IEEE Open Journal of the Communications Society

Received 11 May 2024; accepted 28 May 2024. Date of publication 3 June 2024; date of current version 25 June 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3409191

Learned Parameter Compression for Efficient and Privacy-Preserving Federated Learning

YIMING CHEN[1,2], LUSINE ABRAHAMYAN[3], HICHEM SAHLI[1,2] (Member, IEEE), AND NIKOS DELIGIANNIS[1,2] (Member, IEEE)

[1]Department of Electronics and Informatics, Vrije Universiteit Brussel, B-1050 Brussels, Belgium

[2]Interuniversitair Micro-Elektronica Centrum, B-3001 Leuven, Belgium

[3]BeVi - Best View, 1190 Vienna, Austria

CORRESPONDING AUTHOR: Y. CHEN (e-mail: cyiming@etrovub.be)

This work was supported in part by the Research Foundation - Flanders (FWO) through the Project under Grant G014718N; in part by the imec.icon Surv-AI-llance Project; and in part by the Flemish Government, under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" Programme.

ABSTRACT Federated learning (FL) performs collaborative training of deep learning models among multiple clients, safeguarding data privacy, security, and legal adherence by preserving training data locally. Despite the benefits of FL, its wider implementation is hindered by communication overheads and potential privacy risks. Transiting locally updated model parameters between edge clients and servers demands high communication bandwidth, leading to high latency and Internet infrastructure constraints. Furthermore, recent works have shown that the malicious server can reconstruct clients' training data from gradients, significantly escalating privacy threats and violating regularizations. Different defense techniques have been proposed to address this information leakage from the gradient or updates, including introducing noise to gradients, performing model compression (such as sparsification), and feature perturbation. However, these methods either impede model convergence or entail substantial communication costs, further exacerbating the communication demands in FL. To develop an efficient and privacy-preserving FL, we introduce an autoencoder-based method for compressing and, thus, perturbing the model parameters. The client utilizes an autoencoder to acquire the representation of the local model parameters and then shares it as the compressed model parameters with the server, rather than the true model parameters. The use of the autoencoder for lossy compression serves as an effective protection against information leakage from the updates. Additionally, the perturbation is intrinsically linked to the autoencoder's input, thereby achieving a perturbation with respect to the parameters of different layers. Moreover, our approach can reduce 4.1 $\times$ the communication rate compared to federated averaging. We empirically validate our method using two widely-used models within the context of federated learning, considering three datasets, and assess its performance against several well-established defense frameworks. The results indicate that our approach attains a model performance nearly identical to that of unmodified local updates, while effectively preventing information leakage and reducing communication costs in comparison to other methods, including noisy gradients, gradient sparsification, and PRECODE.

INDEX TERMS Deep learning, federated learning, data privacy, gradient compression, autoencoder.

I. INTRODUCTION

FEDERATED learning (FL) [1] is a promising framework to address the challenges of privacy and data security when training AI models for applications in, for example, mobile Internet [2], healthcare [3], and Internet of things [4], [5]. In the traditional centralized learning, collecting sensitive data on a central server raises concerns in terms of privacy and data ownership. Instead, FL enables collaborative model training across multiple decentralized clients while ensuring the privacy and security of individual data, as Fig. 1 depicted. For example, Google improved predictive text suggestions

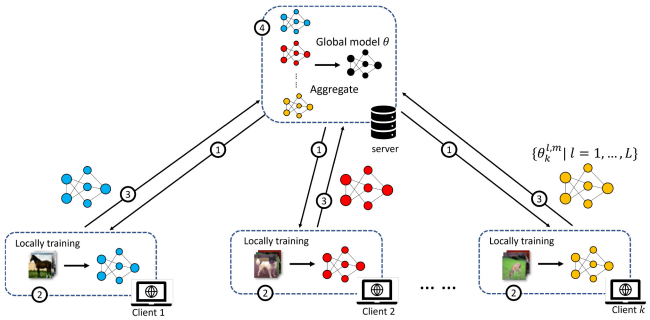


FIGURE 1. Illustration of federated learning, where θ represents the global model maintained by the server, and $\{\theta_k^{l,m} | l = 1, \ldots, L\}$ denotes a set of model parameters indexed by the layer id l, which are locally updated on the client k over m iterations. FL involves the following steps: 1) The server shares the global model with participants. 2) Each client updates the global model using its private data for m iterations, as expressed in (2). 3) The clients transmit the updated model to the server. 4) The server aggregates all received models to form a global model for the subsequent communication round.

in their G-keyboard using FL, without uploading users' information [2].

However, the broad adoption of FL is hampered by significant challenges, including communication overhead and privacy concerns. The exchange of substantial gradient volumes between the central server and edge clients entails high communication rates, which impose bandwidth limitations and latency issues. Moreover, recent studies have shown that the sensitive information of user's training data, such as gender and age, can still be inferred from the exchanged updates of model parameters [6], [7]. An alarming threat is Gradient Inversion Attacks (GIAs) [8], [9], [10], which can directly reconstruct clients' training data from gradients, thereby posing serious privacy risks. GIAs begin by initializing dummy data and calculating corresponding dummy gradients using this data. The attack then proceeds to optimize the dummy data by minimizing the distance between the true and dummy gradients, ultimately enabling the dummy data to closely approximate the original data.

To mitigate information leakage, particularly from GIAs, several defense mechanisms have been explored, including Secure Multi-Party Computation (SMC) methods [11], [12], [13], Differential Privacy (DP) methods [8], [14], [15], [16], [17], [18], and the PRivacy EnhanCing mODulE (PRECODE) strategy [19]. SMC methods enable aggregating sensitive local updates into global updates without disclosing these updates to a potentially malicious server. However, SMC introduces significant computational overhead, necessitating high computational capabilities at the edge clients [20]. DP methods add calibrated noise to updates before sharing, ensuring that these updates do not reveal precise details about the training data. However, this exhibits a balance challenge between model performance and privacy and requires an extensive number of participating clients to achieve model convergence. The PRECODE strategy introduces a Variational Bottleneck (VB) [21] into the model to obscure the data representation, thus hindering the attacks. However, the inserted VB module significantly increases the number of model parameters. Each of these methods further substantially elevates the communication demands in FL.

To establish efficient communication with privacy in FL, some works have explored data compression methods, such as sparsification and quantization [8]. However, these methods confront a trade-off among model performance, communication costs, and privacy preservation. Quantization-based methods reduce data precision to lower bit-widths, which decreases the risk of information leakage and significantly lowers bandwidth needs. Nevertheless, adopting half-precision floating points [22] does not adequately safeguard training data. Conversely, using low-bit representations like INT8 effectively reduces communication costs and prevents data leakage but at the cost of considerable model performance degradation [8]. Sparsification-based methods [23], [24] reduce non-zero elements in local updates by eliminating small data values. Likewise, achieving efficient communication and effective defense requires aggressive sparsification rates, which may also impede training convergence [8].

In this work, we aim to explore an efficient and privacy-preserving FL framework to address the aforementioned gap. This framework should: (1) entail a low communication overhead, (2) maintain the final model's performance despite reduced transmitted information, and (3) protect against information leakage, especially from malicious servers applying GIAs. To fulfill these goals, we employ a lightweight autoencoder to compress (perturb) the local model parameters. By transmitting compressed model parameters through an autoencoder to the server, our approach significantly reduces the communication rate (by 4.1× compared to vanilla FL) and outperforms other defense strategies. In the local training phase, we apply perturbations to the updated model parameters using the autoencoder's lossy compression in each iteration, which obstructs GIAs deployed by malicious servers. Notably, this autoencoder-based perturbation's level is directly related to the autoencoder's input (i.e., model parameters), circumventing the limitations of noisy gradient and sparsification approaches that require a manually predetermined level of perturbation which is hard to choose. We empirically validate our proposed method using two classical models in FL. We train the models using three datasets, both partitioned in independent and identically distributed (IID) and non-independent and non-identically distributed (non-IID) manner. Our method's performance is compared against several established defense mechanisms, including noisy gradient, gradients sparsification, and PRECODE. The experimental results indicate that our method achieves model performance comparable to that of unaltered local updates, while significantly reducing the communication rate and preserving privacy when compared to the aforementioned approaches.

The remainder of the paper reads as follows: Section II discusses background and reviews related work, Section III elaborates on the proposed framework, Section IV

presents our experiments, and Section V draws our conclusion.

II. BACKGROUND AND RELATED WORK

This section discusses background and related work in federated learning (Section II-A), gradient leakage (Section II-B), defence mechanisms (Section II-C) and compression in FL (Section II-D).

