A New Federated Learning Framework Against Gradient Inversion Attacks

Pengxin Guo[1*], Shuang Zeng[2*], Wenhao Chen[1], Xiaodan Zhang[3], Weihong Ren[4], Yuyin Zhou[5], Liangqiong Qu[1†]

[1]School of Computing and Data Science, The University of Hong Kong
[2] Department of Mathematics, The University of Hong Kong
[3] College of Computer Science, Beijing University of Technology
[4] School of Mechanical Engineering and Automation, Harbin Institute of Technology, Shenzhen
[5] Department of Computer Science and Engineering, UC Santa Cruz
{guopx, zengsh9}@connect.hku.hk, wc365@cam.ac.uk, zhangxiaodan@bjut.edu.cn, renweihong@hit.edu.cn, zhouyuyiner@gmail.com, liangqqu@hku.hk

Abstract

Federated Learning (FL) aims to protect data privacy by enabling clients to collectively train machine learning models without sharing their raw data. However, recent studies demonstrate that information exchanged during FL is subject to Gradient Inversion Attacks (GIA) and, consequently, a variety of privacy-preserving methods have been integrated into FL to thwart such attacks, such as Secure Multi-party Computing (SMC), Homomorphic Encryption (HE), and Differential Privacy (DP). Despite their ability to protect data privacy, these approaches inherently involve substantial privacy-utility trade-offs. By revisiting the key to privacy exposure in FL under GIA, which lies in the frequent sharing of model gradients that contain private data, we take a new perspective by designing a novel privacy preserve FL framework that effectively "breaks the direct connection" between the shared parameters and the local private data to defend against GIA. Specifically, we propose a Hypernetwork Federated Learning (HyperFL) framework that utilizes hypernetworks to generate the parameters of the local model and only the hypernetwork parameters are uploaded to the server for aggregation. Theoretical analyses demonstrate the convergence rate of the proposed HyperFL, while extensive experimental results show the privacy-preserving capability and comparable performance of HyperFL.

Code — https://github.com/Pengxin-Guo/HyperFL
Extended version — https://arxiv.org/abs/2412.07187

1 Introduction

Deep Neural Networks (DNN) have achieved remarkable success on a variety of computer vision tasks, relying on the availability of a large amount of training data (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016; Dosovitskiy et al. 2021; Liu et al. 2021; Wang et al. 2023). However, in many real-world applications, training data is distributed across different institutions, and data sharing between these entities is often restricted due to privacy and regulatory concerns. To alleviate these concerns, Federated Learning (FL) (McMahan et al. 2017; Li et al. 2020a; Qu et al. 2022; Zeng et al. 2024; Guo et al. 2024; Zhang et al. 2024) has emerged as a promising approach that enables collaborative and decentralized training of AI models across multiple institutions without sharing of personal data externally.

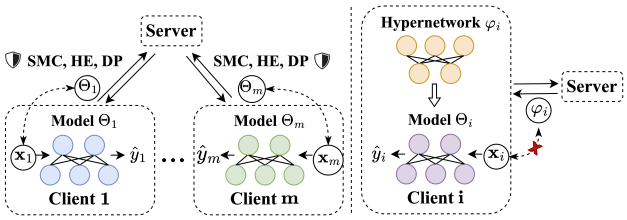


Figure 1: *Left*. Existing methods mainly explore defenses mechanisms on the shared gradients. Such mechanisms, including SMC, HE, and DP, inherently involve substantial privacy-utility trade-offs. *Right*. A novel FL framework that "breaks the direct connection" between the shared parameters and the local private data is proposed to achieve a favorable privacy-utility trade-off.

Despite the privacy-preserving capability introduced by FL, recent works (Geiping et al. 2020; Huang et al. 2021; Hatamizadeh et al. 2023) have revealed that FL models are vulnerable to Gradient Inversion Attacks (GIA) (Fredrikson, Jha, and Ristenpart 2015; Zhu, Liu, and Han 2019). GIA can reconstruct clients' private data from the shared gradients, undermining FL's privacy guarantees. To remedy this issue, various defense mechanisms have been integrated into FL, including Secure Multi-party Computing (SMC) (Yao 1982; Bonawitz et al. 2017; Mugunthan et al. 2019; Mou et al. 2021; Xu et al. 2022), Homomorphic Encryption (HE) (Gentry 2009; Zhang et al. 2020a,b; Ma et al. 2022; Park and Lim 2022) and Differential Privacy (DP) (Dwork 2006; Geyer, Klein, and Nabi 2017; McMahan et al. 2018; Yu, Bagdasaryan, and Shmatikov 2020; Bietti et al. 2022; Shen et al. 2023). These approaches primarily rely on existing defense mechanisms to enhance privacy protection against

*These authors contributed equally.
†Corresponding author.

GIA without altering the FL framework, as illustrated in the left part of Figure 1. However, such mechanisms, including SMC, HE, and DP, inherently involve substantial privacy-utility trade-offs. For example, SMC and HE, while providing strong security guarantees through encrypted information exchange, entail high computation and communication costs, making them unsuitable for DNN models with numerous parameters (Bonawitz et al. 2017; Zhang et al. 2020b). Although DP approaches are easier to implement, they often fall short in providing sufficient model protection while preserving accuracy (Geyer, Klein, and Nabi 2017; McMahan et al. 2018). These trade-offs within current defense algorithms have inspired us to explore alternative privacy-preserving methods that strike a better balance between privacy and utility, leading to the central question of this paper:

Can we design a novel FL framework that offers a favorable privacy-utility trade-off against GIA without relying on existing defense mechanisms?

To this end, we revisit the key to privacy exposure in FL under GIA, which lies in the frequent sharing of model gradients that contain private data. Most efforts aim at exploring various advanced defenses mechanisms on the shared gradients to enhance privacy preservation in FL, as shown in the left part of Figure 1. In contrast, we take a new perspective striving for designing a novel privacy preserve FL framework that "breaks the direct connection" between the shared parameters and the local private data to defend against GIA. In order to achieve this, we explore the potential of hypernetworks, a class of deep neural networks that generate the weights for another network (Ha, Dai, and Le 2017), as a promising solution, as illustrated in the right part of Figure 1 and Figure 2.

