table 2. VFL Inference Accuracy: Accuracy (Acc.) VS Sensitivity (PPM.) for VFL with 8 clients. Homo. is the homomorphic encryption used by FATE, KM. and Ran. are the clustering methods for data aggregation. G. is the generalization, P. is the perturbation, and e and (μ, b) are the generalization and perturbation parameters if differential privacy is applied to the aggregation-level vectors. No privacy protection measures are applied for the centralized scheme.

Scheme	**Privacy**	**MNIST**			**FASHION**			**KDD**		
		Acc.	**Clu**	**PPM.**	**Acc.**	**Clu**	**PPM.**	**Acc.**	**Clu**	**PPM.**
Centralized	N/A	0.993	N/A	N/A	0.895	N/A	N/A	0.925	N/A	N/A
FATE	Homo.	0.878	N/A	N/A	0.765	N/A	N/A	0.891	N/A	N/A
FedEmb	Ran.	0.881	15k	0.868	0.789	15k	0.662	0.893	15k	1.114
	KM.	0.885	15k	0.994	0.777	15k	0.594	0.892	15k	1.253
	Ran.	0.882	1.2k	1.025	0.771	1.2k	0.781	0.884	1.2k	2.643
	KM.	0.884	1.2k	0.999	0.776	1.2k	0.774	**0.894**	1.2k	2.577
	Ran.	0.871	0.6k	1.055	0.741	0.6k	0.817	0.874	0.6k	3.312
	KM.	0.873	0.6k	1.123	0.753	0.6k	0.963	0.883	0.6k	3.002
FedEmb (G.)	Ran + 0.83%	0.889	15k	0.859	0.779	15k	0.602	0.893	15k	1.006
	Ran + 3.33%	0.878	15k	0.761	**0.797**	15k	0.644	0.892	15k	1.118
	Ran + 10.00%	0.868	15k	0.739	0.792	15k	0.668	0.890	15k	2.553
	Ran + 0.83%	**0.890**	1.2k	1.107	0.774	1.2k	0.795	0.889	1.2k	1.619
	Ran + 3.33%	0.884	1.2k	1.048	0.756	1.2k	0.723	0.884	1.2k	2.434
	Ran + 10.00%	0.875	1.2k	1.009	0.749	1.2k	0.789	0.883	1.2k	2.989
	Ran + 0.83%	0.888	0.6k	1.028	0.742	0.6k	0.786	0.879	0.6k	1.212
	Ran + 3.33%	0.874	0.6k	1.026	0.752	0.6k	0.741	0.885	0.6k	1.856
	Ran +10.00%	0.871	0.6k	1.031	0.757	0.6k	0.789	0.882	0.6k	4.107
FedEmb (P.)	Ran + (0,2.5)	0.886	15k	0.866	0.772	15k	1.282	0.893	15k	1.457
	Ran + (0,5.0)	0.889	15k	0.768	0.795	15k	1.897	0.894	15k	1.112
	Ran + (0,10.0)	0.876	15k	1.047	0.762	15k	3.403	0.892	15k	1.867
	Ran + (0,2.5)	0.893	1.2k	1.006	0.764	1.2k	1.257	0.893	1.2k	1.333
	Ran + (0,5.0)	0.885	1.2k	1.133	0.771	1.2k	2.413	0.888	1.2k	2.390
	Ran + (0,10.0)	0.879	1.2k	1.143	0.745	1.2k	5.338	0.887	1.2k	3.049
	Ran + (0,2.5)	0.894	0.6k	1.107	0.755	0.6k	1.371	0.889	0.6k	1.787
	Ran + (0,5.0)	0.890	0.6k	1.108	0.758	0.6k	2.031	0.881	0.6k	4.919
	Ran + (0,10.0)	0.868	0.6k	1.148	0.742	0.6k	4.218	0.876	0.6k	5.523

Table 3. HBFL Inference Accuracy: Accuracy (Acc.) VS Sensitivity (PPM.) with 4 clients. KM. and Ran. are the clustering methods for data aggregation. G. is the generalization, P. is the perturbation, and e and (μ, b) are the generalization and perturbation parameters if differential privacy is applied to the aggregation-level vectors. No privacy protection measures are applied for the centralized scheme