A. THE FEDERATED LEARNING APPROACH

Assume a collection of decentralized clients participating in the FL process [1], indexed by $k = 1, 2, \ldots, K$. Each client has a local dataset $\mathcal{D}_k$. The objective is to train a global deep learning model $f(\theta; \cdot)$, which consists of L layers, by optimizing its parameters θ. The layer's parameters are indexed by the corresponding layer's id $l = 1, \ldots, L$. The training process utilizes data from distributed clients and involves message exchanges across T communication rounds. FL [1] reduces the information transferred during training by performing stochastic gradient descent (SGD) update locally at the client before sending the aggregated model update to the server. This optimization problem can be expressed as follows:

$$\underset{\theta}{\operatorname{argmin}} \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \cdot \mathcal{L}(f(\theta; \cdot); \mathcal{D}_k), \tag{1}$$

where $|\cdot|$ denotes the cardinality of a set, and $\mathcal{L}$ is the loss function. To facilitate communication, at each training round, a subset C of $c\%$ of the total number of clients participates in collaborative learning. The server shares the global model with the selected clients in C, which in turn perform local updates of each layer l in each iteration $i = 1, \ldots, m$ by optimizing locally the loss function using their local dataset $\mathcal{D}_k$:

$$\theta_k^{l,i+1} = \theta_k^{l,i} - \eta \nabla_{\theta_k^{l,i}} \mathcal{L}\Big(f(\{\theta_k^{l,i} | l = 1, \ldots, L\}; \cdot), \mathcal{D}_k^i\Big), \tag{2}$$

with η and i denoting the learning rate and the iteration index, respectively, and $\mathcal{D}_k^i$ and $\{\theta_k^{l,i} | l = 1, \ldots, L\}$ being the mini-batch at iteration i and a set of model parameters indexed by layer's id l. Then, the clients send the set of updated parameters $\{\theta_k^{l,m} | i = 1, \ldots, L\}$ back to the server, which then updates the global model parameter of each layer l as follows:

$$\frac{1}{|C|} \sum_{k} \theta_k^{l,m}. \tag{3}$$

Two prominent algorithms are used in FL: in FedSGD a single local update is performed ($m = 1$), whereas in FedAvg multiple local updates are executed ($m > 1$) at the selected clients. Both algorithms allow clients to send either gradients or model parameters to the server, which are functionally equivalent since gradients can be analytically derived from the parameters, and vice versa. Notable, As clients perform a number of local updates, FedAvg reduces the number of global communication rounds, thereby lowering communication costs. Conversely, FedSGD is more vulnerable to gradient inversion attacks (see Section III-B), leading to increased information leakage from gradients.

B. PRIVACY LEAKAGE IN FEDERATED LEARNING

By using the updated model parameters received from clients, a malicious server can analytically calculate gradients and thus reconstruct the clients' local training data through these gradients [8], [9], [10], [25], [26], [27]. *Deep Leakage from Gradient* (DLG) [8] reconstructs the training data by minimizing the distance between the true gradient and the gradient backpropagated using the reconstructed data. This can be formulated as:

$$x', y' = \min_{(x', y')} d\big(\nabla \mathcal{L}(f(\theta; \cdot), x', y'), \nabla_\theta\big), \tag{4}$$

where x' and y' are the dummy input data and the dummy label, $\mathcal{L}$ is the model's loss function, and ∇_θ and $\nabla \mathcal{L}(f(\theta; \cdot), x', y')$ are the received gradient and the gradient obtained with the dummy data and labels, respectively, and d denotes a distance metric (the Euclidean distance in DLG [8]), respectively. As shown in (4), a malicious server could optimize a dummy input x' and a dummy label y' for several iterations, resulting in the reconstruction of the training data without requiring any additional information. *Improved Deep Leakage from Gradients* (iDLG) [25] derives the true label from the sign of gradients in the last layer of the model, when the batch size is one. This approach replaces the dummy label y' with the true label y in (4), further enhancing the reconstruction quality. The study in [26] reported that DLG is highly sensitive to the initialization of dummy data and showed that using data from the same class as the dummy input benefits the reconstruction attack. *Inverting Gradient Attack* (IGA) [9] showed that most gradients from a trained model are very close to zero for different inputs, posing a challenge for DLG to distinguish them when using the Euclidean distance in (4). Therefore, [9] introduced the cosine distance, to measure the direction rather than the magnitude between the true gradient and the dummy gradient, and a Total Variation (TV) regularizer [28] on the dummy data x'. The authors of [9] also proved that the input to any fully connected layer can be recovered analytically, which implies that the Multi-Layered Perceptron (MLP) is more vulnerable to attacks compared with other models, such as Convolutional Neural Networks (CNNs). All aforementioned attack algorithms can target FedAvg by trying to reconstruct a sequence of training data from the summed gradients (final updated parameters). However, these attack algorithms typically reveal more information when targeting FedSGD due to simpler reconstruction optimization process (see Eq. (4)), which makes reconstructing single image batch from its gradients easier. Thus, compared to FedAvg, FedSGD generally results in greater information leakage [8].

C. DEFENSE AGAINST PRIVACY LEAKAGE

Several defense mechanisms against information leakage in FL have been proposed. The efficacy of the privacy-preserving approach is assessed using parameters (gradients) from a single update (FedSGD). This is because the quality of reconstruction in FedSGD sets the minimum baseline for defense efficacy. *Cryptographic protocols*, such as secure multi-party computation (SMC) [11], [12], [13], distribute the computation across multiple parties while preserving the privacy of each party's data. Cryptographic methods ensure that no individual party can access the data of other parties. However, executing complex cryptographic operations across multiple parties requires significant computational resources and results in a considerable computational overhead. *Differential privacy* (DP) methods [14] add random noise into the gradient, offering rigorous privacy protection guarantees, establishing a tradeoff between model performance and information leakage [16]. The study in [17] introduced a new DP perturbation mechanism with a time-varying noise amplitude, aiming to enhance the privacy protection of FL while simultaneously maintaining the ability to adjust the learning performance. However, DP-based methods require a substantial number of participants in the training to achieve model convergence and it faces a trade-off challenge between model performance and information leakage [16]. Soteria [20] introduced perturbations to the data representation obtained from fully connected layers to severely degrade the reconstructed data's quality while maintaining the FL performance. However, a malicious server could bypass the perturbation by simply dropping the perturbed layer, resulting in a perfect reconstruction of the training data [29]. The study in [19] introduced a data representation perturbation method, coined *PRivacy EnhanCing mODulE* (PRECODE), which inserted a Variational Bottleneck (VB) [30] into the model architecture between the feature extractor and the fully-connected layers. The VB projected the data representation into a sampling space and produced a similar but distinct representation. PRECODE aimed to prevent information leakage from the training data by creating a gap between the projected and true representation. However, it modifies the model architecture to accommodate the insertion of a VAE bottleneck, resulting in additional computation and communication overhead. Moreover, [29] showed that the training data can be perfectly reconstructed from MLP models irrespective of PRECODE. These methods notably escalate communication costs in FL, further exacerbating bandwidth and latency issues.

Data compression can also prevent leakage information. Gradient quantization [31] approximates gradients using a lower number of bits than full precision. The reduction in precision can lead to a loss of information, potentially impacting both the convergence of the model and the possibility of information leakage. The study of [8] revealed that using a half-precision float proposed in [22] failed to protect the training data adequately. On the other hand, utilizing a low-bit representation like INT8 successfully prevented data leakage but resulted in a significant drop in the model's performance. Gradient sparsification [23], [24] aims to reduce the number of non-zero elements in gradient updates before transmitting. By identifying and removing these small gradient values, sparsification effectively reduces the overhead of communication. However, the authors of [8] showed that only a high sparsification rate can achieve desirable defensive performance, which in turn can hinder the training convergence. The aforementioned methods depend on manually predetermined hyper-parameters (e.g., sparsification ratio) and thus face an inherent trade-off between model performance, communication costs, and privacy preservation.

D. EFFICIENT COMMUNICATION IN FEDERATED LEARNING

One objective of Federated Learning (FL) is to minimize communication costs during the training of deep learning models; therefore, the communication efficiency and model performance are typically evaluated under the FedAvg algorithm. FL involves a substantial communication cost across numerous clients and servers, resulting in bandwidth requirements and delays. Various compression techniques such as quantization and sparsification are commonly applied on top of FedAvg [1] to reduce the information rate further. The method in [32] proposed quantizing the clients' updates before uploading to the parameter server. Alternatively, the study in [33] proposed moving momentum and error accumulation from clients to the central aggregator using a count sketch randomized data structure. Therefore, the gradients can be randomly projected several times to lower-dimensional spaces, such that high-magnitude elements can later be approximately recovered.

In our earlier research on distributed training, denoted as Learned Gradient Compression (LGC) [34], [35], we introduced a machine-learning method for compressing gradients before transmission. LGC involved training an autoencoder as the compression model using a representative set of gradients, preserving crucial information while reducing update sizes. Our work demonstrated the autoencoder's ability to compress gradients without compromising model performance, even when training deep convolutional neural networks (CNNs) like ResNet-101 [36] on large-scale datasets such as ImageNet [37]. Moreover, our work showed that an autoencoder could achieve rapid convergence with a minimal number of training iterations (around 200) in the distributed training approach. LGC focuses on enhancing communication efficiency through gradient compression in distributed training scenarios. In contrast, our emphasis in this work lies in privacy preservation achieved through lossy compression. This involves perturbing local model parameters within an autoencoder framework explicitly designed for FL.