Specifically, we introduce a novel Hypernetwork Federated Learning (HyperFL) framework that adopts a dual-pronged approach—network decomposition and hypernetwork sharing— to "break the direct connection" between the shared parameters and the local private data, as shown in Figure 2. In light of recent findings regarding minimal discrepancies in feature representations and considerable diversity in classifier heads among FL clients (Collins et al. 2021; Xu, Tong, and Huang 2023; Shen et al. 2023), we decompose each local model into a shared feature extractor and a private classifier, enhancing performance in heterogeneous settings and mitigating privacy leakage risks. To further strengthen privacy preservation, we employ an auxiliary hypernetwork that generates feature extractor parameters based on private client embeddings. Instead of directly sharing the feature extractor, only the hypernetwork parameters are uploaded to the server for aggregation, while classifiers and embeddings are trained locally. This auxiliary hypernetwork sharing strategy "breaks the direct connection" between shared parameters and local private data, maintaining privacy while enabling inter-client interaction and information exchange.

Remarkably, the design of HyperFL is flexible and scalable, catering to a diverse range of FL demands through its various configurations. We present two major configurations of HyperFL: (1) Main Configuration HyperFL, suitable for simple tasks with small networks, learns the entire feature extractor parameters directly (see Figure 2); (2) HyperFL for Large Pre-trained Models (denoted as HyperFL-LPM) targets for complex tasks by using pre-trained models as fixed feature extractors and generating trainable adapter parameters via a hypernetwork (Houlsby et al. 2019) (see Figure 3). Both theoretical analysis and extensive experimental results demonstrate that HyperFL effectively preserves privacy under GIA while achieving comparable results and maintaining a similar convergence rate to FedAvg (McMahan et al. 2017). We hope that the proposed HyperFL framework can encourage the research community to consider the importance of developing new enhanced privacy preservation FL frameworks, as an alternative to current research efforts on defense mechanisms front.

We summarize our contributions as follows:

- To defend against GIA, we take a new perspective by designing a novel privacy preserve FL framework that effectively "breaks the direct connection" between the shared parameters and the local private data and propose the HyperFL framework.
- We present two major configurations of HyperFL: (1) Main Configuration HyperFL, suitable for simple tasks with small networks, learns the entire feature extractor parameters directly; (2) HyperFL-LPM targets for complex tasks by using pre-trained models as fixed feature extractors and generating trainable adapter parameters via a hypernetwork.
- Both theoretical analysis and extensive experimental results demonstrate that HyperFL effectively preserves privacy under GIA while achieving comparable results and maintaining a similar convergence rate to FedAvg.

2 Related Work

Gradient Inversion Attacks. Gradient Inversion Attacks (GIA) (Fredrikson, Jha, and Ristenpart 2015; Zhu, Liu, and Han 2019) are adversarial attacks that exploit a machine learning model's gradients to infer sensitive information about the training data. It iteratively adjusts the input data based on gradients to approximate private attributes or training samples (Phong et al. 2017; Zhu, Liu, and Han 2019; Geiping et al. 2020; Yin et al. 2021; Luo et al. 2022; Geng et al. 2023; Kariyappa et al. 2023).

Privacy Protection in Federated Learning. To protect the data privacy in FL, additional defense methods have been integrated into FL, such as Secure Multi-party Computing (SMC) (Yao 1982) based methods (Bonawitz et al. 2017; Mugunthan et al. 2019; Mou et al. 2021; Xu et al. 2022), Homomorphic Encryption (HE) (Gentry 2009) based methods (Zhang et al. 2020a,b; Ma et al. 2022; Park and Lim 2022) and Differential Privacy (DP) (Dwork 2006) based methods (Geyer, Klein, and Nabi 2017; McMahan et al. 2018; Yu, Bagdasaryan, and Shmatikov 2020; Bietti et al. 2022; Shen et al. 2023). Apart from these defense methods with theoretical guarantees, there are other empirical yet effective defense strategies, such as gradient pruning/masking (Zhu, Liu, and Han 2019; Huang et al. 2021; Li et al. 2022), noise

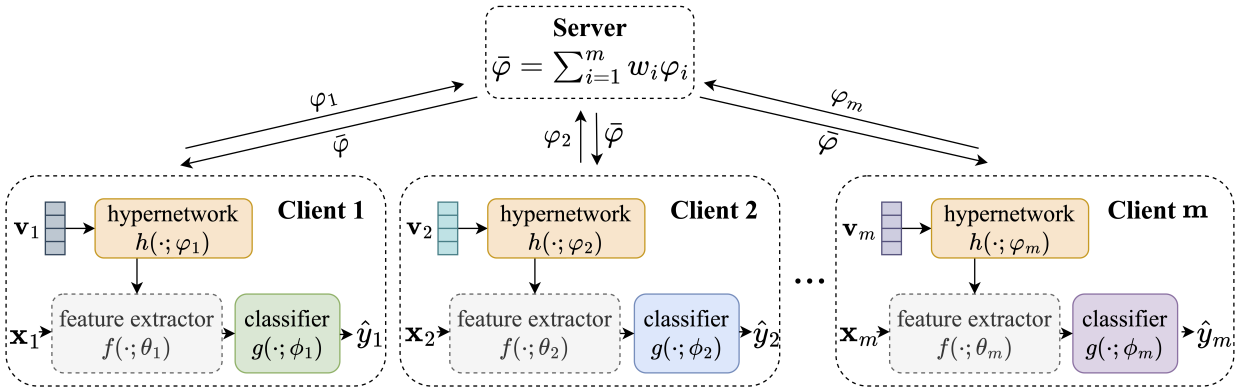


Figure 2: The proposed HyperFL framework. HyperFL decouples each client's network into the former feature extractor $f(;\theta_i)$ and the latter classifier head $g(;\phi_i)$. An auxiliary hypernetwork $h(;\varphi_i)$ is introduced to generate local clients' feature extractor $f(;\theta_i)$ using the client's private embedding vector $\mathbf{v}_i$, i.e., $\theta_i = h(\mathbf{v}_i;\varphi_i)$. These generated parameters are then used to extract features from the input x_i, which are subsequently fed into the classifier to obtain the output $\hat{y}_i$, expressed as $\hat{y}_i = g(f(x_i;\theta_i);\phi_i)$. Throughout the FL training, **only** the hypernetwork φ_i is shared, while all other components are kept private, thus effectively mitigating potential privacy leakage concerns.

addition (Zhu, Liu, and Han 2019; Wei et al. 2020; Huang et al. 2021; Li et al. 2022), Soteria (Sun et al. 2020), PRECODE (Scheliga, Mäder, and Seeland 2022), and FedKL (Ren et al. 2023). However, these methods always suffer from privacy-utility trade-off problems, as illustrated in their papers. In contrast to these approaches, with the help of hypernetworks (Ha, Dai, and Le 2017), this work proposes a novel FL framework that effectively "breaks the direct connection" between the shared parameters and the local private data to defend against GIA while achieving a favorable privacy-utility trade-off.

