Mimir: Data-Free Federated Unlearning Through Client-Specific Prompt Generation for Personalized Models

Wenhan Wu, *Student Member, IEEE*, Huanghuang Liang, Tianyu Tu, Jiawei Jiang, *Member, IEEE*, Chuang Hu, *Member, IEEE*, and Dazhao Cheng, *Senior Member, IEEE*

***Abstract*—Federated unlearning (FU) has become an important area of research due to an increasing need for federated learning (FL) applications to comply with emerging data privacy regulations such as GDPR. It facilitates the removal of certain clients' data from an already trained FL model while preserving the performance on the remaining client without the need to retrain from scratch. Existing FU methods typically require clients to have access to their training data or historical model updates, which may be impractical in real-world scenarios due to privacy constraints and changes in data availability. Moreover, FU methods may cause catastrophic unlearning, where removing a client's data from heterogeneous, non-IID settings can negatively impact the model's performance on data from retained clients. To address the aforementioned issues and leverage the capabilities of personalized federated learning (pFL) in handling non-IID data distributions, this paper introduce Mimir, a novel data-free federated unlearning framework designed for pFL settings. Mimir integrates both learning and unlearning phases by utilizing personalized prompts for each client. We design a distillation structure based on Generative Adversarial Networks (GANs) for client-level unlearning that does not require access to original data or historical updates. By leveraging client-specific prompts generated during the pFL phase, Mimir adapts to heterogeneous data distributions and mitigates catastrophic unlearning on the retained data. We demonstrate the effectiveness of Mimir through extensive experiments on benchmark datasets, showing its ability to forget target client data while preserving model accuracy on the remaining clients.**

***Index Terms*—Federated unlearning, personalized federated learning, data-free machine unlearning, prompt learning.**

I. Introduction

FEDERATED Unlearning (FU) [1], [2] has recently emerged as a critical research area within federated learning (FL) [3], addressing the challenge of removing the influence of specific data from a trained federated model while preserving the performance on the retained data. This process is driven by privacy concerns, compliance requirements, or user requests. For instance, in the FL paradigm, clients collaboratively train a shared model without exchanging their local data directly. However, regulations such as GDPR [4] mandate that trained models support data deletion requests. If a client decides to withdraw from the FL system, either to exercise their "*right to be forgotten*" or to address accidental errors in their data, a suitable FU mechanism should be applied to remove the influence of that client's data without resorting to retraining from scratch. While a straightforward solution would be to retrain the model solely on the remaining data, this approach is computationally expensive and will result in substantial resource waste [5].

Existing FU methods focus on one of the three levels of granularity: class unlearning [6], client unlearning [1], [7], and sample unlearning [2]. These approaches modify the trained model to simulate the scenario where the forgotten class, client, or sample data was never encountered. Since unlearning requests are typically initiated by clients, this paper focuses on client-level FU. Existing client-level FU methods rely on client access to historical training data, such as gradient updates [1], [8], or original and target unlearning data [2], [7], [9] to accelerate retraining or fine-tune pre-trained models with a few additional rounds. However, they are predominantly designed for a single global model in standard FL settings and lack tailored solutions for *personalized federated learning (pFL)* [10] in heterogeneous data environments. In real-world scenarios, client data often exhibits significant heterogeneity in both feature and label spaces [11], resulting in non-IID data distributions. Traditional FL approaches, which aim to train a single global model, may suffer from suboptimal performance [12] under such conditions. pFL addresses this challenge by training personalized models for each client, accounting for their unique data distributions. Despite these advancements, implementing effective unlearning methods in personalized environments remains largely unexplored. Current strategies have yet to address the complexities of FU across models with personalized variations. Intuitively, the client-specific personalized features extracted during the training process could potentially facilitate the client-level unlearning process. This gap presents an opportunity for integrating pFL with FU. In this paper, we bridge this gap and aim to overcome the following two challenges:

The first challenge arises from *data unavailability*. While local data is accessible to clients during FL training, it may

Received 17 December 2024; revised 13 April 2025; accepted 9 May 2025. Date of publication 14 May 2025; date of current version 3 September 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62341410 and Grant 62302348, and in part by the National Key Research, Development Program of China under Grant 2023YFE0205700. Recommended for acceptance by D. Niyato. *(Corresponding authors: Chuang Hu; Dazhao Cheng.)*

The authors are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: wenhanwu@whu.edu.cn; hhliang@whu.edu.cn; meetnailtu30@whu.edu.cn; jiawei.jiang@whu.edu.cn; handc@whu.edu.cn; dcheng@whu.edu.cn).

Digital Object Identifier 10.1109/TMC.2025.3570018

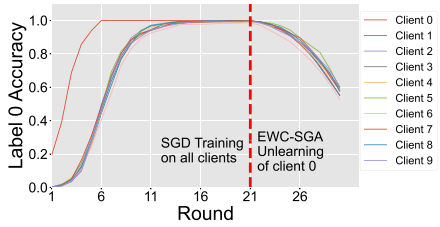


Fig. 1. Catastrophic unlearning of the EWC-SGA method on Non-IID MNIST dataset for label 0.

become unavailable during unlearning. Clients may move or delete training samples and historical updates, or lose access rights to the data after the model has been trained [13], [14] due to privacy concerns, legal regulations, and organizational policies. For example, sensitive local data such as medical records or facial images may be temporarily accessible with user consent but must be deleted after FL training at the user's request. Additional data access for unlearning can incur significant costs or may even be unauthorized. Besides, storing historical updates is also impractical due to storage limitations. Clients typically retain only the trained model after training and forgotten clients are often reluctant to contribute data for unlearning. As a result, existing FU methods may struggle to meet client removal requirements, highlighting the development of *data-free* FU frameworks to effectively remove the influence of client data *without client access to the historical updates and training samples*.

The second challenge is "*catastrophic unlearning*" [2], [15], [16] during the FU process, caused by heterogeneous data distributions and cross-labels across clients. In heterogeneous FL, the data held by different clients often varies significantly, which can negatively affect unlearning performance. For example, as shown in Fig. 1, client 0 mainly holds a small portion of label 0 data from the MNIST [17] dataset. After 20 rounds of training and 10 rounds of unlearning, removing client 0's data should theoretically have minimal impact on label 0 performance. However, existing FU methods like EWC-SGA [2] often cause catastrophic forgetting of label 0 data from other clients in non-IID scenarios, drastically reducing generalization from 99.26% to 57.92%. This highlights the challenge of preventing catastrophic unlearning in heterogeneous data distributions.

In this paper, we introduce Mimir,[1] a data-free FU framework designed for non-IID data distributions. Mimir leverages personalized feature information from pFL to enable client-level data-free unlearning. During the pFL phase, inspired by prompt learning [18], [19], [20], Mimir enhances the model with trainable parameters, called *prompts*, to personalize it for each client. These client-specific descriptors help generate personalized prompts that adapt the model to heterogeneous data distributions. In the FU phase, Mimir uses a Generative Adversarial Network (GAN)-based distillation structure to perform unlearning without access to original data or historical updates. By using client-specific prompts from the learning phase, Mimir eliminates personalized feature extraction capabilities in unlearning clients, achieving data-free unlearning. By utilizing pFL's ability to map similar data from different clients into distinct feature spaces, Mimir also mitigates catastrophic unlearning of identical-label data across clients. We evaluate Mimir on benchmark datasets using standard FU metrics, demonstrating its effectiveness. The main contributions of this paper are as follows:

- To our knowledge, this is the first work addressing the challenge of client-level data-free unlearning, which addresses real-world data deletion challenges, where clients cannot access their training data or historical updates.
- We propose a novel framework that uses pFL to enhance unlearning. Mimir adapts the model to heterogeneous data distributions using trainable prompts, ensuring efficient personalization, which also establishes the foundation for FU and mitigates catastrophic unlearning.
- We present an effective data-free unlearning method combining knowledge distillation and GANs to modify personalized feature extraction capabilities.
- We have implemented and deployed Mimir. Evaluations on benchmark datasets demonstrate its effectiveness in forgetting target client data and preserving model performance on retained data in non-IID settings.

II. Related Work

A. Personalized Federated Learning (pFL)

Personalized Federated Learning (pFL) addresses the challenges posed by data heterogeneity in Federated Learning (FL), where clients often have diverse and non-IID (non Independent and Identically Distributed) data. Traditional Federated Learning [3] facilitates the sharing of a global model across clients while ensuring privacy protection, but real-world data heterogeneity can significantly hinder model aggregation performance [21], [22]. In response, pFL [10], [11] seeks to customize the global model by learning personalized models for each client. Several approaches to pFL focus on decoupling global and client-specific information to address the privacy and heterogeneity challenges. For instance, GPFL [23] and FedCP [24] employ conditional strategies to separate private and global information, enabling better personalization. pFedMo [25] uses adaptive local aggregation methods to tailor updates at the client level, enhancing the model's ability to adapt to each client's unique data. pFedGate [26] takes a more direct approach by modifying client model parameters to further refine personalization. Per-FedAvg [27] incorporates meta-learning to personalize models across heterogeneous clients, while FedProto [28] leverages prototype learning to handle non-IID data, achieving more robust performance in highly diverse settings. The pFL method in this paper is inspired by prompt learning techniques [18], [19], [20], adapting domain-heterogeneous FL approach from Vision Transformers (ViT) [20] to a general pFL model and modifying the prompts from concatenation to Hadamard product. These

[1]The guardian of the well of wisdom in Norse mythology, whose water bestows wisdom but also causes forgetting.

pFL approaches highlight the importance of tailoring federated learning systems to better meet the needs of individual clients, thereby improving the overall performance in heterogeneous environments.

B. Machine and Federated Unlearning

Machine unlearning removes the contribution of specific class-level or sample-level data from trained models without the need to retrain from scratch, which is crucial for data privacy and regulatory compliance. Precise methods, such as SISA [5], achieve this by quickly retraining data partitions, while approximate methods [29] utilize the statistical properties of model parameters or outputs. Traditional unlearning methods require access to historical parameters or original data. To address this, GKT [13] introduces zero-shot unlearning via gated knowledge distillation, and GENIU [14] employs variational autoencoders (VAE) to train an agent generator alongside the original model for data-restricted unlearning on imbalanced datasets.

Machine unlearning typically focuses on removing specific data points in a centralized model, at the class or sample level. Federated unlearning, as an extension of machine unlearning in distributed scenarios, additionally introduces client-level, which aims to efficiently remove specific client contributions on the trained model. FedEraser [1] leverages historical updates' magnitudes to accelerate retraining. Other unlearning methods include employing a quasi-Newton method for rapid retraining [9], and using an asynchronous method for large-scale client unlearning [30]. These methods reduce computational costs but still require training data or historical updates. Challenges in federated unlearning stem from the inability to directly access client data due to its isolated storage, as well as from distributional issues, such as the challenge of class overlap across clients leading to "catastrophic unlearning" [15], [16]. A recent survey [31] provides a comprehensive analysis of federated unlearning frameworks, emphasizing the inherent trade-offs between privacy guarantees, model accuracy, and computational efficiency. Their proposed benchmark highlights the challenges of balancing these objectives, particularly in scenarios with heterogeneous data distributions and strict privacy requirements. In the context of FL, the training data is not directly visible to the central server; however, due to the independence of the clients, data unavailability becomes more pronounced during the unlearning phase. This is because, in the unlearning process, the data might be deleted on the client side or the client's access to it might be revoked [13]. Despite its importance, the issue of unlearning in data-free scenarios, where the client forgets specific information without access to the original data, remains largely unexplored.

III. Preliminaries

We first summarize the primary notations used in the paper, as presented in Table I. Then we introduce the background knowledge of pFL and data-free federated unlearning.

TABLE I
Basic Notations

Notation	Description
$\mathcal{D}_i$	Local dataset of client i
$\mathcal{D}_f$	Dataset on the forgetting client C_f
$\mathcal{D}_r$	Dataset on the retained client C_r
$\mathrm{Dir}(\zeta)$	Dirichlet distribution, where ζ controls the non-IID degree of client data
$\omega_g, \omega_{p,i}$	Global and personalized parameters of client i
$DG(z;\omega_{DG})$	Data generator (input noise z, parameters ω_{DG})
P_{base}	Client-independent base prompt
d_i	Descriptor of client i
$G_p(\cdot;\omega_{G_p})$	Prompt generator with parameters ω_{G_p}
θ, φ	Feature extractor (θ) and model head (φ)
$\mathcal{L}_{CE}$	Cross-entropy loss (for classification tasks)
$\mathcal{L}_K$	KL divergence loss (for knowledge distillation alignment)
$\mathcal{L}_{att}$	Attention loss (for intermediate feature alignment)
$\mathcal{L}_F$	Feature similarity loss (between forgetting client and student)
$\mathcal{L}_{GP}$	Loss of the global prompt generator (including L2 regularization)
lr	Learning rate
τ	Temperature parameter (smoothing factor in KL divergence loss)
λ_1, λ_2	L2 regularization coefficients
β, γ	Distillation loss weights

TABLE II
Details of Datasets

Dataset	Training	Testing	Shape	Description
MNIST	60,000	10,000	28x28	handwritten digits (0-9)
SVHN	73,257	26,032	32x32	real-world numbers (0-9)
FMNIST	60,000	10,000	64x64	10 categories of clothing
CIFAR10	50,000	10,000	32x32	10 categories of nature

A. Personalized Federated Learning (pFL)

We describe the general form of the pFL problem. Consider K clients $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$, where each client C_i possesses its own private dataset $D_i = \{(x_i^j, y_i^j)\}_{j=1}^{n_i}$. Here, x_i^j represents the j-th data sample and y_i^j is its label. Traditional FL assumes identical data structures and feature spaces across clients, aiming to collaboratively train a unified model. However, real-world data often exhibits non-IID characteristics, resulting in suboptimal performance for a single global model [11]. pFL addresses this by training personalized models for each client while maintaining collaborative training. The personalized loss for C_i is:

$$\mathcal{L}_i(\omega_i) = \frac{1}{|D_i|} \sum_{(\boldsymbol{x}_j^i, y_j^i) \in D_i} \mathcal{L}\left(\mathcal{F}_i(\boldsymbol{x}_j^i; \omega_g, \omega_{p,i}), y_j^i\right) + \lambda_g \|\omega_g - \omega_{\text{global}}\|_2^2 + \lambda_l \|\omega_{p,i}\|_2^2, \quad (1)$$

where $\mathcal{L}(\cdot,\cdot)$ denotes the task-specific loss function (e.g., cross-entropy for classification), $|D_i|$ is the number of samples in D_i, $\mathcal{F}_i$ is the client-specific model on client C_i, with $\omega_i = (\omega_g, \omega_{p,i})$ denoting its parameters, where ω_g refers to the shared global parameters and $\omega_{p,i}$ represents the local personalized parameters capturing client-specific characteristics. $\lambda_g > 0$ and $\lambda_l > 0$ are hyperparameters balancing alignment with the server's global model ω_{global} and regularization of the personalized parameters, respectively.