III. PROPOSED METHOD

This section elaborates on the proposed privacy-preserving FL framework, describing the learned parameter compression

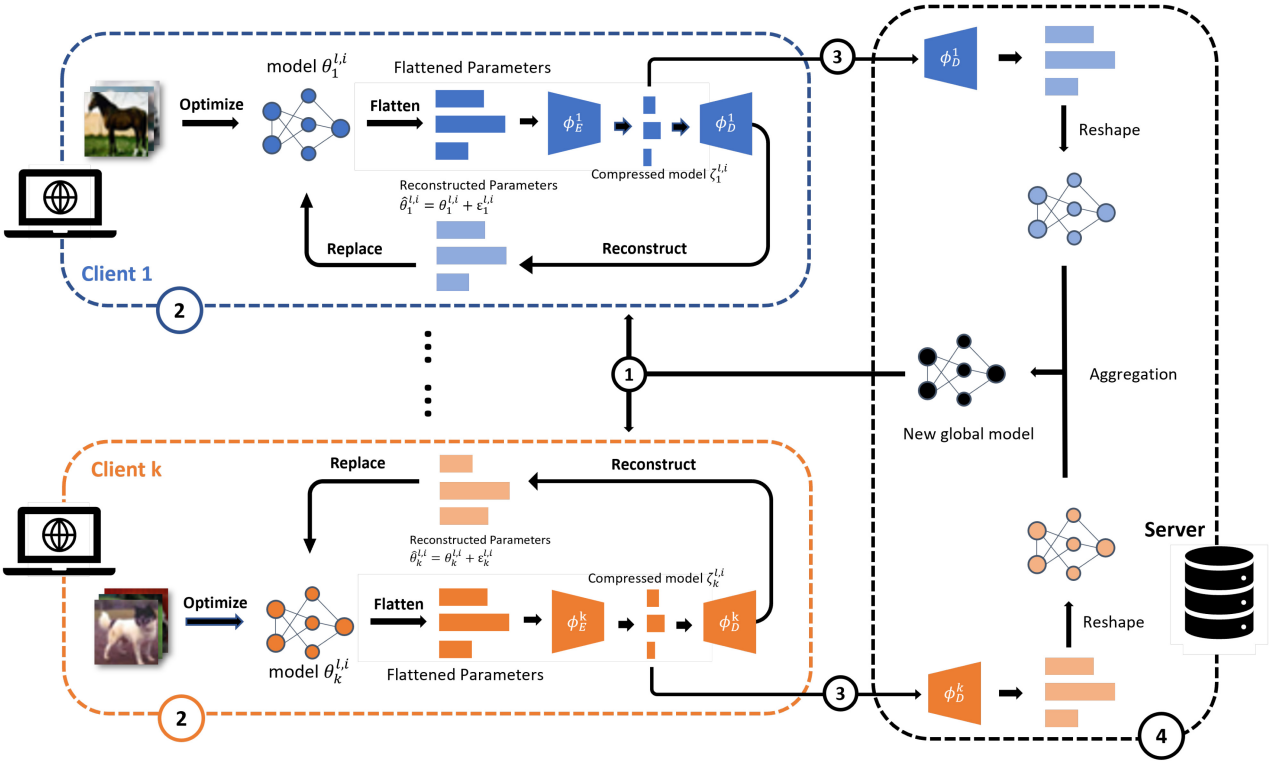


FIGURE 2. Illustration of the proposed federated learning approach. It involves the following steps: 1) The server shares the global model with participants. 2) Each client updates the global model using its private data for *m* iterations and uses the pre-trained autoencoder to perturb (compress) the model parameters in every iteration, as expressed in eq. (11). 3) The clients share the compressed model with the server per eqs. (5) and (6). 4) The server reconstructs the model and aggregates all received models to form a global model for the subsequent communication round.

method (Section III-A), the architecture of the autoencoder we consider (Section III-B), and the way we apply learned parameter compression into FL (Section III-C). Figure 2 depicts the overview of our method.

A. LEARNED PARAMETER COMPRESSION

In the classical FL system, discussed in Section II-A, consider a set of model parameters, denoted as $\{\theta_k^{l,i} | l = 1, \ldots, L\}$, each representing the parameters of the l^{th} layer of the model in the i^{th} iteration at client k. In our approach, these parameters are initially transformed into 1D vectors, that is:

$$\Theta_k^{l,i} = \texttt{flatten}\left(\theta_k^{l,i}\right), \tag{5}$$

where $\texttt{flatten}(\cdot)$ denotes the flattening function and $\Theta_k^{l,i}$ is the resulting parameter vector. At client k an autoencoder, comprising an encoder ϕ_E^k and a decoder ϕ_D^k, is used to compress the model parameters per local iteration. Specifically, the encoder ϕ_E^k compresses the flattened model parameter $\Theta_k^{l,i}$ into a lower-dimensional representation,

$$\zeta_k^{l,i} = \phi_E^k\left(\Theta_k^{l,i}\right) = \phi_E^k\left(\texttt{flatten}\left(\theta_k^{l,i}\right)\right). \tag{6}$$

Depending on the training stage (see Section III-C), the compressed representation is either sent to the server, where it is decoded and aggregated with the decoded parameters from the other clients, or decoded locally at client k, leading to a perturbed version that is used in the subsequent local iteration. Concretely, in the latter case, the decoder at client k performs lossy reconstruction of the flattened model parameters:

$$\hat{\Theta}_k^{l,i} = \phi_D^k\left(\zeta_k^{l,i}\right) = \Theta_k^{l,i} + \epsilon_k^{l,i}, \tag{7}$$

where $\epsilon_k^{l,i}$ denotes the reconstruction error with respect to the input $\Theta_k^{l,i}$. The reconstructed parameters are then reshaped back and replace the original, uncompressed, parameters:

$$\hat{\theta}_k^{l,i} = \texttt{reshape}\left(\hat{\Theta}_k^{l,i}\right), \tag{8a}$$

$$\theta_k^{l,i} \leftarrow \hat{\theta}_k^{l,i}. \tag{8b}$$

The compression error $\epsilon_k^{l,i}$ by the autoencoder acts as a perturbation on the model parameters applied per local iteration, serving as a method to obscure the real update information and hinder an attacker's ability to reconstruct the training data. As we will show in Section IV, this compression error does not hinder training convergence and model performance. Moreover, as the clients exchange low-dimensional (compressed) representation of the model parameters the communication cost across the clients and the server is reduced.

B. AUTOENCODER ARCHITECTURE

We propose using an one-dimensional (1D) convolutional kernel for the construction of the autoencoder model,

TABLE 1. Architecture of the Convolutional Layers of the Encoder.

Layer	Filters	Kernel Size	Stride
conv1	64	3	2
conv2	128	3	2
conv3	256	3	2
conv4	64	3	2
conv5	4	1	1

TABLE 2. Architecture of the Convolutional Layers of the Decoder.

Layer	Filters	Kernel Size	Stride
deconv1	4	3	2
deconv2	32	3	2
deconv3	64	3	2
deconv4	128	3	2
deconv5	32	3	1
conv	1	1	1

operating on flattened parameter tensors. Following this approach, enables compressing the parameters of various deep learning models irrespective of the size of the model. Furthermore, using an 1D convolutional kernel offers a reduction in the autoencoder parameters by approximately 60% compared to a 2D convolutional kernel [34], thereby mitigating the risk of overfitting and reducing encoding and decoding latency, complexity and memory footprint. In this work, the encoder of the autoencoder compresses the parameter vectors through a series of five 1D convolutional kernels, each followed by a leaky-ReLU [38] activation function (see Table 1). The decoder (whose architecture is given in Table 2) consists of five 1D deconvolutional kernels, each also followed by a leaky-ReLU activation function, and concludes with a 1D convolutional kernel of size one.

C. PROPOSED FEDERATED LEARNING PROCESS

Consider a decentralized environment comprising K clients, indexed by k, the objective is to train a deep learning model (with layers $l = 1, \ldots, L$) using FL, specifically FedAvg as detailed in Section II-A. In each communication round $t = 1, \ldots, T$, the server selects a subset C clients to participate the training. The client $k \in C$ trains the global model disseminated by the server across m local iterations.

In the first communication rounds, each client experiences rapid changes in the model parameters during its initial local iterations [39]. We use the terms *warming-up rounds* and *warming-up iterations* to refer to the initial communication rounds and the local iterations therein, respectively. Applying compression to the parameters updated during the warming-up iterations may detrimentally affect model performance. Therefore, in line with alternative decentralized training methods [34], [39], [40], we do not apply compression during the warming up iterations. Instead, during the warming-up iterations each client trains (in the first communication round that the client participates) or fine-tunes (in subsequent communication rounds that the client participates) its client-specific autoencoder to be used for parameter compression in the subsequent iterations. In what follows, we describe how we train and use the autoencoder in the different stages, namely, initialization, warming-up communication rounds, and regular communication rounds, of the FedAvg process.