Hypernetworks in Federated Learning. Hypernetworks (Ha, Dai, and Le 2017) are deep neural networks that generate the weights for another network, known as the target network, based on varying inputs to the hypernetwork. Recently, there have been some works that incorporate hypernetworks into FL for learning personalized models (Shamsian et al. 2021; Carey, Du, and Wu 2022; Li et al. 2023b; Tashakori et al. 2023; Lin et al. 2023). All of these methods adopt the similar idea that a central hypernetwork model are trained on the server to generate a set of models, one model for each client, which aims to generate personalized model for each client. Since the hypernetwork and client embeddings are trained on the server, which makes the server possessing all the information about the local models, enabling the server to recover the original inputs by GIA (see Table 3 and Figure 5 in Appendix). In contrast to existing approaches, this work presents a Hypernetwork Federated Learning (HyperFL) framework, which prioritizes data privacy preservation over personalized model generation through the utilization of hypernetworks.

A more detailed discussion on related work is provided in Appendix.

3 Method

In this section, we first formalize the FL problem, then we present our HyperFL framework.

3.1 Problem Formulation

In FL, suppose there are m clients and a central server, where all clients communicate to the server to collaboratively train their models without sharing raw private data. Each client i is equipped with its own data distribution $P_{XY}^{(i)}$ on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the label space with K categories in total. Let $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$ denotes the loss function given local model Θ_i and data point sampled from $P_{XY}^{(i)}$, then the underlying optimization goal of FL can be formalized as follows

$$\arg\min_{\Theta} \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{(x,y)\sim P_{XY}^{(i)}} \left[\ell\left(\Theta_i; x, y\right)\right], \tag{1}$$

where $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_m\}$ denotes the collection of all local models. In vanilla FL, all clients share the same parameters, i.e., $\Theta_1 = \Theta_2 = \cdots = \Theta_m$. In contrast, personalized FL allows for variation in the parameters across clients, enabling Θ_i to be different for each client.

Since the true underlying data distribution of each client is inaccessible, the common approach to achieving the objective (1) is through Empirical Risk Minimization (ERM). That is, assume each client has access to n_i i.i.d. data points sampled from $P_{XY}^{(i)}$ denoted by $\mathcal{D}_i = \left\{\left(x_i^l, y_i^l\right)\right\}_{l=1}^{n_i}$, whose corresponding empirical distribution is $\hat{P}_{XY}^{(i)}$, and we assume the empirical marginal distribution $\hat{P}_{XY}^{(i)}$ is identical to the true $P_{XY}^{(i)}$. Then the training objective is

$$\arg\min_{\Theta} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_i(\Theta_i), \tag{2}$$

where $\mathcal{L}_i(\Theta_i) = \frac{1}{n_i} \sum_{l=1}^{n_i} \ell(\Theta_i; x_i^l, y_i^l)$ is the local average loss over personal training data, e.g., empirical risk.

3.2 Main Configuration HyperFL

In the Main Configuration HyperFL framework, which is shown in Figure 2, each client i has a classification network parameterized by $\Theta_i = \{\theta_i, \phi_i\}$ consists of a feature extractor $f : \mathcal{X} \to \mathbb{R}^d$ parameterized by θ_i , and a classifier $g : \mathbb{R}^d \to \mathbb{R}^K$ parameterized by ϕ_i, where d is the feature dimension and K is the number of classes. Additionally, each client i has a private client embedding $\mathbf{v}_i$ and a hypernetwork h parameterized by φ_i, which is responsible for generating the parameters of the feature extractor f, i.e., $\theta_i = h(\mathbf{v}_i; \varphi_i)$. In this way, the hypernetwork can generate personalized feature extractor parameters for each client by taking the meaningful client embedding as input. The client embeddings can be trainable vectors or fixed vectors, depending on whether suitable client representations are known in advance. In this work, we adopt trainable vectors. Then, the objective (2) can be reformulated as

$$\arg\min_{\varphi,\phi,\mathbf{v}} \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}_i(h(\mathbf{v}_i;\varphi_i),\phi_i), \tag{3}$$

where $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$, $\phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$, $\mathbf{v} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$. Note that the feature extractor parameters are generated by the hypernetwork and not trainable, whereas the client embedding, the hypernetwork, and the classifier parameters are trainable.

Then, in order to "breaks the direct connection" between the shared parameters and the local private data to defend against GIA while maintaining competitive performance, each client only uploads the hypernetwork parameters to the server for aggregation while keeping the classifier and the private client embedding trained locally. As illustrated in Figure 2, in each FL communication round, each client i uploads its hypernetwork parameters φ_i to the server once the local training is completed while keeps the classifier parameters ϕ_i and client embedding $\mathbf{v}_i$ local to strengthen privacy protection. Then, the server aggregate these φ_i to obtain the global $\bar{\varphi}$. Next, clients download $\bar{\varphi}$ to replace their corresponding local hypernetworks and start the next training iteration. This framework provides a natural way for sharing information across clients while maintaining the privacy of each client, by sharing the hypernetwork parameters. We will elaborate on this workflow in the following.

Local Training Procedure. For local model training at each round, we first replace the local hypernetwork parameters φ_i by the received aggregated hypernetwork parameter $\bar{\varphi}$. Then, we perform stochastic gradient decent steps to iteratively train the model parameters as follows:

- **Step 1: Fix φ_i and $\mathbf{v}_i$, update ϕ_i.** Train the classifier parameters ϕ_i by gradient descent for one epoch:

$$\phi_i \leftarrow \phi_i - \eta_g \nabla_{\phi_i} \ell\left(h(\mathbf{v_i};\varphi_i),\phi_i;\xi_i\right), \tag{4}$$

  where ξ_i denotes the mini-batch of data, η_g is the learning rate for updating the classifier parameters.
- **Step 2: Fix new ϕ_i, update φ_i and $\mathbf{v}_i$.** After getting new classifier, we proceed to update the hypernetwork parameters φ_i and client embedding $\mathbf{v}_i$ for multiple epochs:

$$\begin{aligned}\varphi_i &\leftarrow \varphi_i - \eta_h \nabla_{\varphi_i} \ell\left(h(\mathbf{v_i};\varphi_i),\phi_i;\xi_i\right)\\ \mathbf{v}_i &\leftarrow \mathbf{v}_i - \eta_v \nabla_{\mathbf{v}_i} \ell\left(h(\mathbf{v_i};\varphi_i),\phi_i;\xi_i\right),\end{aligned} \tag{5}$$

  where η_h is the learning rate for updating the hypernetwork parameters and η_v is the learning rate for updating the client embedding.