The first term, the local empirical risk, ensures that the model fits the client's private data. The collaborative regularization term $\lambda_g \|\omega_g - \omega_{\text{global}}\|_2^2$ constrains the deviation of the local

global parameters ω_g from the server's aggregated ω_{global}, facilitating knowledge transfer across clients. The term $\lambda_l \|\omega_{p,i}\|_2^2$ prevents overfitting of the personalized parameters $\omega_{p,i}$ to local data, ensuring a balance between personalization and generalization. The overall objective of pFL is to jointly optimize the parameters:

$$\arg\min_{\omega_g, \{\omega_{p,i}\}_{i=1}^K} \sum_{i=1}^{K} \frac{|D_i|}{N} \mathcal{L}_i(\omega_g, \omega_{p,i}), \tag{2}$$

where $N = \sum_{i=1}^{K} |D_i|$. Equations (1) and (2) encapsulate the dual goals of personalization (via $\omega_{p,i}$) and global collaboration (via ω_g) in the pFL framework.

B. Data-Free Federated Client Unlearning

We present the mathematical objectives for the unlearning problem. Given a training dataset $D = \{D_i\}_{i=1}^K$ from clients $C_1, C_2, \ldots, C_K$, let C_r be the client whose data should be retained, and D_r be its data. Let C_f be the client whose influence C_r wants to eliminate, and D_f its data. In extreme non-IID data scenarios, a single client C_f might uniquely hold almost all data of a specific class, reducing client unlearning to class unlearning. However, in practical cases, data of a particular class is often shared among multiple clients, i.e., $y_f \cup y_r \neq \emptyset$. Careful design is required to unlearn specific clients' data without significantly degrading the performance of the remaining clients. We denote a model trained with retained federated clients from scratch without observing the D_r as the *retrained model*, which is typically used as a benchmark for performance metrics in unlearning methods [1], [5]. We define $\mathcal{FL} : D \to \omega$ as a FL process that maps the client data space D to the hyperparameter space of the global model ω. We define the unlearning process $\mathcal{FU} : \mathcal{FL}(D) \otimes D \otimes D_f \to \omega'$, where the input consists of the participating client data space D, a learned model $\mathcal{FL}(D)$, and the data D_f on client C_f to be forgotten. The output is an unlearned model that is similar to the retrained model. The objective of federated client unlearning is:

$$\Phi\left[\mathcal{FL}(D \setminus D_f)\right] = \Phi\left[\mathcal{FU}(\mathcal{FL}(D), D, D_f)\right]. \tag{3}$$

where $D \setminus D_f$ denotes the client data space excluding D_f, and $\Phi[\cdot]$ indicates the probability distribution of the output. The Kullback Leibler (KL) divergence [32] is commonly used as a measure of difference between two probability distributions $p(x)$ and $q(x)$, which is denoted as below:

$$\text{KL}(p(x)\|q(x)) = \mathbb{E}_{x\sim p(x)}\left[\log p(x)/q(x)\right], \tag{4}$$

General methods for federated unlearning typically require clients to access D_f and D_r during the unlearning process. However, due to privacy protection requirements, such as cases where clients' access to data is revoked by users or clients delete training data to conserve storage space, accessing historical updates or training data is infeasible for the client. If $\mathcal{FU}$ does not require clients to access D_r, D_f or historical updates but instead relies on querying the models on the forgotten and retained clients, it is termed a *Data-free Client Unlearning* method, which can be denoted as:

$$\Phi\left[\mathcal{FL}(D \setminus D_f)\right] = \Phi\left[\mathcal{FU}'(\mathcal{M}_r(D), \mathcal{M}_f(D))\right], \tag{5}$$

where $\mathcal{FU}'$: $\mathcal{M}_r \otimes \mathcal{M}_f \to \omega'$ represents the data-free unlearning process. $\mathcal{M}_r$ and $\mathcal{M}_f$ are the personalized models of the C_r and C_f, respectively, obtained after federated training on D. $\mathcal{FU}'$ does not require re-executing the federated training pipeline $\mathcal{FL}(D)$ and eliminates the access to raw data and historical updates. Instead, it operates solely on the parameters of the pre-trained models $\mathcal{M}_r$ and $\mathcal{M}_f$, which already encapsulate both global knowledge (shared across clients) and client-specific features (via personalized prompts). $\mathcal{FU}'$ enables post hoc removal of a client's influence with access to the parameters of the trained model ω, which are generally assumed to be readily available.

IV. Design of Mimir

In this section, we provide a comprehensive explanation of Mimir's pFL framework and its data-free unlearning process. pFL serves as the foundation for the unlearning, with unlearning being the primary contribution of our work.

A. Overall Architecture

1) Mimir Federated Learning Framework: The objective of Mimir's pFL is to train personalized models for each client in non-IID data distribution, with the extracted client-specific feature information serving as a prerequisite for subsequent unlearning.

Mimir's learning components include a backbone model such as the ResNet [33] and a personalized prompt generation module G_p. Mimir employs a pFL approach to extract client-specific features. Following the principles of FedRep [34] and GPFL [23], the backbone model is split into a feature extraction module (θ) and a model head (φ). We use φ to represent the last fully connected (FC) layer, while the θ comprises the remaining layers. As illustrated in Fig. 2, during training, ①: First, based on the learnable client-agnostic prompt (P_{base}) and client-specific descriptor (d_i), the prompt generator (G_p) produces personalized prompts (P_i) to adapt the global model to local data distributions. ②: Second, using P_i, the feature extractor (θ) transforms the extracted features $\boldsymbol{f_{\theta,i}}$ into $\boldsymbol{f_{P,i}}$, which is then utilized to train the local model. ③: Third, clients upload gradients of the P_{base}, G_p, and $\omega = \{\omega_\theta, \omega_\varphi\}$ to the server, excluding the client-specific descriptor d_i. ④: Then, the server aggregates all learnable components except d_i. ⑤: Finally, the updated parameters are broadcast to clients for the next training round, ensuring collaborative learning. The trainable parameters include ω, P_{base}, ω_{G_p} and d_i.

The client-specific descriptor d_i and personalized prompt P_i are pivotal components in Mimir's architecture, enabling adaptive feature transformation for non-IID data. The d_i acts as a query to retrieve relevant knowledge from P_{base}, while P_i adapts the feature space to mitigate non-IID divergence. Specifically, each client C_i is assigned a low-dimensional, learnable vector $d_i \in \mathbb{R}^{l_d}$ that encodes its local data distribution characteristics (e.g., label skewness or feature correlations). During the

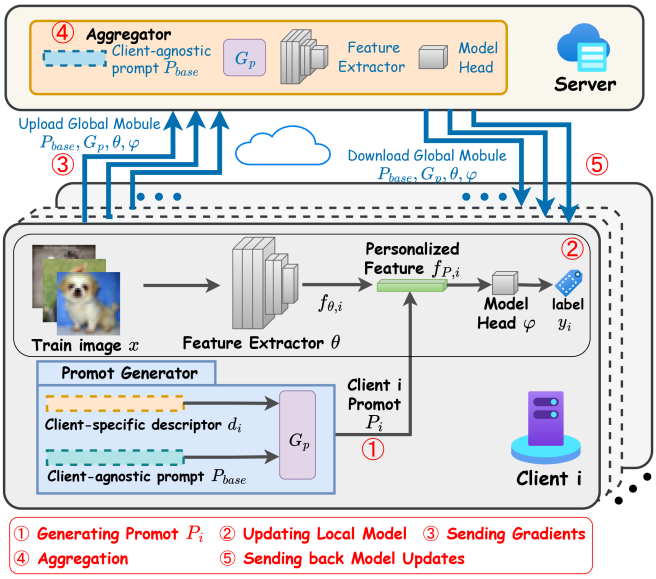


Fig. 2. Mimir personalized federated learning framework.

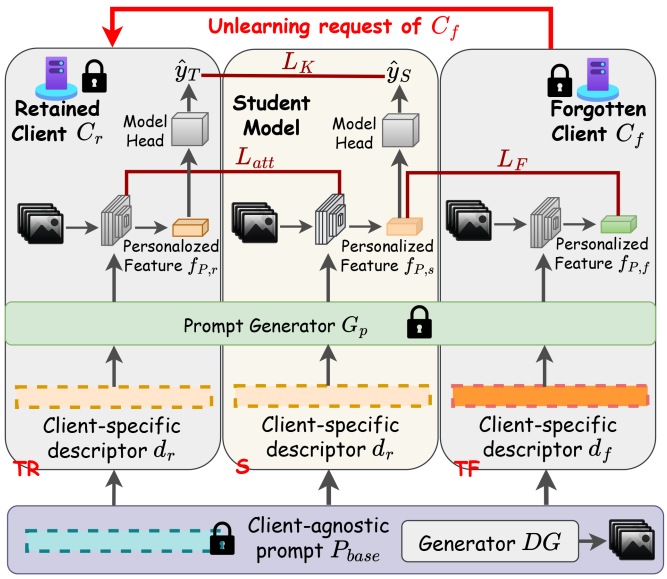


Fig. 3. Mimir data-free federated unlearning framework.

pFL phase, d_i is optimized locally via gradient descent, while remaining private to the client. For clients with imbalanced label distributions, the descriptor learns to emphasize visual features aligned with the label bias and subsequently converts them into prompts to guide the feature extractor. For the prompt (P_i), the global prompt generator G_p dynamically synthesizes a personalized prompt P_i for C_i through cross-attention between d_i and the client-agnostic base prompt P_{base}. This prompt acts as a lightweight transformation matrix, refining the backbone model's intermediate features $f_{\theta,i}$ via a Hadamard product. By aligning $f_{P,i}$ with C_i's data distribution, P_i enables personalized decision boundaries.

2) Mimir Data-Free Federated Unlearning Framework: The unlearning goal of Mimir is to effectively eliminate the influence of the forgotten client (C_f) on the retained clients (C_r) in a data-free scenario, where no data or historical updates from the clients are involved in the unlearning process. This is accomplished by forgetting the feature information generated by the personalized prompts on C_f, which means removing the model's ability to extract data from the forgotten client.

As illustrated in Fig. 3, the unlearning component comprises the personalized models of C_r and C_f, corresponding to the teacher model $TR(x; d_r, \omega_{TR})$ and the forgotten teacher model $TF(x; d_f, \omega_{TF})$ respectively, the student model $S(x; d_r, \omega_S)$, and the data generator DG. Mimir employs knowledge distillation (KD) and Generative Adversarial Networks (GANs) to transfer knowledge from C_r while obscuring the personalized feature information of C_f to achieve unlearning. The student model has the same structure as the model on C_r. Inspired by [13], [35], we use DG to create data points and minimize the loss $\mathcal{L}_F$ by adjusting the model decision boundaries on C_r to achieve unlearning. Intuitively, in the previous personalization learning process, the knowledge extracted by each client is personalized, mapping the information of different clients to distinct decision spaces, even for the same labeled data in different clients. Therefore, during the unlearning phase, we could achieve forgetting by having the personalized information of client C_f overshadowed by that of client C_r. Knowledge transfer is achieved by minimizing the divergence loss $\mathcal{L}_K$ and the distance loss $\mathcal{L}_{att}$ between the teacher and student models. By combining KD with GAN training, we achieve data-free unlearning. P_{base}, d_i and G_p remain unchanged because the goal of unlearning is to alter the backbone network to degrade the extraction capability of specific information, rather than modifying the trained client prompts. Trainable parameters for the student model during unlearning include $\omega_S = \{\omega_\theta, \omega_\varphi\}$.

B. Prompt-Based Federated Learning Process

The pFL phase involves collaboration between $\omega_\theta, \omega_\varphi, P_{base}$, ω_{G_p} and d_i. To fully utilize each client's personalized information, Mimir learns specific prompts P_i tailored to the client from G_p using client-specific descriptors d_i and client-agnostic prompt P_{base}, which are then used to generate the feature $\boldsymbol{f_{P,i}}$ for classification tasks. We detail these components on both client and server sides for pFL.

Feature Extractor: The feature extraction module θ extracts features $\boldsymbol{f_{\theta,i}}$ from the input $\boldsymbol{x_i}$ on client C_i. θ is the backbone network module excluding the model head φ and performs the mapping $\mathbb{R}^{dim(\boldsymbol{x_i})} \to \mathbb{R}^{dim(\boldsymbol{f_\theta})}$, where typically $dim(\boldsymbol{f_\theta}) \ll dim(x)$. For client i, we have:

$$\forall(\boldsymbol{x_i}, \boldsymbol{y_i}) \in D_i, \boldsymbol{f_{\theta,i}} = \theta(\boldsymbol{x_i}; \omega_\theta). \tag{6}$$

Client-specific Personalized Prompt Generation: Traditional FL methods, such as FedAvg [3], utilize the same model across all clients and simply use average aggregation. This approach can lead to the global model deviating from local data distributions, thus reducing learning effectiveness. For personalized federated learning, local adjustments to the global model or changes in the aggregation method are required. Mimir achieves global

model personalization by employing client-specific prompts. Specifically, we consider the locally trained prompt P_i as a client C_i-specific parameter, making the features extracted by θ more suitable for local clients. Mimir generates client-specific personalized prompts $\{P_1, P_2, \ldots, P_K\}$ for K clients. These personalized prompts are generated by a learnable parameterized conditional prompt generator $G_p(\cdot;\omega_{G_p})$ based on cross-attention. The inputs to $G_p(\cdot;\omega_{G_p})$ include a client-agnostic prompt P_{base} and a descriptor d_i for client C_i, which respectively capture client-agnostic information and encode client-specific feature information. Therefore, by querying the client descriptor d_i, client-relevant knowledge can be retrieved from P_{base}. Specifically, we generate personalized prompts P_i as denoted below:

$$\begin{aligned} P_i &= G_p\left(P_{base}, d_i; \omega_{G_p}\right) = P_{base} + [\mathsf{Atten}\left(\mathcal{Q}, \mathcal{K}, \mathcal{V}\right) W^{proj}]^T \\ &= P_{base} + [\mathsf{Softmax}\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{l_k}}\right)\mathcal{V}W^{proj}]^T, \\ &\text{where } \mathcal{Q} = d_i^T W^{\mathcal{Q}}, \mathcal{K} = P_{base}^T W^{\mathcal{K}}, \mathcal{V} = P_{base}^T W^{\mathcal{V}}. \end{aligned} \quad (7)$$

$P_i, P_{base} \in \mathbb{R}^{1\times l_\theta}$, $W^{\mathcal{Q}} \in \mathbb{R}^{1\times l_k}$, $W^{\mathcal{K}} \in \mathbb{R}^{1\times l_k}$, $W^{\mathcal{V}} \in \mathbb{R}^{1\times l_v}$, where l_θ is the embedding dimension of the descriptor d_i, which is the same as the feature dimension extracted by θ, i.e., $l_\theta = \dim(\boldsymbol{f_{\theta,i}})$, $\sqrt{l_k}$ is a scaling factor, $W^{proj} \in \mathbb{R}^{l_v\times 1}$ is a learnable projection matrix. l_k and l_v are the internal embedding dimensions of the prompt generation G_p. Once the personalized prompts P_i are generated, they are used to process the extracted feature information from the client-speciffc data. Specifically, the feature $\boldsymbol{f_{\theta,i}}$ extracted by θ is further refined to produce the personalized feature $\boldsymbol{f_{P,i}}$ for client C_i using P_i as denoted below:

$$\boldsymbol{f_{P,i}} = \frac{||\boldsymbol{f_{\theta,i}}||}{||P_i \circ \boldsymbol{f_{\theta,i}}||} \cdot (P_i \circ \boldsymbol{f_{\theta,i}}), \quad (8)$$

where $\circ$ denotes Hadamard product, P_i acts as a transformation matrix that tailors the extracted features $\boldsymbol{f_{\theta,i}}$ to better fit the specific characteristics of the client's data, thus enhancing the performance on local tasks. The scaling factor $\frac{||\boldsymbol{f_{\theta,i}}||}{||P_i \circ \boldsymbol{f_{\theta,i}}||}$ ensures that the $\boldsymbol{f_{P,i}}$ have the same norm as $\boldsymbol{f_{\theta,i}}$.