1) INITIALIZATION

Before initiating the FL process, the server initializes a global model, and each client k initializes an autoencoder per the architecture outlined in Section III-B. This initial version of the autoencoder is stored locally at the client.

2) WARMING-UP COMMUNICATION ROUNDS

During the warming-up communication rounds, $t = 1, \ldots, T_w - 1$, and for a number of warming-up iterations, $i = 1, \ldots, m_w - 1$, each participating client loads and fine-tunes the local autoencoder.[1] This fine-tuning process utilizes each layer of model parameters $\theta_k^{l,i}$ at every iteration and is described as follows:

$$\phi_E^k = \phi_E^k - \lambda \nabla_{\phi_E^k} L_{\text{rec}}\left(\phi_E^k, \phi_D^k, \theta_k^{l,i}\right), \tag{9a}$$

$$\phi_D^k = \phi_D^k - \lambda \nabla_{\phi_D^k} L_{\text{rec}}\left(\phi_E^k, \phi_D^k, \theta_k^{l,i}\right), \tag{9b}$$

where λ represents the learning step of the autoencoder training. The loss function for the autoencoder training is the mean squared error (MSE) between the actual and predicted parameters:

$$L_{rec}\left(\phi_E, \phi_D, \theta_k^{l,i}\right) = ||\hat{\Theta}_k^{l,i} - \Theta_k^{l,i}||_2^2. \tag{10}$$

As will be shown in Section IV, $m_w = 30$ warming-up iterations per client can lead to a well-trained autoencoder. After these iteration, the autoencoder is stored locally for usage in future rounds. Algorithm 1 depicts the proposed FL process during the warming-up iterations.

In the rest of the local iterations during the warming-up communication rounds, $i = m_w, \ldots, m$, the local model updates become stable and are effectively learned by the autoencoder. Thus, the autoencoder is capable of perturbing the model parameters without compromising model performance. After each training iteration, the original model parameter $\theta_k^{l,i}$ is replaced with the reconstructed parameter $\hat{\theta}_k^{l,i}$, obtained from the autoencoder as detailed in (5), (6), (7), and (8). Concretely, the client estimates the next iteration's model parameters using the reconstructed model parameters from the previous iteration $\hat{\theta}_k^{l,i}$, rather than the original parameters $\theta_k^{l,i}$, that is:

$$\hat{\theta}_k^{l,i} = \texttt{reshape}\left(\phi_D^k(\phi_E^k(\texttt{flatten}(\theta_k^{l,i})))\right); \tag{11a}$$

$$\theta_k^{l,i+1} = \hat{\theta}_k^{l,i} - \eta \nabla_{\hat{\theta}_k^{l,i}} \mathcal{L}\left(f(\{\hat{\theta}_k^{l,i} | l = 1, \ldots, L\}; \cdot), \mathcal{D}_k^i\right), \tag{11b}$$

where η and $\mathcal{D}_k^i$ denote the learning rate of the model and the mini-batch at iteration i, respectively. Algorithm 2

[1] We use T_w and m_w to denote the number of warming-up communication rounds and iterations, respectively.

Algorithm 1 Warming-up Iteration Training by the Client

Require: Pre-saved autoencoders ϕ_E^k, ϕ_D^k for each client k; received model parameters $\theta_k^{l,0}$ indexed by layer l from the server; learning step of the model η; model's loss function $\mathcal{L}(\cdot)$; learning step of autoencoder λ; loss function of autoencoder L_{rec}; the mini-batch $\mathcal{D}_k^i$ at iteration i, respectively.

for each client k **do**
  Load pre-saved autoencoders ϕ_E^k, ϕ_D^k
  for i in warming-up iterations **do**
    $\texttt{loss} = \mathcal{L}(f(\{\theta_k^{l,i-1} | l = 1, \ldots, L\}; \cdot); \mathcal{D}_k^{i-1})$
    for $l = 1$ to L **do**
      $\theta_k^{l,i} \leftarrow \theta_k^{l,i-1} - \eta \nabla_{\theta_k^{l,i-1}} \texttt{loss}$
      $\phi_E^k \leftarrow \phi_E^k - \lambda \nabla_{\phi_E^k} L_{\text{rec}}(\phi_E^k, \phi_D^k, \theta_k^{l,i})$
      $\phi_D^k \leftarrow \phi_D^k - \lambda \nabla_{\phi_D^k} L_{\text{rec}}(\phi_E^k, \phi_D^k, \theta_k^{l,i})$
    end for
  end for
  Store the fine-tuned autoencoder ϕ_E^k, ϕ_D^k locally
end for

Ensure: Fine-tuned autoencoders ϕ_E^k, ϕ_D^k

Algorithm 2 Regular Iteration Training by the Client

Require: Pre-saved autoencoders ϕ_E^k, ϕ_D^k for each client k; model parameters $\theta_k^{l,0}$ indexed by layer l; learning step of the model η; learning step of autoencoder λ; loss function of the model $\mathcal{L}(\cdot)$; the mini-batch $\mathcal{D}_k^i$ at iteration i, respectively.

for each client k **do**
  Load pre-saved autoencoders ϕ_E^k, ϕ_D^k
  for i in regular iterations **do**
    for $l = 1$ to L **do**
      $\theta_k^{l,i} \leftarrow \texttt{reshape}(\phi_D^k(\phi_E^k(\texttt{flatten}(\theta_k^{l,i}))))$
    end for
    $\texttt{loss} = \mathcal{L}(f(\{\theta_k^{l,i} | l = 1, \ldots, L\}; \cdot), \mathcal{D}_k^{i-1})$
    for $l = 1$ to L **do**
      $\theta_k^{l,i} \leftarrow \theta_k^{l,i-1} - \eta \nabla_{\theta_k^{l,i-1}} \texttt{loss}$
    end for
  end for
end for

Ensure: Locally Trained model

illustrates the training process during the regular iterations in the warming-up communication rounds.

At the end of each warming-up communication round, that is, after completing m local training iterations, the clients share the perturbed parameters $\hat{\theta}_k^{l,m}$ with the server. The server then updates the parameters per layer per (3) and shares the updated model with the clients. Note that during the warming-up communication rounds the model parameters change rapidly thereby sending the compressed representations (instead of the perturbed version) does not lead to good model performance.

Algorithm 3 Global Model Aggregation by the Server

Require: Pre-received decoders ϕ_D^k and received compressed model parameters $\zeta_k^{l,i}$ indexed by layer l from each client k; C is the total set of participants, where $k \in C$, the set of new model parameters θ_{new}, respectively.

for each layer l **do**
  $\theta^l = 0$
  for each client k **do**
    $\theta^l = \theta^l + \texttt{reshape}(\phi_D^k(\zeta_k^{l,i}))$
  end for
  $\theta^l = \theta^l / |C|$
  $\theta_{\texttt{new}}.\text{append}(\theta^l)$
end for

Ensure: a set of new global model parameters $\theta_{\texttt{new}}$ indexed by layer's id

3) REGULAR COMMUNICATION ROUNDS

At the conclusion of the warming-up rounds, specifically at $t = T_w - 1$, the clients share the trained decoders ϕ_D^k of the local autoencoder with the server. After participating in training for T_w rounds, the change in the model parameters at the clients becomes stable, and the autoencoders have been effectively trained. Therefore, in the remaining *regular communication rounds,* $t = T_w, \ldots, T$, the clients only need to transmit the compressed model parameters, allowing the server to reconstruct the model on the server side meanwhile reducing the communication costs.

Specifically, during the local iterations $i = 1, \ldots, m$ of a regular communication round, the original model parameters $\theta_k^{l,i}$ are replaced and by the locally reconstructed (perturbed) parameters $\hat{\theta}_k^{l,i}$, aligning with the process described in Section III-C2 (see Algorithm 2). After the completion of m local training iterations, the clients share their compressed model parameters $\zeta_k^{l,m}$ with the server. Once the server receives the compressed model parameters $\zeta_k^{l,m}$, for the participating clients $k = 1, \ldots, K$, it reconstructs them utilizing the corresponding decoders ϕ_D^k [see (10) and (8)]. The server then updates the global model parameters of layer l to:

$$\frac{1}{|C|} \sum_{k \in C} \texttt{reshape}\left(\phi_D^k(\zeta_k^{l,i})\right) \tag{12}$$

Algorithm 3 elaborates on the process of reconstructing and aggregating the model parameters by the server.

IV. EXPERIMENTS

In this section, we evaluate our approach in terms of model performance, information leakage, and communication rate.

A. EXPERIMENTAL SETUP

The experimental settings, which follow prior studies [8], [19], [20], [25], [41], are detailed below.

Models: We evaluate our approach regarding the model performance and privacy by using two models commonly employed in FL, namely (1) LeNet [8], [9], [25]: comprises a sequence of three convolutional layers with Sigmoid activations. The first layer takes input and applies 12 convolutional filters with a kernel size of 5, using a stride of 2. Subsequent convolutional layers also apply 12 filters under similar parameters, with the final layer utilizing a stride of 1. Following the convolutional components, the network flattens the output and passes it through a fully connected layer, mapping to the desired number of classes, and (2) Multilayer Perceptron (MLP) [19], [42]: consists of three layers, with a hidden dimension of 256. Since the input to any fully connected layer can be recovered analytically [9], such an MLP model is widely adopted to show the upper bound of privacy leakage.