Scheme	Privacy	MNIST			FASHION			KDD		
		Acc.	Clu	PPM.	Acc.	Clu	PPM.	Acc.	Clu	PPM.
Centralized	N/A	0.993	N/A	N/A	0.895	N/A	N/A	0.925	N/A	N/A
FedEmb	Ran.	**0.892**	15k	0.832	0.754	15k	0.994	0.804	15k	1.222
	KM.	0.888	15k	1.024	**0.757**	15k	0.893	0.801	15k	1.278
	Ran.	0.884	1.2k	1.027	0.749	1.2k	0.881	0.804	1.2k	2.987
	KM.	0.887	1.2k	0.996	0.756	1.2k	0.797	**0.805**	1.2k	2.678
	Ran.	0.892	0.6k	1.055	0.748	0.6k	0.883	0.799	0.6k	3.444
	KM.	0.865	0.6k	1.123	0.752	0.6k	0.997	0.803	0.6k	3.876
FedEmb (G)	Ran + 0.83%	0.889	15k	0.912	0.750	15k	0.602	0.799	15k	3.996
	Ran + 3.33%	0.875	15k	0.901	0.749	15k	0.897	0.792	15k	1.987
	Ran + 10.00%	0.869	15k	0.839	0.746	15k	0.848	0.790	15k	2.346
	Ran + 0.83%	0.861	1.2k	1.403	0.747	1.2k	0.955	0.789	1.2k	2.669
	Ran + 3.33%	0.889	1.2k	1.388	0.755	1.2k	0.933	0.786	1.2k	2.989
	Ran + 10.00%	0.879	1.2k	1.309	0.739	1.2k	0.989	0.793	1.2k	2.989
	Ran + 0.83%	0.881	0.6k	1.426	0.736	0.6k	0.987	0.777	0.6k	3.014
	Ran + 3.33%	0.878	0.6k	1.499	0.746	0.6k	0.941	0.779	0.6k	3.776
	Ran + 10.00%	0.876	0.6k	1.302	0.757	0.6k	0.957	0.782	0.6k	4.552
FedEmb (P.)	Ran + (0, 2.5)	0.889	15k	1.866	0.756	15k	1.282	0.893	15k	1.457
	Ran + (0, 5.0)	0.883	15k	1.768	0.749	15k	2.437	0.794	15k	1.764
	Ran + (0, 10.0)	0.879	15k	1.774	0.736	15k	2.886	0.791	15k	1.887
	Ran + (0, 2.5)	0.878	1.2k	1.879	0.738	1.2k	2.957	0.795	1.2k	1.899
	Ran + (0, 5.0)	0.880	1.2k	1.933	0.731	1.2k	2.813	0.786	1.2k	2.656
	Ran + (0, 10.0)	0.859	1.2k	1.996	0.733	1.2k	3.338	0.787	1.2k	3.942
	Ran + (0, 2.5)	0.873	0.6k	2.107	0.749	0.6k	3.658	0.789	0.6k	3.587
	Ran + (0, 5.0)	0.869	0.6k	2.408	0.748	0.6k	3.954	0.788	0.6k	5.080
	Ran + (0, 10.0)	0.860	0.6k	2.893	0.739	0.6k	4.996	0.774	0.6k	5.882

5. CONCLUSION

FedEmb is a powerful, efficient and economic algorithm for vertical and hybrid federated learning schemes when DNN modelling is applied. In this paper, we start from analyzing the drawback of existing work, then move forward to how FedEmb is able to address the challenges & issues presented by current methods and the detailed learning pipeline in distributed settings. In terms of the characteristics of the algorithms, we totally abandon traditional homophobic encryption on vertical federated learning setting, introduce the approach to deploying partial neural network in local clients to be aggregated in server to learn the split feature space. The paradigm is also successfully extended to hybrid federated learning setting without additional algorithmic designs.

With regards to the series of experiments for both vertical federated learning paradigm and hybrid federated learning paradigm, it demonstrates extraordinary performances for privacy learning. In the first place, FedEmb has tremendous inference accuracy in relative to baseline approaches; Secondly, we extensively discuss the trade-off between accuracy and privacy-preserving, which is extremely enlightening under the constraints of data protections. Last but not least, multiple differential privacy computations are investigated and proposed for enhancing the thoroughness of learning paradigm analysis. For future work, we plan to investigate more scalable FedEmb framework, which aims at enabling more clients network to congregate in server without memory exceeding issues.