Model Head Output: The refined personal feature $\boldsymbol{f_{P,i}}$ is passed to the model head φ for classification or other downstream tasks. φ converts the feature vector $\boldsymbol{f_{P,i}}$ into the final prediction result $\hat{\boldsymbol{y}}_i$ as below:

$$\hat{\boldsymbol{y}}_i = \varphi(\boldsymbol{f_{P,i}}; \omega_\varphi). \quad (9)$$

Local Training and Model Aggregation: Mimir aims to find a backbone model and prompts that are of interest to the specific data distribution. To prevent overfitting on limited private data, we use prompt adjustments [36] on the basis of feature extraction. We share the prompt generator G_p, the client-agnostic prompt P_{base}, and the backbone model (θ and φ) among clients and perform FedAvg [3] aggregation on the server. Personalization is achieved by the adaptive adjustment of the client descriptor d_i on client C_i without aggregation, which in turn dynamically adjusts the client-specific prompt P_i. For the classification task on C_i, we calculate the cross-entropy loss $\mathcal{L}_{CE}$ on $|D_i|$ samples:

$$\mathcal{L}_{CE_i} = -\frac{1}{|D_i|}\sum\nolimits_{(\boldsymbol{x_i},\boldsymbol{y_i})\in D_i}\sum\nolimits_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}), \quad (10)$$

where $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for sample $\boldsymbol{x_i}$, and $\hat{y}_{i,c}$ is the predicted probability of class c for $\boldsymbol{x_i}$ calculated using (9). $\mathcal{L}_{CE}$ is used to optimize θ, φ and d_i via gradient descent:

$$\begin{cases} \omega_\theta \leftarrow \omega_\theta - lr \cdot \nabla_{\omega_\theta}\mathcal{L}_{CE_i}, \omega_\varphi \leftarrow \omega_\varphi - lr \cdot \nabla_{\omega_\varphi}\mathcal{L}_{CE_i}, \\ \omega_{d_i} \leftarrow \omega_{d_i} - lr \cdot \nabla_{\omega_{d_i}}\mathcal{L}_{CE_i}, \end{cases} \quad (11)$$

where lr is the learning rate. During local training for optimizing the backbone model and d_i, we keep G_p and P_{base} frozen to maintain the performance under the same prompt generation model across clients. After local training, the client C_i undergoes an additional round to update the G_p and P_{base}, with the loss function calculated as follows:

$$\mathcal{L}_{GP} = \mathcal{L}_{CE_i} + \lambda_1\|\omega_{G_p}\|_2^2 + \lambda_2\|P_{base}\|_2^2, \quad (12)$$

where λ_1 and λ_2 are L2 regularization parameters. The updates for G_p and P_{base} based on gradient descent are:

$$\omega_{G_p} \leftarrow \omega_{G_p} - lr \cdot \nabla_{\omega_{G_p}}\mathcal{L}_{GP}, P_{base} \leftarrow P_{base} - lr \cdot \nabla_{P_{base}}\mathcal{L}_{GP}. \quad (13)$$

We aggregate the updates to the global model using (2).

In summary, the learning phase of Mimir is outlined in Algorithm 1. It begins with initialization (line 1). In each communication round, a subset of clients is selected, and the server sends global parameters to these clients (lines 3-4). Each client trains locally by freezing the global prompt components P_{base} and G_p, and updating its local parameters (lines 6-13). After local training, clients compute an additional loss $\mathcal{L}_{GP}$ to update the global prompt generator G_p and the client-agnostic prompt P_{base} (lines 14-15). The server aggregates the global updates from all participating clients (lines 16-18). The process is repeated for multiple rounds until convergence is achieved.

C. Data-Free Federated Unlearning Process

The unlearning phase leverages Knowledge Distillation (KD) and Generative Adversarial Networks (GAN) in a data-free environment. In this phase, the forgotten client (C_f) requests the retained client (C_r) to delete its data, with neither client having access to training data or historical updates. Mimir use a data generator (DG) to produce pseudo-data for knowledge distillation. To eliminate the information extraction capability on the forgotten client, $\mathcal{L}_F$ is minimized to obscure the student model S's ability to extract private information on the forgotten client, similar to the behavior of TF, thereby achieving unlearning. To retain the information extraction capability on the retained client, Mimir minimizes the output KL divergence loss $\mathcal{L}_K$ and the intermediate layer similarity loss $\mathcal{L}_{att}$ between the model TR on C_r and the student model (S). Both $\mathcal{L}_K$ and $\mathcal{L}_F$ are maximized to update DG during GAN training.

Data Generator: We use a data generator $DG(z, \omega_{DG})$ to generate data points from a random noise vector $z \in \mathbb{R}^{dim(z)}$

Algorithm 1: Mimir's pFL Process.

Input: Client C_i training data D_i; Learning rate lr; Local training rounds T_l; Total communication rounds T; Hyperparameters λ_1, λ_2; Ratio of joining clients α.
Output: Updated global model parameters ω^T, client-specific descriptor d_i^T for K clients, client-agnostic prompt P_{base}^T, global prompt generator $\omega_{G_p}^T$.

Initialize global parameters shared among clients: $\omega^0 \leftarrow \{\omega_\theta^0, \omega_\varphi^0\}$, P_{base}^0, $\omega_{G_p}^0$, d_i^0 $(i = 1, 2, ..., K)$.
for *each communication round* t *from* 1 *to* T **do**
  Randomly select a fraction α of clients to form set $\mathcal{O}^t$.
  Server sends $\omega^{t-1}, P_{base}^{t-1}, \omega_{G_p}^{t-1}$ to client $\forall C_i \in \mathcal{O}^t$, set: $\omega_i^t \leftarrow \omega^{t-1}, P_{base}^t \leftarrow P_{base}^{t-1}, \omega_{G_p}^t \leftarrow \omega_{G_p}^{t-1}$.
  for *each selected client* $C_i \in \mathcal{O}^t$ ***in parallel*** **do**
    Freeze P_{base} and global prompt generator G_p.
    for *each local training round* $t_l \leftarrow 1$ **to** T_l **do**
      Extract features $\boldsymbol{f_{\theta,i}}$ using Eq. (6).
      Generate personalized prompt P_i using Eq. (7).
      Compute refined feature $\boldsymbol{f_{P,i}}$ using Eq. (8).
      Compute final prediction $\hat{y}_i$ from φ using Eq. (9).
      Compute loss function $\mathcal{L}_{CE_i}(\omega_i)$ using Eq. (10).
      Update $\omega_{\theta,i}^t, \omega_{\varphi,i}^t$ and d_i^t using Eq. (11).
    Compute additional loss $\mathcal{L}_{GP}$ using Eq. (12).
    Update $\omega_{G_p,i}^t$ and $P_{base,i}^t$ using Eq. (13) on $\mathcal{L}_{GP}$.
    Share $\omega_i^t = \{\omega_{\theta,i}^t, \omega_{\varphi,i}^t\}, \omega_{G_p,i}^t, P_{base,i}^t$ with server.
  Aggregate global backbone parameters ω on the server: $\omega^t \leftarrow \frac{1}{\sum_{i\in\mathcal{O}^t}|D_i|}\sum_{i\in\mathcal{O}^t}|D_i|\omega_i^t$.
  Aggregate updates for global prompt generator and client-agnostic prompt: $\omega_{G_p}^t \leftarrow \frac{1}{\sum_{i\in\mathcal{O}^t}|D_i|}\sum_{i\in\mathcal{O}^t}|D_i|\omega_{G_p,i}^t$, $P_{base}^t \leftarrow \frac{1}{\sum_{i\in\mathcal{O}^t}|D_i|}\sum_{i\in\mathcal{O}^t}|D_i|P_{base,i}^t$.
return Updated ω^T, d_i^T $(i = 1, 2, ..., K)$, P_{base}^T, $\omega_{G_p}^T$.

Algorithm 2: Mimir Data-Free Unlearning Process.

Input: Teacher model on C_r: $TR(x; d_r, \omega_{TR})$; Forgotten teacher model on C_f: $TF(x; d_f, \omega_{TF})$; Student model: $S(x; d_r, \omega_S)$; Temperature parameter τ; Learning rate lr; Knowledge Distillation rounds T_k; Unlearning training rounds T_u; Data Generator: $DG(z; \omega_{DG})$; Hyperparameters β and γ.
Output: Updated student model $S(x; d_r, \omega_S)$.

Initialize student model $S(x; d_r, \omega_S)$ with random weights ω_S.
for *each unlearning round* t *from* 1 *to* T_u **do**
  Sample a batch of random noise $z \sim \mathcal{N}(0, 1)$.
  Generate pseudo-samples using Eq. (14).
  for *each Knowledge Distillation round* k *from* 1 *to* T_k **do**
    Compute features $\boldsymbol{f_{\theta,s}}$ from $S(x; d_r, \omega_S)$.
    Compute features $\boldsymbol{f_{\theta,f}}$ from $TF(x; d_f, \omega_{TF})$.
    Compute feature similarity loss $\mathcal{L}_F$ using Eq. (15).
    Compute predictions $\hat{y}_T$ and $\hat{y}_S$ using Eq. (9).
    Compute KL divergence loss $\mathcal{L}_K$ using Eq. (16).
    Compute L2 attention loss $\mathcal{L}_{att}$ using Eq. (17).
    Compute total loss $\mathcal{L}_S = \mathcal{L}_F + \beta\mathcal{L}_K + \gamma\mathcal{L}_{att}$.
    Update student model parameters using Eq. (18).
  Update $DG(z; \omega_{DG})$ parameters using Eq. (19).
return Updated model $S(x; d_r, \omega_S)$ as unlearned model.

sampled from $\mathcal{N}(0, 1)$. The generator first maps z to a higher-dimensional space using a series of layers, including linear layers, reshaping, batch normalization, and upsampling. The feature mappings are then processed by convolution, batch normalization, and Leaky ReLU activation to generate pseudo-samples for KD training as below:

$$\boldsymbol{x} = DG(z; \omega_{DG}), \tag{14}$$

which are used as inputs for the teacher model TR, the student model S, and the forgotten teacher model TF.

Data-Free Knowledge Distillation: Inspired by [35], [37], our goal is to achieve approximate unlearning by mapping the decision space of personalized information from client C_f to client C_r. Specifically, we aim to make the feature information $\boldsymbol{f_{\theta,f}}$ extracted from client C_f similar to the student model's $\boldsymbol{f_{\theta,s}}$. Intuitively, when the difference between the feature information $\boldsymbol{f_{\theta,f}}$ and $\boldsymbol{f_{\theta,s}}$ is large, it indicates a higher likelihood of exposing the privacy of C_f, reflecting its unique contribution. To achieve unlearning, we need to alter its decision space. When the difference between them is small, it suggests that the private information of both is similar, and the effort required for unlearning is reduced. By aligning $\boldsymbol{f_{\theta,f}}$ and $\boldsymbol{f_{\theta,s}}$, we can naturally achieve the desired unlearning. We set the client descriptors to d_r and d_f for prompt generation using (6), (7), and (8), obtaining $\boldsymbol{f_{\theta,s}}$ and $\boldsymbol{f_{\theta,f}}$ from $S(x; d_r, \omega_f)$ and $T(x; d_f, \omega_f)$, respectively. As a peer-to-peer unlearning framework, Mimir use d_r on the student model because the unlearned model continues to work on C_r. We use cosine similarity to measure the similarity between $\boldsymbol{f_{\theta,s}}$ and $\boldsymbol{f_{\theta,f}}$. The loss function $\mathcal{L}_F$ is:

$$\mathcal{L}_F = 1 - \frac{\boldsymbol{f_{\theta,s}} \cdot \boldsymbol{f_{\theta,f}}^T}{\|\boldsymbol{f_{\theta,f}}\| \cdot \|\boldsymbol{f_{\theta,s}}\|}, \tag{15}$$

where a smaller $\mathcal{L}_F$ loss indicates higher similarity, effectively "masking" the personalized information of C_f without affecting the decision space of C_r. When the data distributions of C_r and C_f are highly similar, $\mathcal{L}_F$ itself is small, preventing catastrophic forgetting during the unlearning process. Let $\hat{y}_T$ and $\hat{y}_S$ be the predictions of the teacher $T(x; d_f, \omega_f)$ and student models $S(x; d_r, \omega_f)$ on x using (9), respectively. To facilitate knowledge transfer between the output of TR and S, $\mathcal{L}_K$ measures the divergence between the outputs using KL divergence:

$$\mathcal{L}_K = \tau^2 \sum_{c=0}^{C} \sigma(\hat{y}_T/\tau)_i \log\left(\frac{\sigma(\hat{y}_T/\tau)_i}{\sigma(\hat{y}_S/\tau)_i}\right), \tag{16}$$

where τ is the temperature parameter and σ denotes the softmax function. Inspired by [35], we also combine attention loss with L2 normalization:

$$\mathcal{L}_{att} = \frac{1}{|\mathcal{N}_L|}\sum_{l\in\mathcal{N}_L}\left\|\frac{f_A(H_l^{(T)})}{\|f_A(H_l^{(T)})\|_2} - \frac{f_A(A_l^{(S)})}{\|f_A(A_l^{(S)})\|_2}\right\|_2, \tag{17}$$

where $H_l^{(T)}$ and $H_l^{(S)}$ denote the outputs of the l-th layer of the teacher and student models, respectively. $\mathcal{N}_L$ represents the subset of layers used to compute the attention loss, particularly the activation layers of θ. The output H_l contains n_l channels. The function $f_A(H_l) = \frac{1}{n_l}\sum_c a_{l,c}^2$, where $a_{l,c}$ represents the c-th channel of the activation block A_l.