FL Tasks and Datasets: We consider three widely-discussed FL tasks, including (1) Zalando's article images classification task on the Fashion-MNIST dataset [43]: which contains a collection of 70,000 grayscale clothing images, divided into a training set of 60,000 samples and a test set of 10,000 samples. Each image has a resolution of 28×28 pixels and belongs to one of 10 classes, (2) 10-class images classification task on the CIFAR-10 dataset [44]: which consists of a collection of 60,000 color images, divided into a training set with 50,000 images and a test set with 10,000 images. Each image has a resolution of 32×32 pixels and is labeled with one out of ten different classes, and (3) 100-class images classification task on the CIFAR-100 dataset [44]: which has 100 classes with 20 superclasses. We evaluate the model performance on both IID and non-IID data partition settings, where the partitions align with the setting in [1].

FL Implementation: The experiments are conducted on a single machine equipped with two GeForce RTX 3090 Ti GPUs and 128 GB of RAM. Random seeds are set to 1234 for reproducibility. For all FL tasks, we conduct 500 communication rounds, involving synchronization between the server and 100 clients. The local training configurations vary, with models undergoing either 1 or 10 epochs of local training at each client before aggregation. In terms of our approach, as detailed in Section III, we set the warming-up rounds T_w to 35 for each client, and the fine-tuning process for the client's autoencoder entails using updates twice over a span of 30 iterations ($m_w = 30$).

Attacks: To evaluate privacy leakage, we attempt reconstructing 128 randomly sampled training data by using two well-established gradient inversion attacks, namely (1) *Deep Leakage from Gradients (DLG) [8]:* employs MSE as the gradient matching loss, and (2) *Inverting Gradients Attack* (IGA) [9] utilizes negative cosine similarity loss as the gradient matching loss with TV as the image prior regularization. For both attack algorithms, the Adam optimizer is leveraged to optimize the dummy training data over 7,000 iterations with an initial learning rate of 0.1, reduced by 10% at milestones of $\frac{1}{8}$, $\frac{3}{8}$, and $\frac{7}{8}$ of the total iterations. We have not considered attack algorithms, such as GradInversion [10], which requires the additional statistics of data as this contradicts the basic concept of FL [45].

Baselines: We compare our method with three established baseline approaches, including (1) *Noisy Gradient* (NG) [14], incorporating two levels of Gaussian noise into the gradients, generated by considering zero means and standard deviations of 10^{-1} and 10^{-2}. These levels are respectively denoted as NG-10^{-1} and NG-10^{-2}, (2) *Gradient Sparsification* (GS) [8], retaining the top 10% and 30% of significant gradients, while assigning zero to the remaining gradients; we refer to these configurations as GS-90% and GS-70%, respectively and (3) *PRivacy EnhanCing mODulE* (PRECODE) [19], appending a variational bottleneck [21] after the feature extractor in the LeNet model and the first two layers in the MLP model.

Evaluation Metrics: To assess our proposed FL framework's effectiveness, we measure the *top-1 accuracy* across various FL tasks to determine model performance. For communication efficiency, we report the *Compression Ratio* (CR), which expresses the ratio between the original and compressed data sizes. To evaluate privacy, we quantify information leakage using four metrics: (1) *Mean Squared Error* (MSE↑), indicating the average squared error between pixels of original and recovered images; (2) *Peak Signal-to-Noise Ratio* (PSNR↓), measuring the maximum potential power of an image against the power of corrupting noise in decibels (dB); (3) *Structural Similarity* (SSIM↓) [46], assessing similarity based on structural, luminance, and contrast differences; and (4) *Attack Success Rate* (ASR↓), defined as the rate of successful image reconstructions with an SSIM value of at least 0.6 [19], [26]. Lower values of PSNR, SSIM, and ASR alongside a higher MSE suggest a better privacy protection.

B. MODEL PERFORMANCE EVALUATION

The performance of the model is evaluated using the FedAvg algorithm. Tables 2, 4, and 5 report the top-1 accuracy achieved by the MLP and LeNet with different defense methods on the Fashion-MNIST, CIFAR-10, and CIFAR-100 dataset, respectively. Bold numbers denote the best performance and underlined numbers indicate the second-best performance. B and E refer to the batch size and local number of epochs. To demonstrate the impact of the different defenses on model performance, we utilize two different batch sizes: a small batch size of 8 and a large batch size of 64. Our framework shows competitive performance across various models and partitions. The accuracy achieved by our approach is nearly close to the baseline (FedAvg). Consistent with the observations reported in [14], noisy gradient with a high noise level ($\sigma = 10^{-1}$) leads to a considerable decrease in accuracy; for example, in the case of the MLP model trained on the non-IID partition of Fashion-MNIST with a batch size of 64 and 8, the accuracy drops by 18.73% and 15.39%, respectively. In contrast, our proposed method exhibits only a marginal decrease of 1.14% and 0.51% in accuracy. When applying high

TABLE 3. Top-1 Accuracy in Federated Learning of the MLP and LeNet on Both the IID and Non-IID Fashion-MNIST Dataset.

	$B = 64, E = 10$				$B = 8, E = 1$			
	LeNet		MLP		LeNet		MLP	
	IID	non-IID	IID	non-IID	IID	non-IID	IID	non-IID
FedAvg	89.02	79.78	89.89	85.06	87.85	80.53	89.70	85.13
DP (10^{-1})	86.68 (-2.34)	76.20 (-3.58)	84.79 (-5.10)	66.33 (-18.73)	86.97 (-0.88)	79.18 (-1.35)	84.70 (-5.0)	69.74 (-15.39)
NG (10^{-2})	89.00 (-0.02)	**79.67 (-0.11)**	89.53 (-0.36)	85.12 (+0.06)	**88.14 (+0.29)**	**80.62 (+0.09)**	89.43 (-0.27)	**85.29 (+0.16)**
GS (90%)	87.28 (-1.74)	76.85 (-2.93)	89.29 (-0.60)	84.43 (-0.63)	86.28 (-1.57)	77.45 (-3.08)	88.93 (-0.77)	84.33 (-0.80)
GS (70%)	88.41 (-0.61)	79.44 (-0.34)	**89.85 (-0.04)**	**85.15 (+0.09)**	87.65 (-0.20)	80.02 (-0.51)	**89.44 (-0.26)**	84.89 (-0.24)
PRECODE	**89.17 (+0.15)**	73.24 (-6.54)	89.33 (-0.56)	82.25 (-2.81)	85.17 (-2.68)	10.91 (-69.62)	88.84 (-0.86)	80.27 (-4.86)
Our work	88.24 (-0.78)	77.64 (-2.14)	89.76 (-0.13)	83.92 (-1.14)	87.05 (-0.80)	77.83 (-2.70)	89.22 (-0.48)	84.62 (-0.51)

TABLE 4. Top-1 Accuracy in Federated Learning of the MLP and LeNet on Both the IID and Non-IID CIFAR-10 Dataset.

	$B = 64, E = 10$			$B = 8, E = 1$		
	LeNet		MLP	LeNet		MLP
	IID	non-IID	IID	IID	non-IID	IID
FedAvg	60.75	46.73	52.88	56.99	46.18	50.59
NG (10^{-1})	54.48 (-6.27)	37.27 (-9.46)	38.73 (-14.15)	51.09 (-5.90)	41.22 (-4.96)	24.62 (-25.59)
NG (10^{-2})	**60.66 (-0.09)**	**48.20 (+1.47)**	52.06 (-0.82)	**56.21 (-0.78)**	**47.16 (+0.98)**	49.87 (-0.72)
GS (90%)	54.48 (-6.27)	40.39 (-6.34)	52.41 (-0.47)	50.14 (-6.85)	40.85 (-5.33)	47.81 (-2.78)
GS (70%)	59.94 (-0.84)	47.62 (+0.89)	**53.07 (+0.19)**	54.00 (-2.99)	45.97 (-0.21)	**49.96 (-0.63)**
PRECODE	56.07 (-4.68)	39.26 (-7.47)	50.22 (-2.66)	52.16 (-4.83)	10.87 (-35.31)	48.52 (-2.07)
Our work	60.43 (-0.32)	45.23 (-1.50)	52.87 (-0.01)	54.82 (-2.17)	41.65 (-4.53)	46.77 (-3.82)

sparsification (90%), we observe a significant degradation in the performance of LeNet on CIFAR-10 and CIFAR-100. Specifically, when using a batch size of 64, the accuracy of LeNet drops by 6.27% and 6.34% on CIFAR-10 for the IID and non-IID partitions, respectively. In contrast, our framework experiences only a minimal loss of 0.32% and 1.50% in accuracy for the same setting. The results reveal that applying gradient sparsification of (70%) and noisy gradient with a low noise level ($\sigma = 10^{-2}$) leads to the best model training performance most of the times. However, as we will see in Section IV, these configurations do not effectively prevent the reconstruction of training data from local updates. Furthermore, PRECODE exhibits limited accuracy robustness, particularly when dealing with non-IID data and smaller batch sizes. When training LeNet on the Fashion-MNIST and CIFAR-10 datasets with a batch size of 8, PRECODE achieves accuracy levels of only 10.91% and 10.87%, respectively, underlying its inability to maintain satisfactory performance under these conditions.