References:

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/2976749.2978318

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys, 51*. https://doi.org/10.1145/3214303

Allen-Zhu, Z., Li, Y., & Liang, Y. (2019). Learning and generalization in over-parameterized neural networks, going beyond two layers. *Advances in Neural Information Processing Systems, 32*.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1175–1191). Association for Computing Machinery. https://doi.org/10.1145/3133956.3133982

Chen, R., Fung, B. C., Mohammed, N., Desai, B. C., & Wang, K. (2013). Privacy-preserving trajectory data publishing by local suppression. *Information Sciences, 231*, 83–97.

Du, S. S., Zhai, X., Poczos, B., & Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. [arXiv:1810.02054].

Dwork, C., & Lei, J. (2009). Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing* (pp. 371–380).

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*, 211–407.

Dwork, C., Rothblum, G. N., & Vadhan, S. (2010). Boosting and differential privacy. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science* (pp. 51–60). IEEE.

Errounda, F. Z., & Liu, Y. (2023). Adaptive differential privacy in vertical federated learning for mobility forecasting. *Future Generation Computer Systems*.

Friedman, A., & Schuster, A. (2010). Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 493–502).

Gao, D., Liu, Y., Huang, A., Ju, C., Yu, H., & Yang, Q. (2019). Privacy-preserving heterogeneous federated transfer learning. In *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)* (pp. 2552–2559). https://doi.org/10.1109/BigData47090.2019.9005992

Guo, J., Ho, I. W. H., Hou, Y., & Li, Z. (2023). FedPos: A federated transfer learning framework for CSI-based Wi-Fi indoor positioning. *IEEE Systems Journal*.

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., & Thorne, B. (2017). Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. [arXiv:1711.10677].

He, W., Liu, X., Nguyen, H., Nahrstedt, K., & Abdelzaher, T. (2007). PDA: Privacy-preserving data aggregation in wireless sensor networks. In *Proceedings of the IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications* (pp. 2045–2053). IEEE.

Hu, Y., Liu, P., Kong, L., & Niu, D. (2019b). Learning privately over distributed features: An ADMM sharing approach. [arXiv:1907.07735].

Hu, Y., Niu, D., Yang, J., & Zhou, S. (2019a). FDML: A collaborative machine learning framework for distributed features. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2232–2240). Association for Computing Machinery. https://doi.org/10.1145/3292500.3330765

Kalra, S., Wen, J., Cresswell, J. C., Volkovs, M., & Tizhoosh, H. (2023). Decentralized federated learning through proxy model sharing. *Nature Communications, 14*, 2899.

Kang, Y., Liu, Y., & Chen, T. (2020). FedMVT: Semi-supervised vertical federated learning with multi-view training. [arXiv:2008.10838].

Li, A., Peng, H., Zhang, L., Huang, J., Guo, Q., Yu, H., & Liu, Y. (2023). FedSDG-FS: Efficient and secure feature selection for vertical federated learning. [arXiv:2302.10417].

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. [arXiv:1812.06127].

Li, Y., & Liang, Y. (2018). Learning over-parameterized neural networks via stochastic gradient descent on structured data. *Advances in Neural Information Processing Systems, 31*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. [arXiv:1602.05629].

McSherry, F., & Talwar, K. (2007). Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)* (pp. 94–103). IEEE.

Nissim, K., & Stemmer, U. (2015). On the generalization properties of differential privacy. [arXiv:1504.05800].

Nozari, E., Tallapragada, P., & Cortés, J. (2016). Differentially private distributed convex optimization via functional perturbation. *IEEE Transactions on Control of Network Systems, 5*, 395–408.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2017). Parameter space noise for exploration. [arXiv:1706.01905].

Qi, W., Zhang, R., Zhou, J., Zhang, H., Xie, Y., & Jing, X. (2023). A resource-efficient cross-domain sensing method for device-free gesture recognition with federated transfer learning. *IEEE Transactions on Green Communications and Networking, 7*, 393–400.