GAN Training Process: During the knowledge distillation process, the student $S(x; d_r, \omega_f)$ updates its weights to forget specific personalized information by minimizing $\mathcal{L}_F$ and maintains the knowledge of C_r by minimizing $\mathcal{L}_K$ and $\mathcal{L}_{att}$ based on gradient descent as follows:

$$\omega_{\theta_s} \leftarrow \omega_{\theta_s} - lr \cdot \nabla_{\omega_{\theta_s}} \mathcal{L}_S, \omega_{\varphi_s} \leftarrow \omega_{\varphi_s} - lr \cdot \nabla_{\omega_{\varphi_s}} \mathcal{L}_S, \quad \text{where } \mathcal{L}_S = \mathcal{L}_F + \beta \mathcal{L}_K + \gamma \mathcal{L}_{att}. \tag{18}$$

Here, β and γ are hyperparameters. The data generator DG is optimized as an adversarial sample generator with an objective opposing that of the student model to improve data generation quality. The objective of DG is:

$$\arg\max_{\omega_{DG}} \mathcal{L}_{DG}, \text{where} \mathcal{L}_{DG} = \mathcal{L}_F + \beta \mathcal{L}_K. \tag{19}$$

The DG and S are alternately updated. The adversarial objective of the generator is defined as $\mathcal{L}_{DG} = \mathcal{L}_F + \beta \mathcal{L}_K$, where $\mathcal{L}_F$ drives the student model to disrupt the decision boundary of the forgotten client, thereby suppressing its ability to extract client-specific information from C_f; in contrast, $\mathcal{L}_K$ ensures the retention of feature extraction capabilities pertaining to the preserved client's knowledge. Finally, the student model effectively forgets the personalized information of the forgotten client while maintaining the general knowledge of the retained client.

We deliberately exclude $\mathcal{L}_{att}$ from the DG objective for the following reasons. First, $\mathcal{L}_{att}$ aligns intermediate features via layer-wise L2 norm similarity and primarily serves to regularize the internal feature learning of the student model. Incorporating it into the DG loss would dilute the adversarial objective—namely, the maximization of $\mathcal{L}_F$ and $\mathcal{L}_K$. This is because these two losses directly address the core forgetting signal by suppressing the features of the forgotten client while preserving global knowledge; hence, the functionality of $\mathcal{L}_{att}$ overlaps with that of $\mathcal{L}_K$. Second, computing $\mathcal{L}_{att}$ across multiple intermediate layers introduces additional computational overhead and may destabilize the adversarial equilibrium between DG and S. Omitting $\mathcal{L}_{att}$ thus contributes to ensuring both efficiency and stability.

In summary, the unlearning phase of Mimir is outlined in Algorithm 2. It starts by initializing the student model (line 1). In each unlearning round, random noise is sampled, and pseudo-samples are generated (lines 3-4). During each KD round, personalized features are computed, and losses such as $\mathcal{L}_F$, $\mathcal{L}_K$, and $\mathcal{L}_{att}$ are calculated (lines 6-11). The student model is then updated based on these losses (lines 12-13). After the distillation rounds, the data generator's parameters are updated (line 14). This process is repeated for several unlearning rounds until convergence.

V. Theoretical Analysis

A. Convergence Analysis of pFL

We prove the convergence of Mimir under Local Stochastic Gradient Descent (LSGD) in the context of heterogeneous personalized FL (pFL). Notably, we relax the convexity assumption on the loss functions to non-convex. Building on the theory of LSGD-PFL and the unique design of Mimir, we introduce the following assumptions:

Assumption 1 (L-Smoothness): The local loss functions in the Mimir framework are L-smooth, which implies:

$$\|\nabla \mathcal{L}_{Mi}(\omega) - \nabla \mathcal{L}_{Mi}(\omega')\| \leq L\|\omega - \omega'\|, \forall \omega, \omega' \in \mathbb{R}^{l_0+l_i},\ i \in [K], \tag{20}$$

Here, $i \in [K]$ denotes the client index within the set $\{1, 2, \ldots, K\}$, $\mathcal{L}_{Mi}$ is the local objective function for client i, l_0 represents the dimension of global parameters (shared across clients), and l_i represents the dimension of client-specific local parameters. The global parameters include the client-agnostic prompt P_{base}, the feature extractor θ, the model head φ, and the prompt generator G_p, all aggregated by the server during FL. The client-specific local parameters are tailored to the data distribution of each client and updated locally, including the client-specific descriptor d_i.

Assumption 2 (Bounded Gradient Variance): For all $i \in [K]$, the local stochastic gradients computed with a random variable ζ, denoted as $\nabla_{\omega_{\text{global}}} \hat{\mathcal{L}}_{Mi}(\omega_{\text{global}}, \omega_i, \zeta)$ and $\nabla_{\omega_i} \hat{\mathcal{L}}_{Mi}(\omega_{\text{global}}, \omega_i, \zeta)$, satisfy the following variance bounds for all $i \in [K]$:

$$\begin{cases} \mathbb{E}_\zeta \|\nabla_{\omega_{\text{global}}} \hat{\mathcal{L}}_{Mi}(\omega_{\text{global}}, \omega_i, \zeta) - \nabla_{\omega_{\text{global}}} \mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i)\|^2 \\ \quad \leq C_1 \|\nabla_{\omega_{\text{global}}} \mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i)\|^2 + \frac{\sigma_1^2}{B}, \\ \mathbb{E}_\zeta \|\nabla_{\omega_i} \hat{\mathcal{L}}_{Mi}(\omega_{\text{global}}, \omega_i, \zeta) - \nabla_{\omega_i} \mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i)\|^2 \\ \quad \leq C_2 \|\nabla_{\omega_i} \mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i)\|^2 + \frac{\sigma_2^2}{B}, \end{cases} \tag{21}$$

where $\omega_{\text{global}} \in \mathbb{R}^{l_0}$ and $\omega_i \in \mathbb{R}^{l_i}$ are global and local parameters, respectively. Constants $C_1, \sigma_1^2, \sigma_2^2$ are positive, and B is the batch size. Gradient boundedness assumptions are widely employed in pFL frameworks (e.g., MAML-based methods [38]). Under a shared objective, Mimir introduces client heterogeneity terms that are specifically tailored to its prompt-driven architecture. This design accounts for the asymmetric updates between global and local parameters induced by non-IID personalized prompts, which is different from conventional MAML-style optimization. In Mimir, the L2-regularized loss $\mathcal{L}_{GP}$ (12) constrains the updates of G_p and P_{base}, ensuring that the global prompt components remain stable during the aggregation process. This, in turn, limits the gradient variance caused by client-specific adjustments of d_i, as reflected by σ_1^2 and σ_2^2.

Assumption 3 (Bounded Heterogeneity): To mathematically characterize the heterogeneity of client data distributions, we assume that there exists a positive constant $\lambda > 0$ such that for all $\omega_{\text{global}} \in \mathbb{R}^{l_0}$ and $\omega_i \in \mathbb{R}^{l_i}, i \in [K]$:

$$\frac{1}{K} \sum_{i=1}^{K} \|\nabla \mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i)\|^2 \leq \lambda \left\| \frac{1}{K} \sum_{i=1}^{K} \nabla \mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i) \right\|^2 + \sigma_{\text{dif}}^2, \tag{22}$$

which reflects the heterogeneity of client data distributions, as used in the Definition 1 in [39]. In Mimir, client-specific prompts $P_i = G_p(P_{base}, d_i)$ (7) induce heterogeneous feature

transformations, a process explicitly bounded by the term σ_{dif}^2, which quantifies the divergence caused by non-IID data and directly reflects the effect of Mimir's prompt-driven personalization. It is important to note that these assumptions do not impose unrealistic constraints on practical FL scenarios but rather provide a theoretical framework to analyze and guarantee convergence. L-smoothness ensures that the local optimization landscape does not exhibit abrupt variations, which aligns with common assumptions in convex and non-convex optimization. Bounded Gradient Variance accounts for the inherent randomness in stochastic gradients due to batch sampling, preventing excessive variance that could destabilize training. Bounded Heterogeneity captures the variability among client models, which is a fundamental challenge in FL due to non-IID data distributions. Under the above assumptions, we can establish the convergence rate of Mimir:

Theorem 1 (Convergence of Mimir under Non-convex Objectives): Under the constraints of L-Smoothness, Bounded Gradient Variance, and Bounded Heterogeneity, when the learning rates of client $i \in [K]$ satisfy certain conditions and t is sufficiently large, there exists an upper bound such that:

$$\mathbb{E}\left[\left\|\frac{1}{K}\sum_{i=1}^{K}\nabla\mathcal{L}_{Mi}(\omega_{\text{global}}^t,\omega_i^t)\right\|^2\right] \le \frac{A}{t}, \tag{23}$$

where A is a constant corresponding to the assumption conditions and training hyperparameters. As t becomes large, the average gradient of each client gradually approaches zero, leading to the convergence of the global model. It refers to the stabilization of model parameter updates, reaching a steady state within a certain precision range. It is noteworthy that Mimir's convergence guarantees stem directly from its unique architecture. Unlike conventional FL, Mimir's dual-parameter structure introduces controlled asymmetry in gradient updates by separating global parameters ($\omega_{global} = \{\omega_\theta, \omega_\varphi, P_{base}, G_p\}$) from client-specific descriptors d_i. The core proof is tightly intertwined with Mimir's design. Furthermore, unlike multi-task personalized FL methods that optimize task-specific parameters [40], [41] or adaptive aggregation approaches [42], [43], Mimir's explicit separation of global and personalized parameters enables Theorem 1 to model their joint optimization while effectively constraining asymmetric updates. Its theoretical guarantees are inseparable from its unique design.

Proof: Under the above assumptions (20), (21), and (22), with a learning rate $lr_k = lr$ satisfying:

$$-1 + lrL\lambda\left(\frac{C_1}{K} + C_2 + 1\right) + lr^2L^2\lambda(\tau-1)\tau(C_1+1) \le 0. \tag{24}$$

After T training rounds, the following holds:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\|\frac{1}{K}\sum_{i=1}^{K}\nabla\mathcal{L}_{Mi}(\omega_{global}^t,\omega_i^t)\right\|^2\right] &\le \frac{lr^2L^2\sigma_1^2(\tau-1)^2}{B} \\ &+ lrL\lambda\left\{\left(\frac{C_1}{K}+C_2+1\right)\sigma_{\text{dif}}^2 + \frac{\sigma_1^2}{KB} + \frac{\sigma_2^2}{B}\right\} \\ &+ lr^2L^2\sigma_{\text{dif}}^2(\tau-1)^2 \\ &\cdot (C_1+1) + \frac{2\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\mathcal{L}_{Mi}(\omega_{global}^0,\omega_i^0) - \mathcal{L}^*\right]}{lrT}, \end{aligned} \tag{25}$$

where τ is the communication period, and $\omega^t = \frac{1}{K}\sum_{i=1}^{K}\omega_i^t$ is a sequence of so-called virtual iterates. By simplifying the (25), we obtain:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\|\frac{1}{K}\sum_{i=1}^{K}\nabla\mathcal{L}_{Mi}(\omega_{\text{global}}^t,\omega_i^t)\right\|^2\right] \\ \le \frac{2\Delta_f}{lrT} + lr\Psi + lr^2\Theta, \end{aligned} \tag{26}$$

where Δ_f is the initial error, and Ψ and Θ depend on gradient noise and heterogeneity-related constants. From (26), it is evident that Theorem 1 holds. As $t \to \infty$, the first term on the right-hand side of (26) tends to zero, while the second and third terms approach constants. Therefore, we have: $\lim_{t\to\infty}\mathbb{E}[\|\frac{1}{K}\sum_{i=1}^{K}\nabla\mathcal{L}_{Mi}(\omega_{\text{global}}^t,\omega_i^t)\|^2] = 0$, which indicates the convergence under non-convex settings. □

Theorem 2 (Convergence Rate under PL Condition). Polyak-Łojasiewicz (PL) condition is commonly used to characterize non-convex objectives in the literature. In addition to the Assumptions 1–3, if the local objective functions satisfy the PL condition, and the learning rates and communication periods of each client meet certain conditions, then the objective function value $\mathcal{L}_{Mi}(\omega_{\text{global}}, \omega_i)$ of the optimization algorithm converges at a rate of $\mathcal{O}(1/T)$ after T iterations.