TABLE 5. Top-1 accuracy in federated learning of LeNet on IID CIFAR-100.

	LeNet	
	$B = 64, E = 10$	$B = 8, E = 1$
FedAvg	27.63	24.71
NG (10^{-1})	22.57 (-0.51)	17.41 (-7.30)
NG (10^{-2})	**27.87 (+0.24)**	25.27 (+0.56)
GS (90%)	22.44 (-5.19)	20.36 (-4.35)
GS (70%)	23.58 (-4.05)	24.97 (+0.26)
PRECODE	25.25 (-2.38)	**25.28 (+0.57)**
Our work	27.02 (-0.61)	23.69 (-1.02)

TABLE 6. Compression Ratio (CR) of Various Defenses.

	FedAvg	NG (10^{-1})	NG (10^{-2})	GS (90%)	GS (70%)	Our work
CR	1.00×	1.00×	1.00×	3.33×	1.11×	**4.10** ×

TABLE 7. Compression Ratio (CR) of Various Models on Different Datasets Using PRECODE.

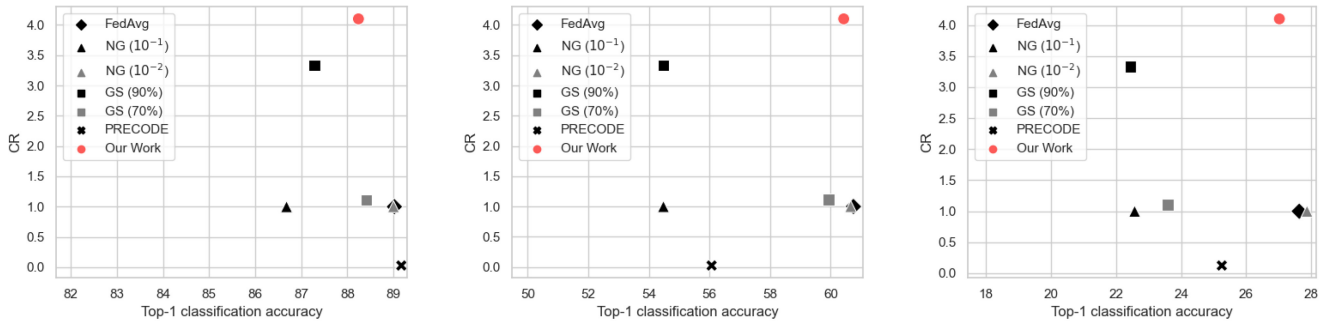
	LeNet-FashionMNIST	LeNet-CIFAR10	LeNet-CIFAR100	MLP-FashionMNIST	MLP-CIFAR10
CR	0.0288×	0.0260×	0.1258×	0.5771×	0.8125×

C. COMMUNICATION COSTS

The communication cost is computed when utilizing the FedAvg algorithm. Tables 6 and 7 report the compression ratio (CR) of all defenses as an ancillary evaluation. CR is defined as

$$\text{CR} = \texttt{size}\left(\theta^{\text{original}}\right)/\texttt{size}\left(\theta^{\text{compressed}}\right), \tag{13}$$

where θ^{original} and $\theta^{\text{compressed}}$ are the uncompressed model and the compressed model by each defense approaches, respectively; the $\texttt{size}(\cdot)$ function computes the size of the parameters. The number of parameters of the decoder ϕ_D of our autoencoder is 72,352, resulting to a size of 289, 408 bytes, which is only transmitted once. The compression ratio results in Table 6 and 7 indicate that our framework can achieve a 4.1× compression ratio compared to the FedAvg method while at the same time maintaining privacy (as we will discuss in Section IV On the contrary, PRECODE incurs a communication rate that is 34.8 times greater than FedAvg when employed to LeNet on the Fashion-MNIST dataset.



(a) Fashion-MNIST (b) CIFAR-10 (c) CIFAR-100

FIGURE 3. The top-1 accuracy of LeNet, trained with a batch size of 64, is plotted against the compression ratio (CR). The CR is calculated as per (13). The results are presented for (a) Fashion-MNIST, (b) CIFAR-10, and (c) CIFAR-100, where the data is split IID in the participating clients.

TABLE 8. Evaluation of the Defense Effectiveness of Untrained Models on the Fashion-MNIST Dataset.

Defense	Untrained model on the Fashion-MNIST dataset							
	LeNet (DLG/IGA)				MLP (DLG/IGA)			
	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓
FedAvg	0.00/0.00	60.46/36.08	0.99/0.95	100/100	0.00/0.00	102.65/83.82	1.00/1.00	100/100
NG (10^{-1})	18.14/0.46	-6.11/9.67	0.05/0.27	0/1	0.34/0.29	10.79/11.38	0.32/0.29	0/0
NG (10^{-2})	0.03/0.02	22.51/24.66	0.81/0.84	98/100	0.00/0.01	30.55/28.76	0.92/0.92	100/100
GS (90%)	13.12/0.68	-4.76/7.68	0.03/0.12	0/0	0.08/0.05	17.23/19.42	0.60/0.64	49/63
GS (70%)	1.74/0.23	4.29/12.53	0.26/0.43	0/1	0.00/0.00	33.55/35.58	0.97/0.97	100/100
PRECODE	2.18/**1.01**	2.66/**6.00**	**0.01/0.04**	0/0	**0.91/1.47**	**6.53/4.36**	**0.06/0.00**	0/0
Our work	**32.67**/0.74	**-7.91**/7.34	0.027/0.072	0/0	0.48/0.40	9.45/9.97	**0.06**/0.03	0/0

TABLE 9. Evaluation of the Defense Effectiveness of Untrained Models on the CIFAR-10 Dataset.

Defense	Untrained model on the CIFAR-10 dataset							
	LeNet (DLG/IGA)				MLP (DLG/IGA)			
	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓
FedAvg	0.38/0.29	18.95/19.01	0.62/0.60	58/56	0.00/0.01	52.89/49.34	1.00/0.99	100/100
NG (10^{-1})	4.98/1.47	5.31/10.63	0.10/0.18	0/0	0.48/0.48	15.30/15.32	0.34/0.33	0.02/0
NG (10^{-2})	0.73/0.50	17.13/17.35	0.56/0.53	54/48	0.01/0.01	33.86/33.67	0.96/0.95	100/100
GS (90%)	5.27/**1.93**	5.03/**9.22**	0.04/0.08	0/0	0.12/0.07	21.97/24.22	0.72/0.76	91/93
GS (70%)	2.90/1.05	7.63/11.92	0.13/0.21	0/0	0.00/0.00	37.07/37.43	0.98/0.98	100/100
PRECODE	2.34/1.90	8.43/9.31	**0.01**/0.04	0/0	**0.66/3.10**	**14.22/7.14**	0.19/**0.05**	0/0
Our work	**6.50**/1.71	**4.05**/9.86	0.02/**0.03**	0/0	0.60/0.66	14.89/14.34	**0.14**/0.10	0/0

Fig. 3 depicts the top-1 accuracy of LeNet trained with a batch size of 64 plotted against the CR for various datasets (IID splits). Our work achieves the best trade-off between communication rate reduction and model accuracy compared to all other methods.

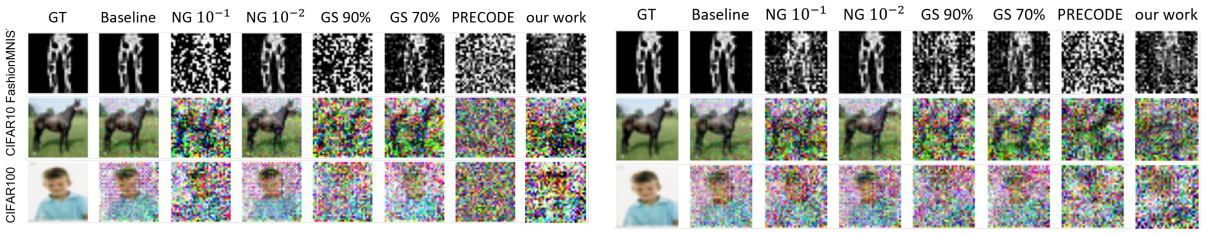
D. DEFENSE RESULTS

Consistent with the established practices in this research field [8], the effectiveness of the defense is assessed using FedSGD [1], attributed to the simpler task of data reconstruction from a single update [8]. Consequently, the quality of reconstruction in FedSGD sets the minimum baseline for defense efficacy.