Sharma, S., Xing, C., Liu, Y., & Kang, Y. (2019). Secure and efficient federated transfer learning. In *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)* (pp. 2569–2576). https://doi.org/10.1109/BigData47090.2019.9006280

Sidahmed, H., Xu, Z., Garg, A., Cao, Y., & Chen, M. (2021). Efficient and private federated learning with partially trainable networks. [arXiv:2110.03450].

Su, S., Li, B., & Xue, X. (2023). One-shot federated learning without server-side training. *Neural Networks, 164*, 203–215.

Terrovitis, M., Poulis, G., Mamoulis, N., & Skiadopoulos, S. (2017). Local suppression and splitting techniques for privacy-preserving publication of trajectories. *IEEE Transactions on Knowledge and Data Engineering, 29*, 1466–1479.

Wang, C., Liang, J., Huang, M., Bai, B., Bai, K., & Li, H. (2020). Hybrid differentially private federated learning on vertically partitioned data. [arXiv:2009.02763].

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., & Khazaeni, Y. (2020). Federated learning with matched averaging. [arXiv:2002.06440].

Wasserman, L., & Zhou, S. (2010). A statistical framework for differential privacy. *Journal of the American Statistical Association, 105*, 375–389.

Wu, J., Bao, W., Ainsworth, E., & He, J. (2023). Personalized federated learning with parameter propagation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 2594–2605).

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., & Fu, Y. (2019). Large scale incremental learning. [arXiv:1905.13260].

Xiao, Y., & Xiong, L. (2015). Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1298–1309).

Yang, H., He, H., Zhang, W., & Cao, X. (2021). FedSteg: A federated transfer learning framework for secure image steganalysis. *IEEE Transactions on Network Science and Engineering, 8*, 1084–1094. https://doi.org/10.1109/TNSE.2020.2996612

Yang, H., Yao, Q., & Zhang, J. (2020). A distributed federated transfer learning framework for edge optical network. In *Proceedings of the Asia Communications and Photonics Conference/International Conference on Information Photonics and Optical Communications 2020 (ACP/IPOC)*. Optica Publishing Group. https://doi.org/10.1364/ACPC.2020.S4C.4

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. [arXiv:1902.04885].

Yang, Q., Zhang, J., Hao, W., Spell, G. P., & Carin, L. (2021). FLOP: Federated learning on medical datasets using partial networks. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/3447548.3467185

Zhang, Q., Wang, C., Wu, H., Xin, C., & Phuong, T. V. (2018). GELU-Net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (pp. 3933–3939). AAAI Press.

Zhang, W., Wang, Z., & Li, X. (2023). Blockchain-based decentralized federated transfer learning methodology for collaborative machinery fault diagnosis. *Reliability Engineering & System Safety, 229*, 108885.

Zhang, Y., & Zhu, H. (2020). Additively homomorphical encryption based deep neural network for asymmetrically collaborative machine learning. [arXiv:2007.06849].

Zhang, Z., He, N., Li, D., Gao, H., Gao, T., & Zhou, C. (2022). Federated transfer learning for disaster classification in social computing networks. *Journal of Safety Science and Resilience, 3*, 15–23. https://doi.org/10.1016/j.jnlssr.2021.10.007

Zhao, E. M., & Geng, Y. (2019). Homomorphic encryption technology for cloud computing. *Procedia Computer Science, 154*, 73–83. *Proceedings of the 9th International Conference of Information and Communication Technology (ICICT-2019), Nanning, Guangxi, China, January 11-13, 2019.* https://doi.org/10.1016/j.procs.2019.06.012

Zhu, L., Liu, Z., & Han, S. (2019). Deep leakage from gradients. [arXiv:1906.08935].

Fanfei Meng
Department of Electrical and Computer Engineering,
Northwestern University,
Evanston, 60208, IL,
United States
fanfeimeng2023@u.northwestern.edu
ORCID 0009-0009-9272-0665

Lele Zhang
Inistitute of Computing Technology,
Chinese Academy of Science,
Beijing,
100190, China
zhanglele@ict.ac.cn
ORCID 0000-0001-9661-2543

Yu Chen
Inistitute of Computing Technology,
Chinese Academy of Science,
Beijing, 100190,
China
chenyu19s@ict.ac.cn
ORCID 0000-0002-9363-105X

Yuxin Wang
Department of Electrical and Computer Engineering,
Northwestern University,
Evanston, 60208, IL,
United States
yuxinwang2023@u.northwestern.edu