Proof: If the local objective functions satisfy the PL condition, there exists a constant $\mu > 0$ such that for all $\omega_{\text{global}} \in \mathbb{R}^{l_0}$ and $\omega_i \in \mathbb{R}^{l_i}, i \in [K]$:

$$\begin{aligned}\frac{1}{2}\left\|\frac{1}{K}\sum_{i=1}^{K}\nabla\mathcal{L}_{Mi}(\omega_{\text{global}},\omega_i)\right\|^2 \\ \ge \mu\left(\frac{1}{K}\sum_{i=1}^{K}\mathcal{L}_{Mi}(\omega_{\text{global}},\omega_i) - \mathcal{L}^*\right), \end{aligned} \tag{27}$$

then, by choosing $lr_k = 1/(\mu(t+\beta\tau+1))$ and β satisfying:

$$\beta > \max\left\{\frac{2\lambda L}{\mu}\left(\frac{C_1}{M}+C_2+1\right) - 2, \frac{2L^2\lambda(C_1+1)}{\mu^2}, 1\right\}, \tag{28}$$

and τ is large enough satisfying:

$$\tau \ge \sqrt{\frac{\max\left\{(2L^2\lambda(C_1+1)/\mu^2)e^{1/\beta} - 4, 0\right\}}{\beta^2 - (2L^2\lambda(C_1+1)/\mu^2)e^{1/\beta}}} \tag{29}$$

Then Mimir can achieve a convergence rate of:

$$\begin{aligned}\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\mathcal{L}_{Mi}(\omega_{global},\omega_i) - \mathcal{L}^*\right] &\le \frac{LT(T+2\beta\tau+2)}{4\mu^2(T+\beta\tau)^3} \\ &\left\{\sigma_{\text{dif}}^2\left(\frac{C_1}{K}+C_2+1\right) + \frac{\sigma_1^2}{KB} + \frac{\sigma_2^2}{B}\right\} + \frac{2L^2(\tau-1)^2T}{\mu^3(T+\beta\tau)^3} \\ &\left\{\sigma_{\text{dif}}^2(C_1+1) + \frac{\sigma_1^2}{B}\right\} \\ &+ \frac{b^3\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\mathcal{L}_{Mi}(\omega_{global}^0,\omega_i^0) - \mathcal{L}^*\right]}{(T+\beta\tau)^3}. \end{aligned} \tag{30}$$

By simplifying the (30), we have

$$\mathbb{E}\left[\frac{1}{K}\sum_{i=1}^{K}\mathcal{L}_{Mi}(\omega_{\text{global}},\omega_i)-\mathcal{L}^*\right]\leq\mathcal{O}\left(\frac{1}{T}\right). \tag{31}$$

From (31), it is evident that Theorem 2 holds. □

The omitted intermediate steps due to page limitations can be found in Appendix B.7 of [44]. Based on the above, by combining the personalized prompt generation and local optimization objectives in the Mimir framework, we decompose the training error of the client models into a joint smooth error of global parameters and personalized parameters. The above proof shows that the Mimir can converge in non-convex pFL setting, while achieving good convergence speed under PL conditions.

B. Convergence Analysis of Unlearning

Traditional FU methods [45] rely on the convergence of the federated process. In contrast, Mimir's convergence analysis primarily focuses on the GAN process due to its peer-to-peer unlearning setting, we conduct a rigorous analysis based on fixed-point theory and dynamical systems. We formalize the unlearning process as a minimax optimization problem and extends the theoretical foundation of LGDA [46] to the FU scenario with data-free constraints, with the objective function defined as:

$$\min_{\omega_S}\max_{\omega_{DG}}\mathcal{L}(\omega_S,\omega_{DG})=\mathbb{E}_{z\sim\mathcal{N}(0,1)}\left[\mathcal{L}_F+\beta\mathcal{L}_K+\gamma\mathcal{L}_{att}\right], \tag{32}$$

where Primal variable ω_S is the parameter of the student model, which is optimized to minimize the objective function and thereby forget specific client information. Dual variable ω_{DG} is the parameter of the generator, which is optimized to maximize the objective function by generating adversarial synthetic data. This optimization process follows a training strategy similar to that of GANs: the generator attempts to fool the student model with misleading data, while the student model learns to resist such interference, ultimately achieving the goal of targeted forgetting. The assumptions are as below:

Assumption 4 (Unlearning Loss Smoothness): The loss functions $\mathcal{L}_F$, $\mathcal{L}_K$, and $\mathcal{L}_{att}$ are L-smooth with respect to both the student model parameters ω_S and generator parameters ω_{DG}. Formally, for any ω_S,ω_S', and ω_{DG},ω_{DG}':

$$\begin{aligned}&\|\nabla_{\omega_S}\mathcal{L}\left(\omega_S,\omega_{DG}\right)-\nabla_{\omega_S}\mathcal{L}\left(\omega_S',\omega_{DG}'\right)\|\\&\quad\leq L\left(\|\omega_S-\omega_S'\|+\|\omega_{DG}-\omega_{DG}'\|\right),\end{aligned} \tag{33}$$

and similarly for $\nabla_{\omega_{DG}}\mathcal{L}(\cdot)$.

Assumption 5 (Bounded Gradient Variance): The stochastic gradients computed on pseudo-data generated by DG have bounded variance. For all ω_S,ω_{DG}:

$$\begin{aligned}&\mathbb{E}_z\left[\|\nabla_{\omega_S}\mathcal{L}(\cdot;z)-\nabla_{\omega_S}\mathcal{L}\|^2\right]\leq\sigma^2,\\&\mathbb{E}_z\left[\|\nabla_{\omega_{DG}}\mathcal{L}(\cdot;z)-\nabla_{\omega_{DG}}\mathcal{L}\|^2\right]\leq\sigma^2.\end{aligned} \tag{34}$$

Assumption 6 (Nonconvex-Strongly-Concave Structure): For the student S (Primal), the objective $\mathcal{L}_S=\mathcal{L}_F+\beta\mathcal{L}_K+\gamma\mathcal{L}_{att}$ is nonconvex in ω_S. For the DG (Dual), the objective $\mathcal{L}_{DG}=\mathcal{L}_F+\beta\mathcal{L}_K$ is μ-strongly concave in ω_{DG}, i.e.,

$$\begin{aligned}\mathcal{L}_{DG}(\omega_S,\omega_{DG})\leq\mathcal{L}_{DG}(\omega_S,\omega_{DG}^*)+\langle\nabla_{\omega_{DG}}\mathcal{L}_{DG}(\omega_S,\omega_{DG}^*),\\\omega_{DG}-\omega_{DG}^*\rangle-\frac{\mu}{2}\|\omega_{DG}-\omega_{DG}^*\|^2.\end{aligned} \tag{35}$$

Theorem 3 (Convergence Under Nonconvex-Strongly-Concave Case): Under Assumptions 4–6, let lr_S and lr_{DG} be the learning rates for the student model and generator, respectively. If we set $lr_S=\frac{1}{L\sqrt{T}}$, $lr_{DG}=\frac{1}{\mu T^{1/3}}$, and synchronization gap $\tau=T^{1/3}$, then after T total iterations, the student model ω_S converges to a first-order stationary point:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla\Phi\left(\omega_S^{(t)}\right)\|^2\right]\leq O\left(\frac{L\sigma^2}{\sqrt{T}}+\frac{\mu\sigma^2}{T^{1/3}}\right), \tag{36}$$

where $\Phi(\omega_S)=\max_{\omega_{DG}}\mathcal{L}(\omega_S,\omega_{DG})$ is the primal function.

Proof: First, for the primal-dual dynamics, the updates for the student model (primal) and generator (dual) follow:

$$\begin{cases}\omega_S^{(t+1)}=\omega_S^{(t)}-lr_S\nabla_{\omega_S}\mathcal{L}_S\left(\omega_S^{(t)},\omega_{DG}^{(t)}\right),\\\omega_{DG}^{(t+1)}=\omega_{DG}^{(t)}+lr_{DG}\nabla_{\omega_{DG}}\mathcal{L}_{DG}\left(\omega_S^{(t)},\omega_{DG}^{(t)}\right).\end{cases} \tag{37}$$

Then we define the primal function $\Phi(\omega_S)=\max_{\omega_{DG}}\mathcal{L}(\omega_S,\omega_{DG})$. Using the smoothness of Φ:

$$\begin{aligned}\Phi(\omega_S^{(t+1)})\leq\Phi\left(\omega_S^{(t)}\right)-lr_S\langle\nabla\Phi\left(\omega_S^{(t)}\right),\nabla_{\omega_S}\mathcal{L}_S\rangle\\+\frac{Llr_S^2}{2}\|\nabla_{\omega_S}\mathcal{L}_S\|^2.\end{aligned} \tag{38}$$

Due to the stochastic gradient $\nabla_{\omega_S}\mathcal{L}_S$ can be decomposed into true gradient and noise, we have:

$$\nabla_{\omega_S}\mathcal{L}_S=\nabla\Phi\left(\omega_S^{(t)}\right)+\underbrace{\nabla_{\omega_S}\mathcal{L}_S-\nabla\Phi\left(\omega_S^{(t)}\right)}_{\text{Noise }\xi_t}. \tag{39}$$

From Assumption 5, $\mathbb{E}[\|\xi_t\|^2]\leq\sigma^2$. Taking expectation over the randomness:

$$\begin{aligned}\mathbb{E}\left[\Phi\left(\omega_S^{(t+1)}\right)\right]\leq\mathbb{E}\left[\Phi\left(\omega_S^{(t)}\right)\right]-lr_S\mathbb{E}\left[\|\nabla\Phi\left(\omega_S^{(t)}\right)\|^2\right]\\+\frac{Llr_S^2}{2}\left(\mathbb{E}[\|\nabla\Phi\left(\omega_S^{(t)}\right)\|^2]+\sigma^2\right).\end{aligned} \tag{40}$$

Simplifying:

$$\begin{aligned}\mathbb{E}\left[\Phi\left(\omega_S^{(t+1)}\right)\right]\leq\mathbb{E}\left[\Phi\left(\omega_S^{(t)}\right)\right]-lr_S\left(1-\frac{Llr_S}{2}\right)\\\times\mathbb{E}\left[\|\nabla\Phi\left(\omega_S^{(t)}\right)\|^2\right]+\frac{Llr_S^2\sigma^2}{2}.\end{aligned} \tag{41}$$

Using Lemma from [46], the deviation between local and global models is bounded as:

$$\mathbb{E}\left[\|\omega_S(t)-\omega_S^*\|^2\right]\leq\frac{4lr_S^2\sigma^2}{\mu^2}+\frac{8lr_{DG}^2\tau^2\sigma^2}{\mu} \tag{42}$$

Substituting $\tau=T^{1/3}$ and $lr_{DG}=O(\frac{lr_S}{\tau})$, we get:

$$\mathbb{E}\left[\|\omega_S(t)-\omega_S^*\|^2\right]\leq O\left(\frac{\sigma^2}{T^{2/3}}\right) \tag{43}$$

Summing over $t = 1, \ldots, T$ and dividing by T, we have final convergence rate:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla\Phi(\omega_S(t))\|^2\right] \leq O\left(\frac{L\sigma^2}{\sqrt{T}} + \frac{\mu\sigma^2}{T^{1/3}}\right) \tag{44}$$

□

By aligning the unlearning framework with the assumptions and techniques of Local SGDA, we rigorously prove its convergence under nonconvex-strongly-concave settings.

C. pFL Complexity Analysis

The total complexity of Mimir in the pFL process includes four main parts: feature extraction, personalized prompt generation, local updates, and global aggregation. The first is feature extraction. For each client C_i has complexity $\mathcal{O}(|D_i| \cdot l_x \cdot l_\theta)$, where $|D_i|$ is the number of local samples, l_x is the input dimension, and l_θ is the feature dimension. For prompt generation, it involves attention operations and feature refinement, with complexity $\mathcal{O}(|D_i| \cdot (l_\theta^2(l_k + l_v) + l_v^2 \cdot l_\theta))$. Local training (classification and backpropagation over T_l iterations) adds $\mathcal{O}(T_l \cdot |D_i| \cdot (l_x \cdot l_\theta + l_\theta^2(l_k + l_v) + l_v^2 \cdot l_\theta + l_\theta \cdot C))$, where C is the number of classes. Global aggregation via FedAvg over K clients has complexity $\mathcal{O}(K \cdot |\omega_{\text{avg}}|)$, where $|\omega_{\text{avg}}|$ is the total size of transmitted model parameters. Thus, for T rounds of communication and K clients, the overall complexity is: $\mathcal{O}(\text{pFL}) = T \cdot (\sum_{i=1}^{K} \mathcal{O}(|D_i| \cdot \text{model terms}) + \mathcal{O}(K \cdot |\omega_{\text{avg}}|))$ Mimir scales linearly with the number of clients and local dataset sizes. Although prompt generation contributes nontrivially to computation, the backbone model training remains the main cost. Communication is dominated by $|\omega_{\text{avg}}|$, which is kept manageable to ensure scalability.

D. Unlearning Complexity Analysis

We analyze the algorithmic complexity of the unlearning stage, which integrates KD and GAN. During unlearning, client C_f requests C_r to remove its data, without access to training data or historical updates. In each unlearning iteration, the generator DG generates pseudo-samples from noise, with a complexity of $\mathcal{O}(B \cdot \omega_{DG})$, where ω_{DG} is the number of generator parameters and B is the number of pseudo-samples. During KD, the student and forgetting teacher models compute features $f_{\theta,s}$, $f_{\theta,f}$, and classification outputs $\hat{y}_S$, $\hat{y}_T$, with a single forward pass complexity of $\mathcal{O}(l_x \cdot l_\theta + l_\theta^2 \cdot (l_k + l_v) + l_v^2 \cdot l_\theta + l_\theta \cdot C)$. Loss computation includes feature similarity ($\mathcal{L}_F$, $\mathcal{O}(l_\theta)$), KL divergence ($\mathcal{L}_K, \mathcal{O}(C)$), and attention loss ($\mathcal{L}_{att}, \mathcal{O}(n_{\text{layer}}(\mathcal{N}_L) \cdot n_l)$), where n_l is the number of channels per layer, and $n_{\text{layer}}(\mathcal{N}_L)$ is the number of selected layers. Over T_u unlearning iterations, each with T_k KD rounds, the total complexity is approximately: $\mathcal{O}(T_u \cdot B \cdot (T_k \cdot (\mathcal{O}(l_x \cdot l_\theta + l_\theta^2 \cdot (l_k + l_v) + l_v^2 \cdot l_\theta + d_\theta \cdot C) + \mathcal{O}(n_{\text{layer}}(\mathcal{N}_L) \cdot n_l)) + \mathcal{O}(\omega_{DG})))$. The scalability of the proposed process is linear with respect to the number of unlearning rounds and KD rounds. The primary computational bottleneck lies in the size of the student model and the parameters of the generator, which is manageable.

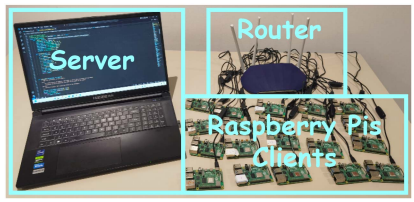


Fig. 4. Real-world federated experiment environment.

VI. Evaluations

A. Evaluation Setups

1) Testbed: The experiments were conducted in a real FL environment, including multiple clients equipped with ARM11 microprocessor Raspberry Pis and a laptop server, as shown in Fig. 4. The system was implemented using PyTorch 1.6.0.

2) Datasets and Backbone Models: We utilized four benchmark datasets for machine unlearning as shown in Table II: MNIST[2], SVHN[3], FASHION-MNIST[4], and CIFAR10[5]. Unless otherwise specified, the backbone model adopts a convolutional neural network (CNN) comprising two cascaded convolutional modules followed by a fully connected layer. Specifically, the first convolutional module includes a 2D convolutional layer with 32 output channels, a 5×5 kernel, stride 1, no padding, followed by ReLU activation and 2×2 max-pooling. The second module follows the same structure but with 64 output channels. The extracted features are flattened and passed through a fully connected layer with 512 neurons and ReLU activation, followed by a classification layer mapping to the target class dimension.