1) INFORMATION LEAKAGE FROM UNTRAINED MODELS

When machine learning models are trained from scratch in FL, the magnitude of the gradient is generally large, making it easier to infer the training data [9]. Therefore, following the settings in [8], [9], [19], [20], we first evaluate the privacy-preserving capability of our approach for untrained, randomly initialized models. For a given test data sample, we perform a single training iteration of a randomly initialized model (LeNet or MLP) using this data sample and obtain the updated model parameters. Then, we use the updated model parameters to train the autoencoder using 100 SGD iterations. Subsequently, we employ the trained autoencoder to compress the model parameters and then attempt to reconstruct the training data using the decompressed model parameters. It is clear that the autoencoder is overfitting in this case, resulting in a lower bound for privacy preservation.

Tables 8, 9, and 10 report information leakage results when attacking LeNet and MLP using DLG and IGA on the Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, respectively. The first number represents the metric achieved



(a) Attack on untrained LeNet by DLG

(b) Attack on untrained LeNet by IGA

FIGURE 4. Reconstruction results on the Fashion MNIST, CIFAR10, and CIFAR100 datasets for the initialized LeNet model and different defense mechanisms.

TABLE 10. Evaluation of the Defense Effectiveness of Untrained Models on the CIFAR-100 Dataset.

Untrained model	Untrained LeNet on CIFAR-100							
	DLG				IGA			
	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓
FedAvg	0.4684	15.156	0.399	3	0.5639	14.2303	0.3044	1
NG (10^{-1})	3.7914	5.7627	0.0702	0	1.271	10.5457	0.1178	0
NG (10^{-2})	0.5094	14.7163	0.381	0	0.6138	13.8747	0.2783	0
GS (90%)	3.0733	6.7929	0.0432	0	1.3862	10.2143	0.0624	0
GS (70%)	1.739	9.2713	0.1294	0	0.9316	11.9544	0.154	0
PRECODE	2.1969	8.203	0.0066	0	**1.6899**	**9.3411**	**0.0331**	0
Our work	**5.2635**	**4.3801**	0.0153	0	1.3550	10.3967	0.0442	0

TABLE 11. Evaluation of the Information Leakage on MLP Models for the Fashion-Mnist and CIFAR Datasets by using IGA in the warming-up round.

Defense	Fashion-MNIST				CIFAR			
	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓
FedAvg	0.00	35.97	0.97	100	1.97	14.58	0.37	28.75
NG (10^{-1})	1.69	3.94	0.03	**0**	**4.10**	**6.28**	0.03	**0**
NG (10^{-2})	0.33	10.90	0.29	0	3.39	10.18	0.22	6.35
GS (90%)	0.66	14.36	0.45	42.5	2.05	11.28	0.23	5
GS (70%)	0.91	9.28	0.29	18.75	2.16	13.70	0.37	27.5
PRECODE	1.58	4.52	0.04	1.25	1.84	10.40	0.05	2.5
Our work	**1.69**	**3.77**	**0.00**	**0**	1.52	10.82	**0.00**	**0**

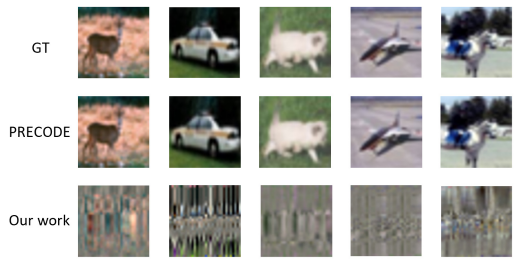


FIGURE 5. Reconstruction results on the CIFAR-10 by using Bayesian Framework for Gradient Leakage [29].

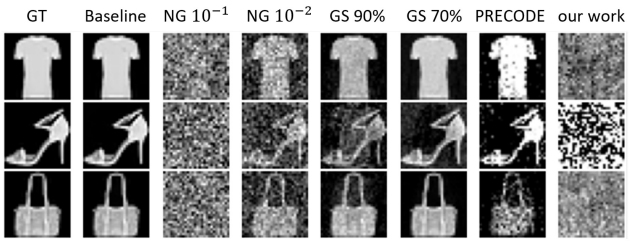
using DLG, and the second number corresponds to the metric obtained using IGA. The bold numbers denote the best performance and the underlined numbers indicate the second-best performance. The results show that without privacy protection (FedAvg), both attacks can effectively reconstruct the data. When using high sparsification (90%) with gradient sparsification or high noise level within noisy gradient (NG-10^{-1}), the data can be effectively protected. However, as discussed in Section IV-B, these methods significantly impact the model's performance. Conversely, the data can be attacked when using low noise level (NG-10^{-2}) with noisy gradient. In the case of MLP, neither high sparsification (90%) in gradient sparsification nor noisy gradient with a significant noise level (NG-10^{-1}) effectively prevent data leakage. PRECODE leads to non-convergence for non-IID data when employed with small batch sizes, particularly when training CNN models. Moreover, a malicious server could circumvent the defense of PRECODE by using the attack proposed in [29] if the model is fully-connected. The training data can be analytically reversed by multiplying the gradient of weights and biases in the first layer.

Fig. 4 depicts visual examples of reconstructed data samples. noisy gradient and gradient sparsification techniques are ineffective in concealing information from the gradients. However, our framework successfully removes nearly all useful information, resulting in the attacker being able to reconstruct only a highly blurred dummy image. Fig. 5 depicts the reconstruction results using the attack in [29], which successfully circumvents the PRECODE defense mechanism in the MLP model. Contrarily, our work demonstrates effective countermeasures against such forms of attacks.

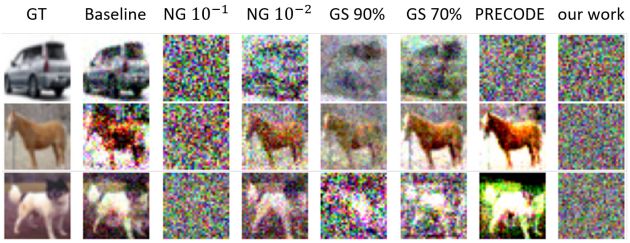
2) INFORMATION LEAKAGE FROM TRAINED MODELS

We also examine the effectiveness of various defenses in preventing privacy leakage during both the warming-up and regular rounds. Specifically, training data are inverted after 5 (warming-up) and 40 (regular) training communication rounds. We select a client randomly and acquire 128 local training data samples, which are then organized into batches of 8 for updating model parameters. The model parameters are compressed and decompressed by the autoencoder and used to reconstruct the data batch using the corresponding attack.

Warming-Up Round: Table 11 reports the results of information leakage when attacking the locally trained MLP model using IGA on the Fashion-MNIST and CIFAR-10 in the 5^{th} round, respectively. The results obtained with noisy gradient and gradient sparsification are consistent with those reported in Section IV-D2, while our approach is the most effective in protecting information. Fig. 6 depicts visual examples of data reconstructed from attacking the MLP model. It is clear that intentionally perturbing model parameters with an autoencoder during local training

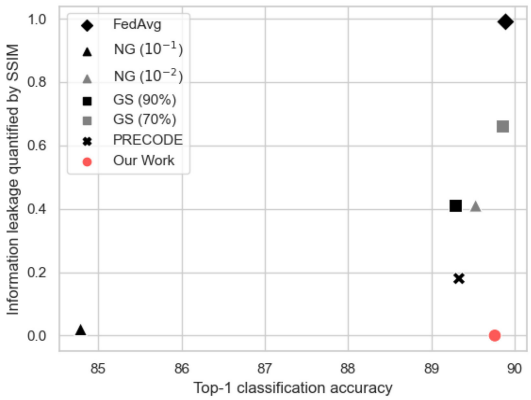


(a) Fashion-MNIST data reconstructed by IGA

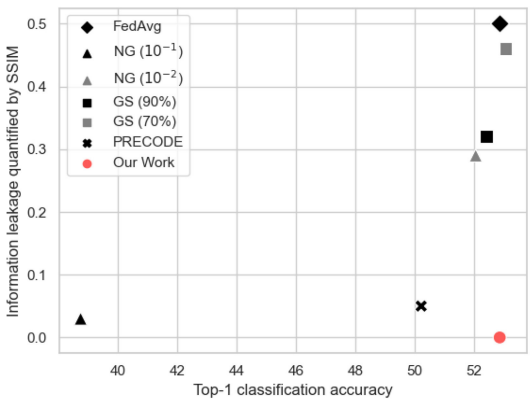


(b) CIFAR-10 data reconstructed by IGA

FIGURE 6. Reconstructed training data including, (a) Fashion-MNIST and (b) CIFAR-10, from the gradient of trained MLP, in the warming-up round.



(a) Fashion-MNIST



(b) CIFAR-10

FIGURE 7. The top-1 accuracy of MLP, trained with a batch size of 64, is plotted versus the Structural Similarity Index Measure (SSIM) of the reconstructed training data across various datasets (IID splits), which includes (a) Fashion-MNIST and (b) CIFAR-10 datasets.

iterations can act as a defense mechanism against gradient inversion attacks, aligning with our motivation.