Global Aggregation. Similar to common FL algorithms, the server performs weighted averaging of the hypernetwork parameters as

$$\bar{\varphi} = \sum_{i=1}^{m} w_i \varphi_i, \tag{6}$$

where w_i is the aggregation weight for client i, usually determined by the local data size, i.e., $w_i = \frac{n_i}{\sum_{i=1}^{m} n_i}$.

3.3 HyperFL-LPM

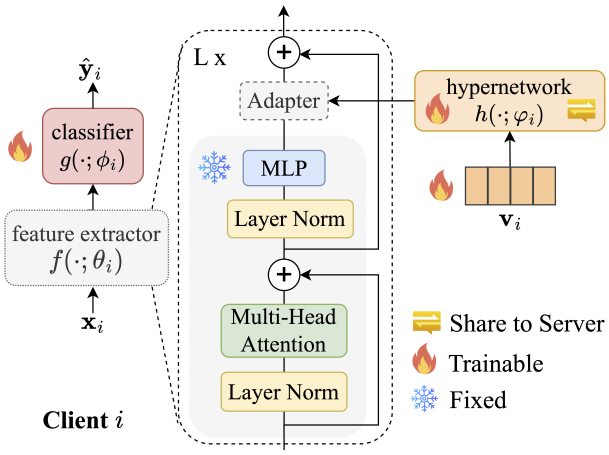


Figure 3: The proposed HyperFL-LPM framework within each client. In this framework, the weights of the pre-trained model are fixed, while only the classifier, hypernetwork, and client embedding are trainable. Note that θ here represents the parameters of the adapters.

When confronted with a large feature extractor, using a hypernetwork to generate the parameters can be challenging. However, in scenarios where the feature extractor is significantly sizable, there are often numerous pre-trained models that are readily available (He et al. 2016; Vaswani et al. 2017; Dosovitskiy et al. 2021; Liu et al. 2021; He et al. 2022). Consequently, we can leverage these pre-trained models and employ parameter-effective fine-tuning techniques to adapt and fine-tune them (Houlsby et al. 2019; Hu et al. 2022; Jia et al. 2022; Guo et al. 2024). To this end, we extend our HyperFL framework to address this situation and propose HyperFL for Large Pre-trained Models (HyperFL-LPM). The difference between HyperFL-LPM and Main Configuration HyperFL is the model adopted withen each client. In the HyperFL-LPM famework, instead of using the hypernetwork to generate the entire feature extractor parameters, we employ it to generate the adapter parameters to fine-tune the pre-trained models. Taking the Transformer-based pre-trained models as an example, the framework within each client is shown in Figure 3. By adopting this approach, our framework can utilize large pre-trained models as fixed feature extractors to handle complex tasks.

4 Analysis and Insights

4.1 Privacy Protection Analysis

In this section, we present a comprehensive privacy analysis of our HyperFL framework. We consider the most common and widely adopted setting, where the server is an *honest-but-curious* adversary, which obeys the training protocol but attempts to obtain the private data of clients according to model weights and updates (Liu et al. 2022; Li et al. 2023a).

Background: Gradient Inversion Attacks. Given a neural network with parameters Θ and the gradients $\nabla_\Theta \mathcal{L}_\Theta(x^*, y^*)$ computed with a private data batch (x^*, y^*), GIA tries to recover x, an approximation of x^* as:

$$\arg\min_x \mathcal{L}_{\text{grad}}\left(x; \Theta, \nabla_\Theta \mathcal{L}_\Theta\left(x^*, y^*\right)\right) + \alpha \mathcal{R}_{\text{aux}}\left(x\right), \quad (7)$$

where $\mathcal{L}_{\text{grad}}\left(x; \Theta, \nabla_\Theta \mathcal{L}_\Theta\left(x^*, y^*\right)\right)$ is a gradient loss term used to enforce matching the gradients of recovered batch x with the provided gradients $\mathcal{L}_\Theta(x^*, y^*)$, $\mathcal{R}_{\text{aux}}\left(x\right)$ is a regularization term utilized to regularize the recovered image based on image priors, and α is a regularization coefficient. The differences between previous works lie in the choice of gradient loss terms and regularization terms (Zhu, Liu, and Han 2019; Geiping et al. 2020; Yin et al. 2021; Luo et al. 2022; Geng et al. 2023; Kariyappa et al. 2023). For example, Zhu et al. (Zhu, Liu, and Han 2019) use ℓ_2-distance as $\mathcal{L}_{\text{grad}}$ but do not use a regularization term $\mathcal{R}_{\text{aux}}$. Geiping et al. (Geiping et al. 2020) adopt cosine similarity as $\mathcal{L}_{\text{grad}}$ and the total variance as $\mathcal{R}_{\text{aux}}$. Luo et al. (Luo et al. 2022) utilize cosine similarity as $\mathcal{L}_{\text{grad}}$ and apply two types of regularization within $\mathcal{R}_{\text{aux}}$: one gradient regularization for fully connected layer and another total variation regularization for convolution features. Geng et al. (Geng et al. 2023) adopt ℓ_2-distance as $\mathcal{L}_{\text{grad}}$ and divide $\mathcal{R}_{\text{aux}}$ into three terms, total variation on the input x, clip and scale operation on the input x.

Analysis on Proposed Leakage Defense. Unlike previous FL models where the entire model parameters are uploaded to the server for aggregation, the HyperFL framework only requires each client to upload the hypernetwork parameters to the server. Therefore, the objective function of an attacker tries to recover x from the HyperFL framework should be changed as:

$$\arg\min_x \mathcal{L}_{\text{grad}}\left(x; \varphi, \nabla_\varphi \mathcal{L}_{\varphi,\phi}\left(x^*, y^*, \mathbf{v}^*\right)\right) + \alpha \mathcal{R}_{\text{aux}}\left(x\right), \quad (8)$$

where φ denotes the hypernetwork parameters, ϕ is the parameters of the classifier, $\mathbf{v}^*$ is the client embedding. Note that there is a major difference between objective (8) and objective (7). In objective (7), the server can obtain gradients of the entire model, while in objective (8), it can only access the gradients of the hypernetwork.