3) Baselines: Given the scarcity of work on data-free unlearning in FL environments, we extended data-free methods from the machine unlearning scenario to the federated client unlearning for comparison. To ensure a fair comparison in a non-IID setting, we applied Mimir's prompt learning configuration to the following methods: i) *FedMM*: This method utilizes *Error Minimization-Maximization Noise* [13], [29] during client unlearning. It learns noise matrices for unlearned model training that minimize the cross-entropy loss on retained client while maximizing it on the forgotten client. We retrain the original model using the generated noise matrix as data samples. ii) *FedGKT*: We extended the *Gated Knowledge Transfer* [13] method to the FL scenario using Mimir's learning configuration. For unlearning, the originally trained model acts as the teacher, and a network with the same structure serves as the student. A band-pass filter calculates the similarity between the personalized information $\boldsymbol{f_{P,r}}$ and $\boldsymbol{f_{P,f}}$, extracted from C_r and C_f by the prompt. Only samples with low similarity are allowed to pass through, filtering out and preventing the information of C_f from being transmitted to the student model.

4) Evaluation Metrics: The evaluation use training and testing datasets, though they aren't involved during unlearning. The

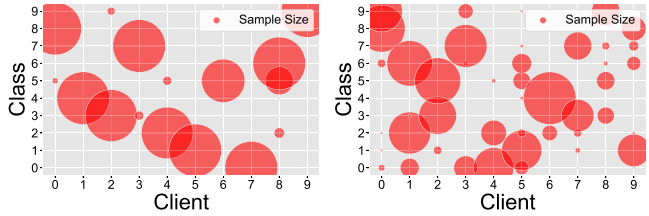
[2]https://yann.lecun.com/exdb/mnist/
[3]http://ufldl.stanford.edu/housenumbers/
[4]https://github.com/zalandoresearch/fashion-mnist
[5]https://www.cs.toronto.edu/~kriz/cifar.html

TABLE III
UNLEARNING ACCURACY(%) RESULTS COMPARISON UNDER DIFFERENT DATA DISTRIBUTIONS ζ AND DIFFERENT DATASETS

	ζ	C_r	C_f	Origin		Retrained		Mimir		FedMM		FedGKT	
				D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
MNIST	0.01	0	1	94.76±0.73	93.11±1.25	98.46±0.38	0.04±0.02	**87.92±0.34**	**0.00±0.00**	75.24±2.64	29.32±5.08	83.03±3.67	40.85±5.36
		8	9	93.49±2.08	96.82±1.84	98.52±1.01	0.00±0.00	**92.11±0.02**	**0.00±0.00**	82.89±0.92	0.00±0.00	84.87±1.89	32.15±6.24
	0.1	4	5	99.32±0.53	99.27±0.08	99.58±0.16	50.81±2.03	94.34±3.21	**33.75±5.49**	**95.41±0.23**	15.77±2.24	95.18±0.68	16.23±0.07
		3	6	97.65±1.44	98.05±0.05	97.29±0.59	25.61±0.36	**91.75±2.08**	**33.01±4.63**	85.10±0.11	47.81±0.05	71.12±3.68	45.23±2.54
SVHN	0.01	0	1	96.06±2.34	95.13±1.89	96.02±0.12	4.87±1.26	**96.02±0.01**	**1.21±0.08**	96.02±0.01	0.00±0.00	95.98±0.03	0.00±0.00
		8	9	96.03±0.85	99.99±0.01	96.17±0.48	0.00±0.00	**95.99±0.01**	**0.00±0.00**	95.05±0.05	0.20±0.16	93.67±0.06	0.00±0.00
	0.1	4	5	98.14±2.08	79.74±4.69	97.19±0.23	69.70±0.68	**98.03±1.64**	**46.91±2.51**	95.99±0.08	36.98±0.24	92.53±1.52	30.58±6.43
		3	6	85.19±4.76	80.37±3.99	84.40±1.45	54.40±0.69	**83.19±1.10**	**21.93±0.54**	82.77±0.04	0.00±0.00	71.31±1.60	3.82±0.42
FASHION	0.01	0	1	97.39±1.65	96.60±0.72	99.99±0.01	4.58±0.12	**100.0±0.00**	**9.20±0.92**	98.42±0.10	12.67±0.25	99.50±0.03	15.49±0.24
		8	9	96.99±0.45	99.13±0.04	99.99±0.00	2.83±0.03	**99.05±0.01**	5.90±0.12	96.40±0.06	12.37±0.06	90.69±2.51	**4.73±1.38**
	0.1	4	5	95.17±0.66	94.75±0.21	98.18±0.08	11.70±2.31	93.24±2.52	15.31±3.90	**97.37±0.36**	15.78±1.90	92.20±0.63	**15.27±0.32**
		3	6	90.06±2.02	98.68±1.08	90.29±0.82	65.04±3.39	**77.69±5.24**	**47.52±5.87**	76.26±2.52	47.36±1.86	67.05±5.02	10.72±5.23
CIFAR10	0.01	0	1	98.78±0.06	100.00±0.00	98.78±0.05	0.00±0.00	**98.78±0.05**	**0.00±0.00**	97.34±0.24	2.09±0.14	98.53±0.02	0.00±0.00
		8	9	81.66±3.25	99.97±0.02	80.65±2.51	0.00±0.25	**79.04±4.23**	**0.00±0.00**	74.82±0.90	14.46±1.83	75.70±1.06	0.00±0.00
	0.1	4	5	72.66±3.67	66.71±4.20	81.19±5.67	6.54±3.39	**71.01±2.38**	**7.07±0.67**	65.99±2.01	7.82±0.24	61.76±1.32	20.02±0.98
		3	6	69.08±3.24	63.36±6.02	91.61±0.52	3.26±1.43	**63.22±2.02**	2.19±0.54	51.78±3.44	0.00±0.00	54.44±10.02	**1.88±1.04**

metrics include: i) *Accuracy*: comparing the accuracy of D_f and D_r for the forgotten client C_f and the retained client C_r to that of the retrained model, aiming for close resemblance. ii) *Membership Inference Attack (MIA):* assessing privacy leakage by determining if a data point was used in training on C_f, framed as a binary classification problem with a logistic regressor [47] attacker. The threat model is trained based on $f_{P,r}$ extracted from client C_r's local data D_r using unlearned model, with the labels indicate whether this data was used during training [1], [48]. Our goal is to ensure that the threat model cannot successfully attack the unlearned client C_f, achieving an attack accuracy similar to that on a retrained model. This demonstrates reduced attack capability, as the threat model's accuracy should approach random guessing. iii) *Backdoor Attack Success Rate (ASR)*: evaluating the effectiveness of unlearning through a backdoor attack. A 3×3 white square is used as a trigger at the bottom-right corner of the most frequent class in D_f, with corresponding labels flipped to a target class $y_{\text{backdoor}} = (y+1)\%U$. During federated training, the global model inadvertently learns both the main and backdoor tasks. After applying Mimir's unlearning to C_f, the model is tested on the backdoor samples to assess whether the attack has been mitigated. ASR is defined as the proportion of backdoor inputs misclassified into y_{backdoor}. A significant drop in ASR, ideally matching that of a retrained model without C_f, indicates that Mimir effectively eliminates the malicious influence of C_f. iv) *Weight Distance:* measuring the difference between the parameters of the retrained and unlearned model using L2 Norm Distance, where smaller differences indicate the unlearned model aligns more closely with the retrained model with better unlearning effectiveness [49]. v) *Runtime*: measuring the time efficiency of executing the unlearning process.

5) Heterogeneous Settings and Hyperparameters: For our main experiments, we utilized 10 clients with a joining ratio α set to 1 per round. We simulated the FL environment using label shift statistical heterogeneity, encompassing both pathological and practical settings as described in prior work [3], [23]. In the case of *pathological label skew*, data with varying labels and sizes was sampled for each client. For *practical label skew*, a Dirichlet distribution (denoted as $Dir(\zeta)$) with the concentration parameter $\zeta = 0.01$ or $\zeta = 0.1$ was used. The dataset with $\zeta = 0.01$ is highly imbalanced, aligning with the experimental setup in [8], where some clients have a majority of data from a few classes and others have very sparse data. This setup simulates class unlearning to some extent, where forgetting a single client closely approximates the process of class unlearning. For the default hyperparameter setting, the learning rate lr was set to 0.005 with $T_l = 3$. The regularization parameters were $\lambda_1 = \lambda_2 = 0.1$. The batch size is set to 32. During the unlearning phase, $\tau = 2$ and the loss function hyperparameters were $\beta = 5.0$ and $\gamma = 2.0$. We provide visualizations of the data distributions across clients on the example datasets CIFAR10 as shown in Fig. 5 under both label skew scenarios.

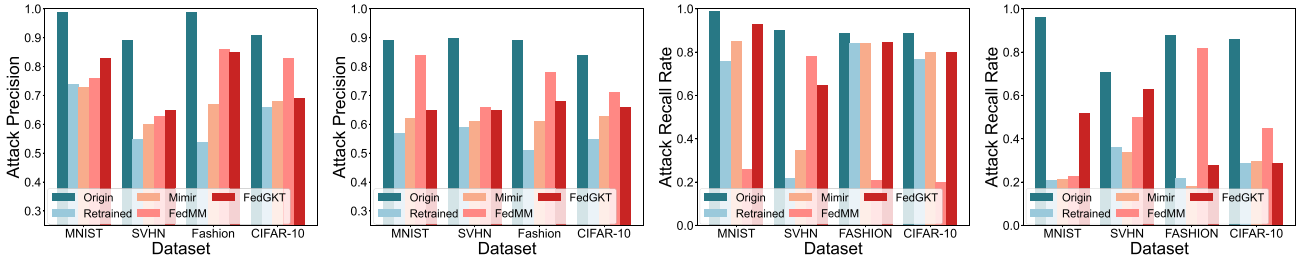


(a) Data Distribution ($\zeta = 0.01$) (b) Data Distribution ($\zeta = 0.1$)

Fig. 5. Client data distribution visualization on CIFAR10.

B. Overall Performance

1) Performance on Retained and Forgotten Data: Table III presents the accuracy of Mimir compared to baseline methods on retained data D_r and forgotten data D_f, with the best values highlighted in bold. However, due to the overlapping label distributions, we had to ensure that unlearning did not result in significant performance drops on D_r. For the learning performance, the "Origin" column shows that Mimir's pFL method performs well in non-IID scenarios. Mimir's unlearning performance is also superior compared to other methods. Specifically, under $\zeta = 0.01$,



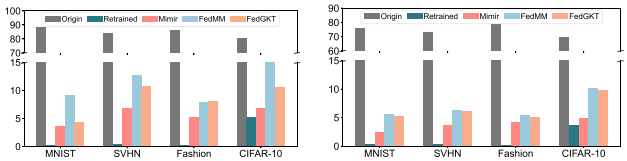
(a) Attack Precision ($\zeta = 0.01$) (b) Attack Precision ($\zeta = 0.1$) (c) Attack Recall ($\zeta = 0.01$) (d) Attack Recall ($\zeta = 0.1$)

Fig. 6. Performance of membership inference attacks (MIAs).

the average difference between Mimir and the retrained model on D_r is 2.46% across four datasets, while FedMM and FedGKT show differences of 6.55% and 5.83%, respectively. On D_f, Mimir's average difference is 1.42%, compared to baselines of 8.56% and 11.33%. Under $\zeta = 0.1$, Mimir's average difference on D_r is 8.61%, while baseline methods show 11.13% and 16.76%. On D_f, Mimir's average difference is 12.81%, while baselines differ by 21.33% and 27.08%. Intuitively, the gating mechanism in FedGKT has limitations in filtering out specific information from clients to be forgotten, which may result in the catastrophic loss of global information and a significant performance drop. Similarly, the noise approach in FedMM inevitably reduces accuracy. In contrast, Mimir effectively balances forgetting and retaining information by disrupting the decision boundary for D_f without affecting the retained clients.

2) Privacy Leakage of Unlearned Model: To evaluate Mimir's effectiveness in privacy protection, we first performed membership inference attacks (MIAs) on the forgotten data D_f under non-IID settings. As shown in Fig. 6, Mimir's unlearned model achieves precision and recall close to the retrained model, outperforming other methods and demonstrating strong unlearning and privacy protection capabilities. However, since approximate unlearning methods still depend on the original model, their privacy leakage is slightly higher than that of retraining. Specifically, as shown in Fig. 6(a) and (b), at $\zeta = 0.01$, FedMM and FedGKT show 9.50% and 7.98% higher MIA precision than Mimir, respectively. At $\zeta = 0.1$, the gap increases to 13.92% and 4.25%. We also measured MIA recall, as shown in Fig. 6(c) and (d). At $\zeta = 0.01$, FedMM and FedGKT achieve 50.17% and 9.55% higher recall than Mimir. On MNIST, FASHION, and CIFAR10, FedMM even exceeds the retrained model, triggering the "Streisand effect" [48], which may unintentionally increase privacy risks. When $\zeta = 0.1$, FedMM and FedGKT still show 21.23% and 14.23% higher recall than Mimir. In summary, Mimir effectively reduces the success rate of MIAs on forgotten data and provides stronger privacy protection.