Regular Round: Table 12 reports a quantitative evaluation of information leakage for the locally trained MLP model on the Fashion-MNIST dataset in the 40^{th} round. By checking the SSIM values, it's evident that our approach still effectively safeguards information and does not exhibit information leakage in rounds of the regular phase.

Balancing Privacy and Model Performance: Fig. 7 illustrates the top-1 accuracy of MLP, which is trained using a batch size of 64, plotted against the SSIM of the reconstructed training data by using IGA for various IID datasets in the 5^{th} round. Notably, our work secures the highest level of privacy protection while simultaneously preserving the model's performance.

TABLE 12. Evaluation of the Information Leakage on MLP Models for the Fashion-Mnist and CIFAR Datasets by using IGA in the regular round.

Defense	Fashion-MNIST				CIFAR			
	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓	MSE ↑	PSNR ↓	SSIM ↓	ASR ↓
FedAvg	0.00	32.68	0.95	100	2.01	13.53	0.34	28.75
NG (10^{-1})	1.78	3.56	0.03	**0**	**4.25**	**6**	0.02	**0**
NG (10^{-2})	0.45	10.18	0.23	0	3.59	10.02	0.19	6.01
GS (90%)	0.73	13.72	0.41	42	2.18	11.01	0.19	4.5
GS (70%)	0.98	9.20	0.24	18.75	2.22	13.61	0.35	27.5
PRECODE	1.52	4.64	0.05	1.5	1.87	10.51	0.05	2.5
Our work	**1.81**	**3.55**	**0.00**	**0**	1.68	8.29	**0.00**	**0**

V. CONCLUSION

We proposed an autoencoder-based method to reduce communication costs and safeguard neural network models against the risk of information leakage during the exchange of model updates in federated learning. We assessed our method through experiments on two classical neural network architectures and three datasets, considering scenarios with both iid and non-iid data partitions. The evaluation encompassed two well-known gradient inversion attacks. When contrasted with other defense strategies, including differential privacy, data compression, and PRECODE, our approach stands out due to its ability to balance the communication overhead, privacy, and model performance effectively. Our method achieves accuracy levels comparable to those obtained during unprotected training and less communication cost with a compression ratio of at least four times less rate compared to federated averaging, while simultaneously preventing information leakage.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from Decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2016, pp. 1–10.

[2] A. AbhishekV, S. Binny, R. JohanT, N. Raj, and V. Thomas, "Federated learning: Collaborative machine learning without centralized training data," *Int. J. Eng. Technol. Manage. Sci.*, vol. 6, no. 5, pp. 355–359, 2022.

[3] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Inform. Res.*, vol. 5, pp. 1–19, Nov. 2021.

[4] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1622–1658, 3rd Quart., 2021.

[5] X. Liu et al., "Distributed intelligence in wireless networks," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 1001–1039, 2023.

[6] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning: Revisited and enhanced," in *Proc. Int. Conf. Appl. Techn. Inf. Secur.*, 2017, pp. 100–110.

[7] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 619–633.

[8] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients–how easy is it to break privacy in federated learning?" 2020, *arXiv:2003.14053*.

[10] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16337–16346.

[11] R. Canetti, "Security and composition of multiparty cryptographic protocols," *J. Cryptol.*, vol. 13, pp. 143–202, Apr. 2000.

[12] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 639–648.

[13] J. Zhao, H. Zhu, F. Wang, R. Lu, Z. Liu, and H. Li, "PVD-FL: A privacy-preserving and verifiable Decentralized federated learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2059–2073, 2022.

[14] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.

[15] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[16] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:207847853

[17] X. Yuan, W. Ni, M. Ding, K. Wei, J. Li, and H. V. Poor, "Amplitude-varying perturbation for balancing privacy and utility in federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1884–1897, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257405067

[18] Z. Liang, P. Yang, C. Zhang, and X. Lyu, "Secure and efficient hierarchical decentralized learning for Internet of Vehicles," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 1417–1429, 2023.

[19] D. Scheliga, P. Mäder, and M. Seeland, "PRECODE-a generic model extension to prevent deep gradient leakage," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, 2021, pp. 3605–3614.

[20] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 9307–9315.

[21] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: https://api.semanticscholar.org/CorpusID:216078090

[22] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2017, pp. 1051–1056.

[23] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2017, pp. 1–6.

[24] Y. Tsuzuku, H. Imachi, and T. Akiba, "Variance-based gradient compression for efficient distributed deep learning," 2018, *arXiv:1802.06058*.

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

[26] W. Wei et al., "A framework for evaluating client privacy leakages in federated learning," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2020, pp. 545–566.

[27] A. Hatamizadeh et al., "Gradvit: Gradient inversion of vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10021–10030.

[28] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D, Nonlin. Phenomena*, vol. 60, pp. 259–268, Nov. 1992. [Online]. Available: https://api.semanticscholar.org/CorpusID:13133466

[29] M. Balunovi'c, D. I. Dimitrov, R. Staab, and M. T. Vechev, "Bayesian framework for gradient leakage," 2021, *arXiv:2111.04706*.

[30] A. A. Alemi, I. S. Fischer, J. V. Dillon, and K. P. Murphy, "Deep variational information bottleneck," 2016, *arXiv:1612.00410*.

[31] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[32] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2021–2031.

[33] D. Rothchild et al., "FetchSGD: Communication-efficient federated learning with sketching," 2020, *arXiv:2007.07682*.

[34] L. Abrahamyan, Y. Chen, G. Bekoulis, and N. Deligiannis, "Learned gradient compression for distributed deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7330–7344, Dec. 2022.

[35] L. Abrahamyan, Y. Chen, G. Bekoulis, and N. Deligiannis, "Autoencoder-based gradient compression for distributed training," in *Proc. 29th Eur. Signal Process. Conf. (EUSIPCO)*, 2021, pp. 2179–2183.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[37] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, Dec. 2015.

[38] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*. [Online]. Available: https://api.semanticscholar.org/CorpusID:14083350

[39] Y. Lin, S. Han, H. Mao, Y. Wang, and W. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: https://openreview.net/pdf?id=SkhQHMW0W

[40] J. Ma, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Structured federated learning through clustered additive modeling," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023, pp. 1–11. [Online]. Available: https://openreview.net/forum?id=2XT3UpOv48

[41] W. Wei, L. Liu, Y. Wu, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2021, pp. 797–807.

[42] Q. Li, B. He, and D. X. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 10708–10717.

[43] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[44] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1–60.

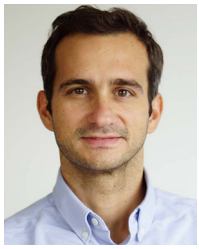
[45] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 7232–7241.

[46] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, pp. 600–612, 2004.

YIMING CHEN received the B.S. degree in automatics from Northwestern Polytechnical University, Xi'an, China, in 2019, and the M.S. degree in applied computer science from Vrije Universiteit Brussel, Brussels, Belgium, in 2020, where he is currently pursuing the Ph.D. degree with the Electronics and Informatics Department (ETRO) and Interuniversitair Micro-Elektronica Centrum (IMEC), Belgium. His research focuses on privacy-preserving and distributed deep learning.

LUSINE ABRAHAMYAN received the B.S. and M.S. degrees in physics from Yerevan State University, Armenia, in 2012 and 2014, respectively, and the Doctoral degree in engineering from the Department of Electronics and Informatics, Vrije Universiteit Brussel, Belgium, in 2022. She was a Researcher with PicsArt Inc. from 2014 to 2018. She is currently a Founder and the CTO of BeVi AI FlexCo. Her research interests include optimization of deep learning models, distributed training, and applications of information theory for machine learning.

HICHEM SAHLI (Member, IEEE) is currently a Professor in computer vision and machine learning with the Department of Electronics and Informatics, Vrije Universiteit Brussel, Brussels, Belgium, and the Principal Scientist with Interuniversitair Micro-Elektronica Centrum vzw, Leuven, Belgium. He has authored over 300 journals, conference publications, and book chapters. His research focuses on signal processing and machine learning theory and algorithms in computer vision (image, radar, and video processing), health informatics (disease progression prediction, diagnosis, and treatment outcome prediction), and natural language processing (electronic health record coding).



NIKOS DELIGIANNIS (Member, IEEE) received the Diploma degree in electrical and computer engineering from the University of Patras, Patras, Greece, in 2006, and the Ph.D. degree (Hons.) in engineering sciences from Vrije Universiteit Brussel, Brussels, Belgium, in 2012, where he is currently an Associate Professor with the Department of Electronics and Informatics, and a Principal Investigator with imec, Leuven, Belgium. From 2013 to 2015, he was a Senior Researcher with the Department of Electronic and Electrical Engineering, University College London, London, U.K. His current research interests focus on interpretable and explainable machine learning, image and video processing, computer vision, and distributed deep learning. He served as the Chair for the EURASIP Technical Area Committee on Signal and Data Analytics for Machine Learning from 2021 to 2023. He serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and he was the Lead Guest Editor of the special issue on "Understanding and Designing Deep Neural Networks" at the *EURASIP Journal on Advances in Signal Processing*. He is a member of the EURASIP.