Then, given that x is only exposed to the feature extractor, obtaining the information of the feature extractor is essential to recover x. However, in HyperFL, the parameters of the feature extractor are obtained by inputting the client embedding into the hypernetwork. Therefore, it is necessary to first recover the client embedding. To achieve this pipeline, the objective (8) can be reformulated as a bi-level optimization problem:

$$\begin{aligned} &\arg\min_x \mathcal{L}_{\text{grad}}\left(x, \hat{\mathbf{v}}; \varphi, \Delta_\theta\right) + \alpha \mathcal{R}_{\text{aux}}\left(x\right) \\ &\text{s.t. } \hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \mathcal{L}_{\text{grad}}\left(x, \mathbf{v}; \varphi, \nabla_\varphi \mathcal{L}_{\varphi,\phi}\left(x^*, y^*, \mathbf{v}^*\right)\right), \end{aligned} \quad (9)$$

where $\mathbf{v}$ is an approximation of $\mathbf{v}^*$, θ is the parameters of the feature extractor that generated by the hypernertwork h, i.e., $\theta = h(\mathbf{v}; \varphi)$, $\Delta_\theta = \theta_t - \theta_{t-1}$ serves as an approximation for the gradient of the feature extractor θ (Zhang et al. 2019). However, a challenge arises when solving the lower-level subproblem in objective (9). According to the chain rule, $\nabla_\varphi \mathcal{L}(x, y^*, \mathbf{v}) = \frac{\mathcal{L}(x, y^*, \mathbf{v})}{\partial \phi} \frac{\partial \phi}{\partial \varphi}$. To compute the gradients of the hypernetwork, it is necessary to calculate the gradients of the classifier first [1]. However, since the classifier is trained locally and not shared with the server, it is not feasible to compute these gradients. As a result, the gradients of the hypernetwork cannot be determined, making it challenging to recover the client embedding.

One may question whether we can eliminate the need for the classifier in the process of recovering the client embedding. Drawing inspiration from the GIA procedure (Zhu, Liu, and Han 2019; Li et al. 2023a), we can replace the label information that needs to be optimized with the output of the hypernetwork in this context. By employing this approach, it's able to bypass the requirement of the classifier. Then, the lower-level subproblem in objective (9) will be reformulated as

$$\arg\min_{\mathbf{v},\theta} \mathcal{L}_{\text{grad}}\left(\theta, \mathbf{v}; \varphi, \nabla_\varphi \mathcal{L}_{\varphi,\phi}\left(x^*, y^*, \mathbf{v}^*\right)\right), \quad (10)$$

where θ is the outputs of the hypernetwork. However, previous works have shown that simulating the optimization of both the input and output is challenging (Zhu, Liu, and Han 2019; Zhao, Mopuri, and Bilen 2020; Ma et al. 2023). Therefore, researchers propose to first identify the output and then optimize the input (Zhao, Mopuri, and Bilen 2020; Geiping et al. 2020; Zhu and Blaschko 2021; Yin et al. 2021; Ma et al. 2023; Wang, Liang, and He 2024). Specifically, they can identify the label y^* based on the relationship between the known gradient and label information, as y^* is typically low-dimensional (i.e., a simple one-hot vector) (Zhao, Mopuri, and Bilen 2020; Yin et al. 2021; Ma et al. 2023). However, the output of our hypernetwork (i.e., θ) is high-dimensional and complex. This complexity makes it challenging to identify the ground-truth output θ^* using the known gradient information, and consequently, recovering the embedding becomes difficult. Even if we attempt to optimize both the input and output simultaneously, solving Eq. (10) remains challenging due to the large search space (Zhu, Liu, and Han 2019; Dang et al. 2021; Huang et al. 2021; Kariyappa et al. 2023). Thus, it's challenging to recover the client embedding. Furthermore, even if the client embedding can be recovered (albeit with significant error), the input x is still difficult to recover due to the same problems (i.e., inability to infer the output first and a large search space) encountered when solving the upper-level subproblem in objective (9).

In summary, the HyperFL framework effectively safeguards data privacy, as the combination of the hypernetwork, locally trained classifier, and private client embedding renders the recovery of x using GIA unattainable, which is also demonstrated in our experiments.

[1] We assume the label information can be obtained (Geiping et al. 2020; Zhao, Mopuri, and Bilen 2020; Yin et al. 2021; Ma et al. 2023).

4.2 Convergence Analysis

To facilitate the convergence analysis of HyperFL, we make the assumptions commonly encountered in literature (Li et al. 2020b) to characterize the smooth and non-convex optimization landscape.

Assumption 1. *$\mathcal{L}_1, \cdots, \mathcal{L}_m$ are all L-smooth: for all $(\phi_j, \varphi_j, \mathbf{v}_j)$ and $(\phi_k, \varphi_k, \mathbf{v}_k)$, $\mathcal{L}_i(h(\mathbf{v}_k, \varphi_k), \phi_k) \leq \mathcal{L}_i(h(\mathbf{v}_j, \varphi_j), \phi_j) + ((\phi_k, \varphi_k, \mathbf{v}_k) - (\phi_j, \varphi_j, \mathbf{v}_j))\nabla\mathcal{L}_i(h(\mathbf{v}_j, \varphi_j), \phi_j) + \frac{L}{2}\|(\phi_k, \varphi_k, \mathbf{v}_k) - (\phi_j, \varphi_j, \mathbf{v}_j)\|_2^2$.*

Assumption 2. *Let ξ_i^t be sampled from the i-th client's local data uniformly at random at t-th training step. The variance of stochastic gradients in each client for each variable is bounded: $\mathbb{E}\left\|\nabla_\phi L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^t, \xi_i^t\right) - \nabla_\phi L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^t\right)\right\|^2 \leq \sigma_i^2, \mathbb{E}\left\|\nabla_\varphi L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^{t+1}, \xi_i^t\right) - \nabla_\varphi L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^{t+1}\right)\right\|^2 \leq \sigma_i^2$, $\mathbb{E}\left\|\nabla_{\mathbf{v}} L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^{t+1}, \xi_i^t\right) - \nabla_{\mathbf{v}} L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^{t+1}\right)\right\|^2 \leq \sigma_i^2$ for $i = 1, \cdots, m$.*

Assumption 3. *The expected squared norm of stochastic gradients is uniformly bounded, i.e., $\mathbb{E}\left\|\nabla_\phi L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^t, \xi_i^t\right)\right\|^2 \leq G^2, \mathbb{E}\left\|\nabla_\varphi L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^{t+1}, \xi_i^t\right)\right\|^2 \leq G^2$, $\mathbb{E}\left\|\nabla_{\mathbf{v}} L_i\left(h\left(\mathbf{v}_i^t, \varphi_i^t\right), \phi_i^{t+1}, \xi_i^t\right)\right\|^2 \leq G^2$ for all $i = 1, \cdots, m$ and $t = 0, \cdots, T-1$. Here T denotes the total number of every client's training steps.*

Then we present the convergence rate for HyperFL.