We then conducted Backdoor Attack experiments on D_f to evaluate Mimir's effectiveness in defending against malicious behavior. As shown in Fig. 7, the original model before unlearning exhibits high ASR across all datasets (for example, 84.7% and 74.2% on average for $\zeta = 0.01$ and $\zeta = 0.1$, respectively), indicating that the backdoor injected by the malicious client C_f successfully influences the global model. After applying Mimir's unlearning process, the ASR drops significantly,



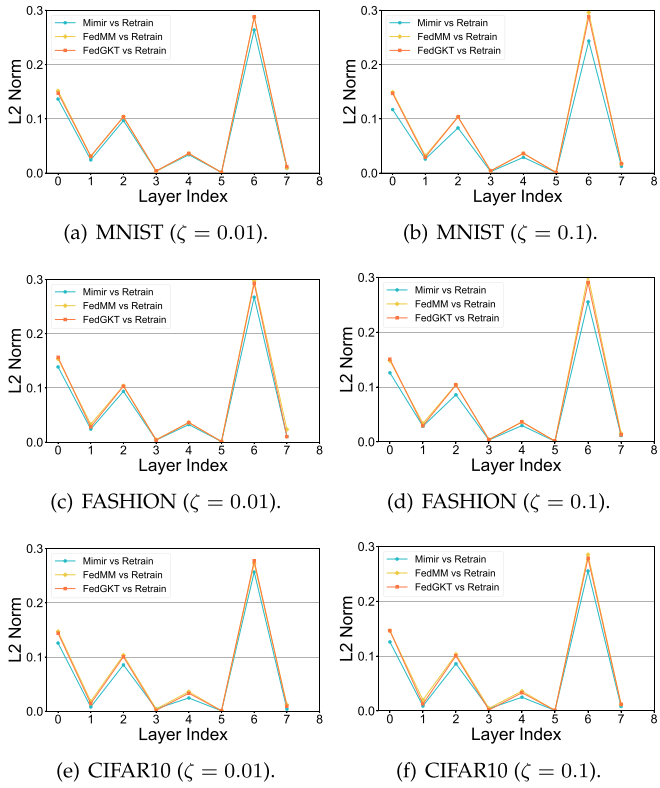
(a) Backdoor ASR $\zeta = 0.01$ (b) Backdoor ASR $\zeta = 0.1$

Fig. 7. Performance of backdoor attack.

approaching the level of a retrained model excluding C_f (e.g., under $\zeta = 0.01$, Mimir achieves an average ASR of 5.6%, while the retrained model achieves 1.6%). This demonstrates that Mimir effectively mitigates the influence of the malicious client. In comparison, existing methods like FedMM and FedGKT retain higher ASR post-unlearning (with average ASRs of 11.3% and 8.5%, respectively), indicating weaker backdoor defense. A similar trend is observed at $\zeta = 0.1$. Thus, Mimir provides robust defense against malicious backdoor injections, exhibiting superior unlearning capability.

3) Weight Deviation of Unlearned Models: To investigate the differences between the unlearned and the retrained model, we calculated the L2 norm distance between the weights of each layer on the MNIST and FASHION datasets. As shown in Fig. 8, we found that the weights of the unlearned models created using Mimir are generally closer to those of the retrained models, demonstrating better unlearning performance. Specifically, on the MNIST dataset, Mimir's parameter deviation is 8.88% and 8.70% lower than FedMM and FedGKT at $\zeta = 0.01$, and 19.57% and 17.83% lower at $\zeta = 0.1$. On the FASHION dataset, Mimir's parameter deviation is 12.56% and 9.73% lower than FedMM and FedGKT at $\zeta = 0.01$, and 16.24% and 13.92% lower at $\zeta = 0.1$. On the CIFAR10 dataset, when $\zeta = 0.01$, Mimir's deviation is reduced by 16.69% and 13.91% compared to FedMM and FedGKT, respectively. At $\zeta = 0.1$, the reduction further increases to 18.39% and 13.98%. These results indicate Mimir's greater similarity in the parameter space compared to the retrained models.

4) Visualization: We conducted a visualization experiment on MNIST, FASHION, and CIFAR10 under the condition $\zeta = 0.1$, aiming to investigate the structural characteristics of the knowledge extracted from samples D_r and D_f. This visualization experiment not only demonstrates the effectiveness of Mimir in pFL but also reveals the reasons why our federated

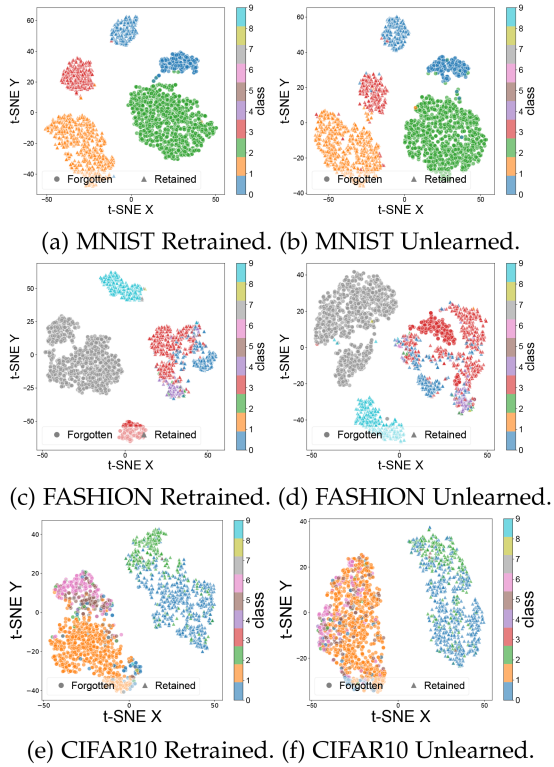


(a) MNIST ($\zeta = 0.01$). (b) MNIST ($\zeta = 0.1$).

(c) FASHION ($\zeta = 0.01$). (d) FASHION ($\zeta = 0.1$).

(e) CIFAR10 ($\zeta = 0.01$). (f) CIFAR10 ($\zeta = 0.1$).

Fig. 8. Deviation of parameters of each layer between unlearned and retrained model under different methods.



(a) MNIST Retrained. (b) MNIST Unlearned.

(c) FASHION Retrained. (d) FASHION Unlearned.

(e) CIFAR10 Retrained. (f) CIFAR10 Unlearned.

Fig. 9. t-SNE visualizations of $\boldsymbol{f}_{P,i}$.

unlearning method does not lead to catastrophic forgetting of classification accuracy for identical label data across clients. Specifically, we employed the t-SNE method [50] to visualize the distribution of personalized feature vectors $\boldsymbol{f}_{P,i}$ in both the retrained and unlearned models. As shown in Fig. 9, the distribution of the unlearned model demonstrates structural similarity with that of the retrained model in the feature space; while they are isomorphic, they are not isometric. By mapping different label data from different clients to distinct state spaces, Mimir's pFL achieves efficient personalized learning in non-IID scenarios. Furthermore, since identical label data from different clients are mapped to separate feature spaces, forgetting the data associated with a specific label from one client does not result in catastrophic forgetting of the same label data retained in other clients. This is because Mimir focuses solely on the state space of the forgotten client's distribution. In conclusion, this visualization explains the potential principles behind Mimir's performance in pFL and client-level unlearning.

C. Comparison With the SOTA Unlearning Methods

In addition to comparing with the data-free methods in Section VI-B1, we also conducted comparisons with four state-of-art federated unlearning methods. As shown in Table V, these methods include FedEraser [1], which accelerates retraining by leveraging the magnitude of historical gradient updates; FedRecovery [8], which uses historical training parameters for differential privacy-based unlearning; Knot [30], which speeds up unlearning through sub-cluster retraining based on asynchronous federated learning; and AdaClipping [51], which removes forgotten client-related parameters via adaptive clipping guided by Fisher information matrices and employs zero-shot KD to recover model performance on remaining data. These four client-level FU methods update models based on historical data stored during training or federated clients with available data. According to the experimental results in Table IV, in IID scenarios, Mimir performs comparably to these methods. For example, across four datasets, Mimir's accuracy deviations after unlearning on D_r and D_f are 2.23% and 1.89%, respectively, while Knot achieves 1.39% and 1.36%, FedEraser achieves 4.10% and 2.46%, AdaClipping achieves 3.77% and 2.68%, and FedRecovery may result in accuracy deviations exceeding 10% on certain datasets. Knot, as the best-performing method, slightly outperforms Mimir due to its access to client data.

In non-IID scenarios, when $\zeta = 0.1$, Mimir's prompt-based learning design in personalized federated learning (pFL) enables higher original training accuracy, particularly on the FASHION dataset, where it outperforms other methods by more than 30%. When $\zeta = 0.01$, Mimir's pFL design further enhances its original training accuracy, surpassing other methods by more than 35% on the SVHN, FASHION, and CIFAR10 datasets. This is because existing unlearning baselines primarily focus on global model unlearning based on FedAvg [3], without optimization for pFL. By integrating pFL, Mimir gains a clear advantage over other FedAvg-based unlearning methods when handling non-IID data. In terms of unlearning effectiveness, Mimir performs on par with other methods in most cases. In the

TABLE IV
PERFORMANCE COMPARISON OF MIMIR WITH STATE-OF-THE-ART FEDERATED CLIENT UNLEARNING METHODS: CONSIDERING IT CANNOT ACCESS THE TRAINING DATASET, MIMIR PERFORMS QUITE WELL

(a) IID Scenario

Dataset	Model	FedEraser		FedRecovery		Knot		AdaClipping		Mimir	
		D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
MNIST	retrain	99.01	98.44	99.01	98.44	99.15	98.48	99.01	98.44	97.97	97.46
	unlearn	96.23	95.84	92.08	91.89	97.66	97.94	97.05	94.80	96.20	94.58
SVHN	retrain	94.07	89.07	94.07	89.07	97.89	89.36	94.07	89.07	92.35	89.81
	unlearn	87.55	87.50	81.24	78.90	94.97	93.18	90.24	88.24	90.16	90.89
FASHION	retrain	93.79	91.09	93.79	91.09	93.97	90.99	93.79	91.09	93.68	90.73
	unlearn	90.36	90.09	86.42	83.78	86.27	85.90	91.02	90.55	92.89	90.04
CIFAR10	retrain	88.73	68.23	88.73	68.23	90.65	87.75	88.73	68.23	89.83	89.11
	unlearn	85.08	63.57	75.34	63.21	85.33	85.28	85.98	62.54	85.67	86.20

(b) non-IID Scenario ($\zeta = 0.01$)

Dataset	Model	FedEraser		FedRecovery		Knot		AdaClipping		Mimir	
		D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
MNIST	retrain	86.48	0.00	86.48	0.00	83.36	0.00	86.48	0.00	98.52	0.00
	unlearn	77.06	0.00	77.32	0.00	82.52	0.00	78.36	0.00	92.11	0.00
SVHN	retrain	55.31	0.00	55.31	0.00	50.23	0.00	55.31	0.00	96.17	0.00
	unlearn	48.25	0.00	50.68	0.00	50.16	0.00	49.24	2.13	95.99	0.00
FASHION	retrain	51.18	11.03	51.18	11.03	49.54	8.65	51.18	11.03	99.99	2.83
	unlearn	48.65	16.32	46.38	14.78	48.60	5.45	48.98	18.25	99.05	5.90
CIFAR10	retrain	43.09	0.00	43.09	0.00	42.68	0.00	43.09	0.00	80.65	0.00
	unlearn	39.88	0.00	41.36	0.00	37.23	0.00	40.02	5.68	79.04	0.00

(c) non-IID Scenario ($\zeta = 0.1$)

Dataset	Model	FedEraser		FedRecovery		Knot		AdaClipping		Mimir	
		D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
MNIST	retrain	98.72	87.79	98.72	87.79	97.37	88.72	98.72	87.79	97.29	25.61
	unlearn	77.16	65.89	90.78	82.36	94.34	85.46	80.24	72.56	91.75	33.01
SVHN	retrain	84.26	59.26	84.26	59.26	81.95	79.05	84.26	59.26	84.40	54.40
	unlearn	57.81	42.94	70.12	40.25	80.31	77.63	67.52	40.32	83.19	21.93
FASHION	retrain	58.46	22.49	58.46	22.49	58.22	12.55	58.46	22.49	90.29	65.04
	unlearn	42.32	15.70	49.56	10.98	48.09	38.21	45.36	12.32	77.69	47.52
CIFAR10	retrain	52.79	39.58	52.79	39.58	71.31	33.89	52.79	39.58	81.19	6.54
	unlearn	44.10	34.54	40.78	32.08	53.41	33.91	43.26	42.36	71.01	7.07

TABLE V
A COMPARISON BETWEEN THE MIMIR WITH EXISTING STATE-OF-THE-ART FU METHODS

Method	All training samples?	Subset of all samples?	Historical data stored during FL?	Data-free?
FedEraser	✔	✘	✔	✘
FedRecovery	✘	✘	✔	✘
Knot	✘	✔	✘	✘
AdaClipping	✘	✘	✔	✘
Mimir	✘	✘	✘	✔

$\zeta = 0.1$ scenario, after unlearning, Mimir's accuracy deviations on D_r and D_f average 7.38% and 14.48%, respectively, while FedEraser averages 18.21% and 12.51%, FedRecovery averages 10.75% and 10.86%, Knot averages 8.18% and 7.59%, and AdaClipping averages 14.46% and 10.39%. In the $\zeta = 0.01$ scenario, after unlearning, Mimir's accuracy deviations on D_r and D_f are 2.29% and 0.77%, respectively, while FedEraser records 5.56% and 1.32%, FedRecovery records 5.08% and 0.94%, Knot records 1.82% and 0.82%, and AdaClipping records 4.87% and 1.81%. It is worth noting that while Mimir may exhibit slightly worse performance in certain metrics compared to other methods due to the lack of access to historical gradients, Fisher matrices, and other auxiliary information, it remains capable of performing data-free unlearning even when other methods fail to work due to the unavailability of training data or historical updates, achieving satisfactory results under a more stringent setting.

D. Runtime Efficiency

We evaluate the runtime efficiency of Mimir compared to Retrain, FedEraser [1], FedRecovery [8], Knot [30], and AdaClipping on MNIST ($\zeta = 0.01$) and CIFAR10 ($\zeta = 0.1$). The results as in Fig. 10 show that all the unlearning methods significantly accelerate the process compared to full retraining. Mimir achieves a $6.49 - 7.65\times$ speedup over Retrain with its frozen prompt generator and lightweight GAN architecture. Compared to FedEraser and Knot, which leverage historical gradients or client selection for acceleration, Mimir achieves $4.31 - 5.33\times$ and $3.73 - 4.78\times$ additional speedups, respectively. FedRecovery achieves the fastest acceleration since it does not require

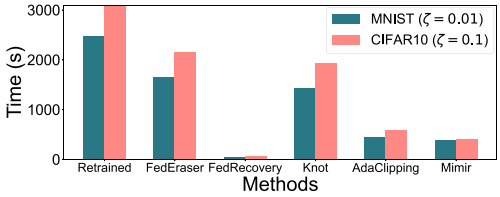


Fig. 10. Runtime comparison among different FUs.

TABLE VI
ACCURACY AND MEMORY USAGE COMPARISON BETWEEN MIMIR AND OTHER PFL METHODS

	FedAvg	FedProx	Per-FedAvg	FedPer	GPFL	Mimir
Acc ($\zeta = 0.01$) (%)	29.94	30.10	35.54	53.82	98.71	92.02
Acc ($\zeta = 0.1$) (%)	40.89	52.09	65.06	67.82	75.96	72.90
Average Mem (MB)	384.21	476.46	914.67	426.38	527.27	485.96

retraining and benefits from historical gradient information. However, due to the lack of access to data and historical updates, Mimir relies on GAN-based knowledge distillation, leading to a slightly longer execution time. Compared to other methods that also employ knowledge distillation, such as AdaClipping, which incurs additional costs from Fisher information matrix calculations, Mimir achieves a 13%–31% speedup. These highlight Mimir's efficiency in data-free FU scenarios.