Theorem 1. *Let Assumptions 1, 2 and 3 hold and L, M, σ_i, G be defined therein. Denote $\eta_{\min} = \min\left\{\eta_g, \frac{1}{2}\eta_h, \eta_v\right\}$ and E as the number of local training iterations between two communication rounds. Then we have*

$$\frac{1}{mT}\sum_{i=1}^{m}\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla\mathcal{L}_i^t\right\|^2\right] \leq 2\sqrt{\frac{LMG^2D}{2T}}, \tag{11}$$

where $\mathcal{L}_i^0 - \mathcal{L}_i^* \leq D, \forall i$, and $\eta_g^2 + \eta_v^2 + (E-1)\eta_h^2 + \frac{E-1}{L}\eta_h \leq M\eta_{min}^2$.

According to Theorem 1, we can obtain an $O(\frac{1}{\sqrt{T}})$ convergence rate towards the stationary solution under smooth and non-convex conditions. This convergence rate is comparable to that of FedAvg in the non-convex scenario (Yu, Yang, and Zhu 2019). Further expediting the convergence for Polyak-Lojasiewicz (PL) functions (Karimi, Nutini, and Schmidt 2016) and detailed proofs for these theorems are provided in Appendix.

5 Experiments

5.1 Experimental Setup

Datasets. For the Main Configuration HyperFL, we evaluate our method on four widely-used image classification datasets: (1) EMNIST (Cohen et al. 2017); (2) Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017); (3) CIFAR-10 (Krizhevsky, Hinton et al. 2009); and (4) CINIC-10 (Darlow et al. 2018). For the HyperFL-LPM, we evaluate our method on the EMNIST (Cohen et al. 2017) and CIFAR-10 (Krizhevsky, Hinton et al. 2009) datasets.

Model Architectures. For the Main Configuration HyperFL, simlar to (Xu, Tong, and Huang 2023), we adopt two different CNN target models for EMNIST/Fashion-MNIST and CIFAR-10/CINIC-10, respectively. For the HyperFL-LPM, we adopt the ViT (Dosovitskiy et al. 2021) and ResNet (He et al. 2016) pre-trained on the ImageNet dataset (Deng et al. 2009) as the feature extractor. The hypernetworks of HyperFL and HyperFL-LPM both are a fully-connected neural network with one hidden layer, multiple linear heads per target weight tensor. The client embeddings are learnable vectors with dimension equals 64.

Compared Methods. For the Main Configuration HyperFL, we compare the proposed method with the following approaches: (1) Local-only; (2) FedAvg (McMahan et al. 2017); (3) pFedHN (Shamsian et al. 2021); and some DP-based FL methods, including (4) DP-FedAvg (McMahan et al. 2018); (5) PPSGD (Bietti et al. 2022); and (6) CENTAUR (Shen et al. 2023). For the HyperFL-LPM, we compare our method with (1) Local-only with fixed feature extractor; (2) Local-only with adapter fine-tuning; (3) FedAvg with fixed feature extractor; and (4) FedAvg with adapter fine-tuning.

Training Settings. We employ the mini-batch SGD (Ruder 2016) as a local optimizer for all approaches, and the number of local training epochs is set to 5. The number of global communication rounds is set to 200 for all datasets. Average test accuracy of all local models is reported for performance evaluation.

For privacy evaluation, we adopt the widely used IG (Geiping et al. 2020), state-of-the-art ROG (Yue et al. 2023), and a tailored attack method for our defense framework to recover the input images. More details about experimental setup are provided in Appendix.

5.2 Experimental Results

Performance Evaluation. As demonstrated in Table 1, the performance of all the compared DP-based FL methods is inferior to FedAvg and Local-only. This is due to the incorporation of DP mechanisms, which adversely affect model usability and result in decreased performance. In contrast, our proposed HyperFL consistently surpasses these methods across various datasets, demonstrating its outstanding utility. Notably, HyperFL excels in both situations where Local-only outperforms (i.e., Fashion-MNIST and CINIC-10) and where FedAvg prevails (i.e., EMNIST and CIFAR-10). This further highlights HyperFL's adaptability, excelling in both centralized FL scenarios and cases requiring personalization. Furthermore, the learned client embeddings, which are meaningful, can be found in the Appendix. Although pFedHN outperforms our method in two scenarios, it exhibits poor defense capability against GIA, as illustrated in Table 3.

The performance of HyperFL-LPM compared with Local-only and FedAvg is shown in Table 2. From this table, we can see that HyperFL-LPM can achieve comparable performance to baseline adapter fine-tuning methods with different pre-trained models, regardless of whether Local-only or FedAvg performs better. This highlights the effectiveness of HyperFL-LPM.

Method	EMNIST		Fashion-MNIST		CIFAR-10		CINIC-10	
	20-C	100-C	20-C	100-C	20-C	100-C	20-C	100-C
Local-only	73.41	75.68	85.93	87.01	65.47	66.11	63.60	64.84
FedAvg	72.77	78.87	85.67	88.11	70.02	76.24	57.00	59.11
pFedHN	**80.86**	77.37	87.64	89.80	70.18	**80.07**	63.88	70.36
DP-FedAvg	35.12	45.73	59.88	68.29	29.12	32.03	27.30	29.94
CENTAUR	68.82	67.24	83.07	79.77	50.85	51.86	48.82	51.01
PPSGD	71.16	71.18	84.47	82.94	52.17	53.92	49.98	52.91
HyperFL	76.29	**80.22**	**88.28**	**90.41**	**73.03**	78.73	**66.74**	**72.21**

Table 1: The comparison of final test accuracy (%) of different methods on various datasets. We apply full participation for FL system with 20 clients (20-C), and apply client sampling with rate 0.3 for FL system with 100 clients (100-C).

	Arch	Local-only†	Local-only††	FedAvg†	FedAvg††	HyperFL-LPM
EMNIST	ResNet	72.83	80.35	68.99	75.21	80.32
	ViT	76.95	80.04	70.92	76.42	79.92
CIFAR-10	ResNet	68.57	73.57	62.35	75.57	75.03
	ViT	91.82	89.70	92.32	95.56	95.40

Table 2: The comparison of final test accuracy (%) of different methods on various datasets with 20 clients. † Fixed feature extractor. †† Adapter fine-tuning.

Privacy Evaluation. The reconstructed results of IG (Geiping et al. 2020) are provided in Table 3, while more results of ROG (Yue et al. 2023) and tailored attack method are provided in Table 5 and Figures 6 and 7 in Appendix. From Table 3, we can observe that the native FedAvg, pFedHN, and pFedHN-PC methods have a much higher risk of leaking data information (indicated by the higher PSNR and SSIM values and lower LPIPS value). This can also be seen in the reconstructed images, which closely resemble the original ones, as illustrated in Figure 5 in Appendix. Although introducing DP improves data privacy, there is a significant drop in model performance, as shown in Table 1. In contrast, HyperFL achieves a similar level of privacy protection while outperforming all DP-based methods and the native FedAvg in terms of model accuracy.

Method	EMNIST			CIFAR-10		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
FedAvg	32.64	0.8925	0.0526	16.16	0.6415	0.0536
pFedHN	31.24	0.8701	0.0807	16.02	0.6351	0.0504
pFedHN-PC	28.38	0.8713	0.0645	15.80	0.6247	0.4407
DP-FedAvg	7.74	0.2978	0.7051	7.90	0.2716	0.3204
CENTAUR	9.52	0.2136	0.6712	9.80	0.2723	0.2882
PPSGD	9.73	0.1889	0.6466	9.70	0.2788	0.2643
HyperFL	7.85	0.3010	0.7147	8.35	0.2732	0.3132

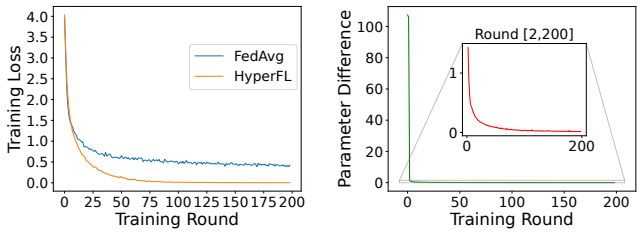
Table 3: Reconstruction results of IG.

Training Efficiency. To validate the training efficiency of the proposed HyperFL framework, we compare the training time of HyperFL with other DP-based FL methods in Table 4. This table clearly shows the efficiency of the proposed HyperFL framework. Specifically, from this table we can see that the proposed HyperFL framework runs faster than all the compared DP-based FL methods and only slightly slower than the FedAvg method. This is because DP-based FL methods often incur additional computation cost due to their privacy-preserving mechanisms, whereas HyperFL achieves faster training by leveraging the advantages of hypernetworks, all while ensuring data privacy.

	FedAvg	DP-FedAvg	PPSGD	CENTAUR	HyperFL
# Time (s)	23	194	223	210	37

Table 4: Training time of per training round on the EMNIST dataset with 20 clients of different methods.

Convergence Evaluation. To validate to convergence of the proposed HyperFL framework, we draw the training loss of FedAvg and HyperFL in Figure 4(a) and the trend of feature extractor parameters' variation in Figure 4(b). From Figure 4(a), we can observe that HyperFL almost has the same convergence rate as FedAvg, which demonstrates the convergence property of HyperFL. Moreover, after convergence, the training loss of HyperFL is lower than that of FedAvg, which reflects why HyperFL performs better than FedAvg. Furthermore, in the later stages of the training process, the variation of the feature extractor parameters approaches zero, as depicted in Figure 4(b). This further confirms the convergence property of HyperFL.



(a) (b)

Figure 4: (a) Average training loss of different methods on the EMNIST dataset with 20 clients. (b) Parameter difference of the generated feature extractor of one client between adjacent training round on the EMNIST dataset with 20 clients.

6 Conclusion

In this paper, we propose HyperFL, a novel federated learning framework that "breaks the direct connection" between the shared parameters and the local private data to defend against GIA. Specifically, this framework utilizes hypernetworks to generate the parameters of the local model and only the hypernetwork parameters are uploaded to the server for aggregation to defend against GIA while without compromising performance or incurring heavy computation overhead. We hope that the proposed HyperFL framework can encourage the research community to consider the importance of developing enhanced privacy preservation FL frameworks, as an alternative to current research efforts on defense mechanisms front.

Acknowledgments

This work was supported by National Natural Science Foundation of China (62306253, 62206075), Guangdong Natural Science Fund-General Programme (2024A1515010233), and UCSC hellman fellowship.

References

Bietti, A.; Wei, C.-Y.; Dudik, M.; Langford, J.; and Wu, S. 2022. Personalization improves privacy-accuracy tradeoffs in federated learning. *ICML*.

Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. *CCS*.

Carey, A. N.; Du, W.; and Wu, X. 2022. Robust Personalized Federated Learning under Demographic Fairness Heterogeneity. *IEEE BigData*.

Cohen, G.; Afshar, S.; Tapson, J.; and Van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. *IJCNN*.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting shared representations for personalized federated learning. *ICML*.

Dang, T.; Thakkar, O.; Ramaswamy, S.; Mathews, R.; Chin, P.; and Beaufays, F. 2021. Revealing and protecting labels in distributed training. *NeurIPS*.

Darlow, L. N.; Crowley, E. J.; Antoniou, A.; and Storkey, A. J. 2018. Cinic-10 is not imagenet or cifar-10. *arXiv:1810.03505*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. *CVPR*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*.

Dwork, C. 2006. Differential privacy. *ICALP*.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. *CCS*.

Geiping, J.; Bauermeister, H.; Dröge, H.; and Moeller, M. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *NeurIPS*.

Geng, J.; Mou, Y.; Li, Q.; Li, F.; Beyan, O.; Decker, S.; and Rong, C. 2023. Improved Gradient Inversion Attacks and Defenses in Federated Learning. *TBD*.

Gentry, C. 2009. *A fully homomorphic encryption scheme*. Stanford university.

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

Guo, P.; Zeng, S.; Wang, Y.; Fan, H.; Wang, F.; and Qu, L. 2024. Selective Aggregation for Low-Rank Adaptation in Federated Learning. *arXiv:2410.01463*.

Ha, D.; Dai, A. M.; and Le, Q. V. 2017. HyperNetworks. *ICLR*.

Hatamizadeh, A.; Yin, H.; Molchanov, P.; Myronenko, A.; Li, W.; Dogra, P.; Feng, A.; Flores, M. G.; Kautz, J.; Xu, D.; et al. 2023. Do gradient inversion attacks make federated learning unsafe? *TMI*.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *CVPR*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. *ICML*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*.

Huang, Y.; Gupta, S.; Song, Z.; Li, K.; and Arora, S. 2021. Evaluating gradient inversion attacks and defenses in federated learning. *NeurIPS*.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual prompt tuning. *ECCV*.

Karimi, H.; Nutini, J.; and Schmidt, M. 2016. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. *ECML-PKDD*.

Kariyappa, S.; Guo, C.; Maeng, K.; Xiong, W.; Suh, G. E.; Qureshi, M. K.; and Lee, H.-H. S. 2023. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. *ICML*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *NeurIPS*.

Li, B.; Gu, H.; Chen, R.; Li, J.; Wu, C.; Ruan, N.; Si, X.; and Fan, L. 2023a. Temporal Gradient Inversion Attacks with Robust Optimization. *arXiv:2306.07883*.

Li, H.; Cai, Z.; Wang, J.; Tang, J.; Ding, W.; Lin, C.-T.; and Shi, Y. 2023b. FedTP: Federated Learning by Transformer Personalization. *TNNLS*.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020a. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020b. On the Convergence of FedAvg on Non-IID Data. *ICLR*.

Li, Z.; Zhang, J.; Liu, L.; and Liu, J. 2022. Auditing privacy defenses in federated learning via generative gradient leakage. *CVPR*.

Lin, Y.; Wang, H.; Li, W.; and Shen, J. 2023. Federated learning with hyper-network—A case study on whole slide image analysis. *Scientific Reports*.

Liu, Z.; Guo, J.; Yang, W.; Fan, J.; Lam, K.-Y.; and Zhao, J. 2022. Privacy-preserving aggregation in federated learning: A survey. *TBD*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*.

Luo, Z.; Zhu, C.; Fang, L.; Kou, G.; Hou, R.; and Wang, X. 2022. An effective and practical gradient inversion attack. *IJIS*.

Ma, J.; Naas, S.-A.; Sigg, S.; and Lyu, X. 2022. Privacy-preserving federated learning based on multi-key homomorphic encryption. *IJIS*.

Ma, K.; Sun, Y.; Cui, J.; Li, D.; Guan, Z.; and Liu, J. 2023. Instance-wise Batch Label Restoration via Gradients in Federated Learning. *ICLR*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. *AISTATS*.

McMahan, H. B.; Ramage, D.; Talwar, K.; and Zhang, L. 2018. Learning Differentially Private Recurrent Language Models. *ICLR*.

Mou, W.; Fu, C.; Lei, Y.; and Hu, C. 2021. A verifiable federated learning scheme based on secure multi-party computation. *WASA*.

Mugunthan, V.; Polychroniadou, A.; Byrd, D.; and Balch, T. H. 2019. Smpai: Secure multi-party computation for federated learning. *NeurIPS Workshop*.

Park, J.; and Lim, H. 2022. Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences*.

Phong, L. T.; Aono, Y.; Hayashi, T.; Wang, L.; and Moriai, S. 2017. Privacy-preserving deep learning: Revisited and enhanced. *ATIS*.

Qu, L.; Zhou, Y.; Liang, P. P.; Xia, Y.; Wang, F.; Adeli, E.; Fei-Fei, L.; and Rubin, D. 2022. Rethinking architecture design for tackling data heterogeneity in federated learning. *CVPR*.

Ren, H.; Deng, J.; Xie, X.; Ma, X.; and Ma, J. 2023. Gradient leakage defense with key-lock module for federated learning. *arXiv:2305.04095*.

Ruder, S. 2016. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*.

Scheliga, D.; Mäder, P.; and Seeland, M. 2022. Precode-a generic model extension to prevent deep gradient leakage. *WACV*.

Shamsian, A.; Navon, A.; Fetaya, E.; and Chechik, G. 2021. Personalized federated learning using hypernetworks. *ICML*.

Shen, Z.; Ye, J.; Kang, A.; Hassani, H.; and Shokri, R. 2023. Share your representation only: Guaranteed improvement of the privacy-utility tradeoff in federated learning. *ICLR*.

Sun, J.; Li, A.; Wang, B.; Yang, H.; Li, H.; and Chen, Y. 2020. Provable defense against privacy leakage in federated learning from representation perspective. *arXiv:2012.06043*.

Tashakori, A.; Zhang, W.; Wang, Z. J.; and Servati, P. 2023. SemiPFL: personalized semi-supervised federated learning framework for edge intelligence. *IOT*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *NeurIPS*.

Wang, Y.; Liang, J.; and He, R. 2024. Towards Eliminating Hard Label Constraints in Gradient Inversion Attacks. *ICLR*.

Wang, Y.; Wang, X.; Dinh, A.-D.; Du, B.; and Xu, C. 2023. Learning to Schedule in Diffusion Probabilistic Models. *KDD*.

Wei, W.; Liu, L.; Loper, M.; Chow, K.-H.; Gursoy, M. E.; Truex, S.; and Wu, Y. 2020. A framework for evaluating gradient leakage attacks in federated learning. *arXiv:2004.10397*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*.

Xu, J.; Tong, X.; and Huang, S.-L. 2023. Personalized Federated Learning with Feature Alignment and Classifier Collaboration. *ICLR*.

Xu, Y.; Peng, C.; Tan, W.; Tian, Y.; Ma, M.; and Niu, K. 2022. Non-interactive verifiable privacy-preserving federated learning. *FGCS*.

Yao, A. C. 1982. Protocols for secure computations. *23rd annual symposium on foundations of computer science (sfcs 1982)*.

Yin, H.; Mallya, A.; Vahdat, A.; Alvarez, J. M.; Kautz, J.; and Molchanov, P. 2021. See through gradients: Image batch recovery via gradinversion. *CVPR*.

Yu, H.; Yang, S.; and Zhu, S. 2019. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. *AAAI*.

Yu, T.; Bagdasaryan, E.; and Shmatikov, V. 2020. Salvaging federated learning by local adaptation. *arXiv:2002.04758*.

Yue, K.; Jin, R.; Wong, C.-W.; Baron, D.; and Dai, H. 2023. Gradient obfuscation gives a false sense of security in federated learning. *USENIX Security*.

Zeng, S.; Guo, P.; Wang, S.; Wang, J.; Zhou, Y.; and Qu, L. 2024. Tackling data heterogeneity in federated learning via loss decomposition. *MICCAI*.

Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; and Liu, Y. 2020a. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. *USENIX ATC*.

Zhang, J.; Zeng, S.; Zhang, M.; Wang, R.; Wang, F.; Zhou, Y.; Liang, P. P.; and Qu, L. 2024. FLHetBench: Benchmarking Device and State Heterogeneity in Federated Learning. *CVPR*.

Zhang, M.; Lucas, J.; Ba, J.; and Hinton, G. E. 2019. Lookahead optimizer: k steps forward, 1 step back. *NeurIPS*.

Zhang, X.; Fu, A.; Wang, H.; Zhou, C.; and Chen, Z. 2020b. A privacy-preserving and verifiable federated learning scheme. *ICC*.

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

Zhu, J.; and Blaschko, M. B. 2021. R-GAP: Recursive Gradient Attack on Privacy. *ICLR*.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep leakage from gradients. *NeurIPS*.