E. Comparison With the pFL Methods

We compared Mimir's pFL performance with several mainstream approaches under non-IID conditions on CIFAR10, including standard FedAvg [3] and four pFL methods, FedProx [52], Per-FedAvg [27], FedPer [53], and GPFL [23]. As shown in Table VI, Mimir achieves significantly higher accuracy than FedProx, Per-FedAvg, and FedPer in imbalanced data scenarios, outperforming them by over 40% under $\zeta = 0.01$ and by 23.87%, 10.90%, and 8.14% under $\zeta = 0.1$, respectively. This advantage stems from Mimir's explicit modeling of client-specific data features, which methods like FedProx and Per-FedAvg lack. Although FedPer opts to update only specific layers locally to explicitly model increased client personalization, these layers lack the personalized global guidance. Compared to GPFL, Mimir's accuracy is lower, with a gap of 6.69% under $\zeta = 0.01$ and 3.06% under $\zeta = 0.1$, as GPFL leverages client label distributions for conditional computations. In terms of memory overhead, Mimir exceeds FedProx and FedPer due to client-specific prompts but remains lower than GPFL's extensive conditional modules and Per-FedAvg's resource-intensive second-order derivative calculations. Overall, Mimir balances strong pFL performance with unlearning capabilities, making it ideal for privacy-sensitive deployments.

F. Ablation Study

We first validated the effectiveness of the various components in the design of Mimir and then evaluated the sensitivity to hyperparameters β, γ from (18), and L2 regularization parameters λ_1, λ_2 from (12).

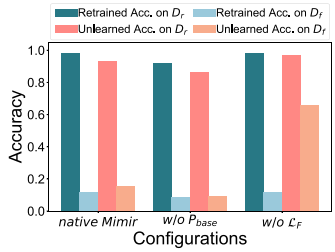


Fig. 11. Ablation study: Impact of P_{base} and $\mathcal{L}_F$.

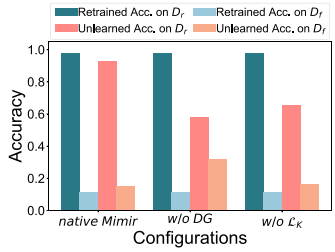
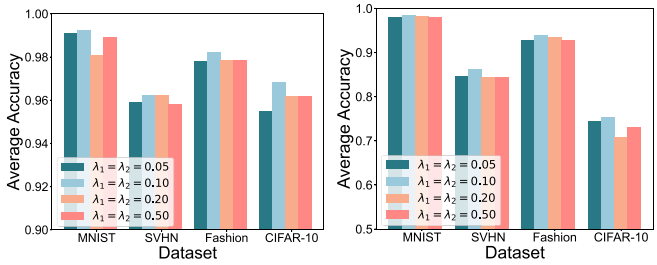


Fig. 12. Ablation study: Impact of DG and $\mathcal{L}_K$.



(a) Learning Acc. ($\zeta = 0.01$). (b) Learning Acc. ($\zeta = 0.1$).

Fig. 13. Impact of λ_1, λ_2 on pFL performance.

Impact of $P_{base}, \mathcal{L}_F$: We conducted ablation experiments on the SVHN dataset ($\zeta = 0.1$) using two variants of Mimir: (a) *w/o* P_{base} and (b) *w/o* $\mathcal{L}_F$ (where *w/o* stands for "without"). The results, as shown in Fig. 11, indicate that (a) removing the model-agnostic prompt (P_{base}) led to a 6.29% decrease in learning accuracy. (b) Without $\mathcal{L}_F$, the effectiveness of unlearning significantly declined. During the unlearning phase, the accuracy on the D_f dataset was 53.70% higher than the retrained model, demonstrating that $\mathcal{L}_F$ is crucial for disrupting the decision boundaries of the forgotten client to achieve effective unlearning.

Impact of $DG, \mathcal{L}_K$: We conducted ablation experiments on SVHN ($\zeta = 0.1$) to evaluate the importance of the data generator (DG) and KL divergence loss ($\mathcal{L}_K$) in Mimir. Two variants were tested: (a) *w/o* DG, where the data generator was replaced with random noise generation (eliminating adversarial training), and (b) *w/o* $\mathcal{L}_K$, where the KL divergence loss was removed from knowledge distillation. The results, shown in Fig. 12, reveal that (a) without DG, accuracy on D_r decreased dramatically by 39.93%, while the gap between D_f and the retrained model widened to 20.52%. This demonstrates that DG plays a critical

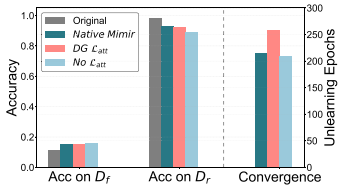


Fig. 14. Ablation study: Impact of $\mathcal{L}_{att}$.

TABLE VII
IMPACT OF β ON UNLEARNING PERFORMANCE

(a) Under $\zeta = 0.01$ Scenario

		$\beta = 0.5$		$\beta = 1.0$		$\beta = 5.0$		$\beta = 10.0$		Origin		Retrained	
C_r	C_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
0	**1**	99.82	9.20	99.98	9.20	**100.00**	**9.20**	100.00	9.20	97.39	96.60	99.99	4.58
8	**9**	78.13	0.00	85.71	0.00	**99.05**	**5.90**	95.87	11.72	96.99	99.13	99.99	2.83

(b) Under $\zeta = 0.1$ Scenario

		$\beta = 0.5$		$\beta = 1.0$		$\beta = 5.0$		$\beta = 10.0$		Origin		Retrained	
C_r	C_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
4	**5**	86.82	15.41	89.95	15.94	93.24	**15.31**	**96.99**	18.62	95.17	94.75	98.18	11.70
3	**6**	60.11	13.01	63.85	24.78	**77.69**	**47.52**	69.22	44.74	90.06	98.68	90.26	65.04

role in generating high-quality pseudo-data that effectively simulates the distribution of retained clients while masking forgotten client features. (b) Similarly, removing $\mathcal{L}_K$ caused D_r accuracy to drop by 32.67%, confirming that the KL divergence loss is essential for knowledge transfer, as using $\mathcal{L}_{att}$ alone proves insufficient for preserving knowledge from C_f.

Impact of $\mathcal{L}_{att}$: We conducted study on the FASHION dataset under setting $\zeta = 0.1$, comparing two variants of Mimir: (a) $DG\ \mathcal{L}_{att}$, where $\mathcal{L}_{att}$ is included in the generator's loss, i.e., $\mathcal{L}_{DG} = \mathcal{L}_F + \beta\mathcal{L}_K + \gamma\mathcal{L}_{att}$; and (b) $No\ \mathcal{L}_{att}$, where $\mathcal{L}_{att}$ is excluded from the student model's loss, i.e., $\mathcal{L}_S = \mathcal{L}_F + \beta\mathcal{L}_K$. As shown in Fig. 14, incorporating $\mathcal{L}_{att}$ in the generator loss yields unlearning performance on both D_f and D_r that is comparable to Native Mimir. However, due to the additional computational overhead from layer-wise similarity calculations and the resulting instability in the adversarial equilibrium, the number of training rounds increased by 19.72%. On the other hand, the $No\ \mathcal{L}_{att}$ variant resulted in a 4.27% decrease in performance on D_f compared to Native Mimir. These results demonstrate that omitting $\mathcal{L}_{att}$ from the generator's objective optimally balances training efficiency and stability without compromising the core unlearning performance, while incorporating $\mathcal{L}_{att}$ in $\mathcal{L}_S$ helps enhance performance on D_f.s

Impact of β, γ*:* Table VII shows the effect of varying β (0.5, 1.0, 5.0, 10.0) with fixed $\gamma = 2.0$ on FASHION under $\zeta = 0.01/0.1$. Results indicate that $\beta = 5.0$ achieves the closest performance to the retrained model, minimizing the accuracy gap on D_r. For instance, under $\zeta = 0.01$, $C_r = 8$, and $C_f = 9$, the gap for $\beta = 0.5$, 1.0, and 10.0 exceeds that for $\beta = 5.0$ by 20.92%, 13.34%, and 3.18%, respectively. Similarly, Table VIII explores varying γ (0.1, 0.5, 2.0, 10.0) with fixed $\beta = 5.0$. When $\gamma = 2.0$, Mimir delivers stable and optimal unlearning performance, avoiding inefficiencies observed at $\gamma = 0.1$ and performance declines at $\gamma = 10.0$.

TABLE VIII
IMPACT OF γ ON UNLEARNING PERFORMANCE

(a) Under $\zeta = 0.01$ Scenario

		$\gamma = 0.1$		$\gamma = 0.5$		$\gamma = 2.0$		$\gamma = 10.0$		Origin		Retrained	
C_r	C_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
0	**1**	100.00	9.20	100.00	9.20	**100.00**	**9.20**	100.00	9.20	97.39	96.60	99.99	4.58
8	**9**	98.21	7.09	91.37	0.15	**99.05**	5.90	78.06	**1.77**	96.99	99.13	99.99	2.83

(b) Under $\zeta = 0.1$ Scenario

		$\gamma = 0.1$		$\gamma = 0.5$		$\gamma = 2.0$		$\gamma = 10.0$		Origin		Retrained	
C_r	C_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
4	**5**	**95.60**	19.11	93.08	18.69	93.24	15.31	89.82	**14.93**	95.17	94.75	98.18	11.70
3	**6**	77.14	82.96	64.29	23.71	**77.69**	**47.52**	65.61	44.32	90.06	98.68	90.26	65.04

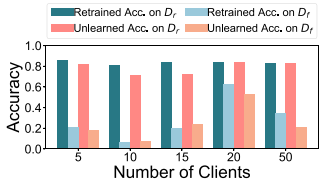


Fig. 15. Scalability study: Performance impact of client number.

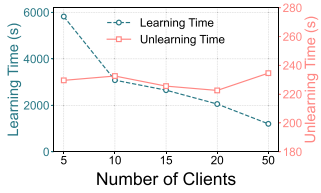


Fig. 16. Scalability study: Runtime impact of client number.

Impact of λ_1, λ_2*:* Fig. 13 examines L2 regularization ($\lambda_1 = \lambda_2 = 0.05, 0.10, 0.20, 0.50$) on learning performance. Setting $\lambda_1 = \lambda_2 = 0.10$ consistently yields the best accuracy across $\zeta = 0.01/0.1$, improving average accuracy by 0.57–1.77% over other values. Excessive regularization ($\lambda_1 = \lambda_2 = 0.20/0.50$) degrades accuracy due to over-penalization. Results confirm $\lambda_1 = \lambda_2 = 0.10$ as the optimal choice for balancing regularization and model flexibility.

G. Scalability Study

To explore the impact of client numbers on Mimir's learning and unlearning performance, we conducted scalability experiments in a non-IID scenario ($\zeta = 0.1$) with 5/10/15/20/50 clients. The performance results on the CIFAR10 dataset are shown in Fig. 15. As the number of clients varied, learning accuracy remained relatively stable. During the unlearning process, performance on forgotten data D_f declined as the number of clients increased. With 50 clients, the performance gap reached 13.64%. This is because Mimir's peer-to-peer unlearning design may lead to shared forgetting among clients with similar prompts. Furthermore, in terms of runtime, Mimir demonstrates strong scalability in both learning and unlearning phases. As shown in Fig. 16, we conducted additional scalability experiments on CIFAR10 ($\zeta = 0.1$). For unlearning, the peer-to-peer adversarial distillation mechanism localizes computations between the retained client (C_r) and forgotten

TABLE IX
UNLEARNING ACCURACY(%) RESULTS COMPARISON UNDER DIFFERENT BACKBONE MODELS AND DIFFERENT DATASETS

	Backbone Model	Origin		Retrained		Mimir		FedMM		FedGKT	
		D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f	D_r	D_f
SVHN	LeNet	98.03	81.04	98.03	9.15	97.34	**5.62**	**97.98**	33.16	51.97	0.00
	ResNet9	98.04	87.51	98.02	41.68	**98.03**	**36.98**	98.03	16.98	98.03	32.57
	ViT-Tiny	99.09	94.23	99.14	87.54	**98.54**	**78.63**	98.09	56.08	94.52	90.04
FASHION	LeNet	96.54	96.24	90.74	12.53	**90.22**	**16.10**	69.41	16.13	89.52	16.11
	ResNet9	98.10	94.87	97.74	15.58	**96.58**	**16.06**	89.37	21.47	95.48	16.07
	ViT-Tiny	98.77	94.33	97.01	29.60	**95.64**	25.01	94.43	32.13	92.79	**30.65**
CIFAR10	LeNet	74.89	73.64	81.36	5.08	**71.01**	**7.07**	67.46	2.71	71.01	10.92
	ResNet9	75.17	74.18	81.45	21.07	**70.77**	**17.84**	52.52	16.15	56.87	6.33
	ViT-Tiny	98.34	98.38	98.55	96.37	**97.98**	**65.68**	95.23	44.78	88.64	52.31

client (C_f), which ensures stable processing time per request (230 s for 5–50 clients on CIFAR10), independent of the total number of clients. For learning, the total training time decreases as the number of clients increases, primarily due to parallelized local training and reduced data per client, which shortens local training time, despite the linearly increasing server aggregation overhead. Mimir's time scaling pattern is similar to FedAvg, except that Mimir requires additional aggregation of P_{base} and G_p and personalized prompt generation, which may introduce additional overhead. However, its overall trend remains consistent with FedAvg. By adopting strategies such as dynamic client selection and employing hierarchical federated architectures [54], [55], the communication overhead introduced by large-scale deployments can be further mitigated. For resource-constrained devices, the system supports lightweight model substitution and dynamic adjustment of local training epochs. In high-latency federated learning environments, Mimir can employ asynchronous aggregation strategies [56], [57] to better accommodate unstable network conditions. We plan to further explore these optimization techniques in future work.

H. Backbone Robustness Study

To evaluate the robustness of Mimir's learning and unlearning methods across different backbone models, we conducted experiments using LeNet [58], ResNet9 [33], and the pre-trained ViT-Tiny [59]. Table IX shows the performance on retained data D_r and forgotten data D_f for the SVHN, FASHION, and CIFAR10 datasets with $\zeta = 0.1$. For learning, the "Origin" results confirm that Mimir performs well under non-IID settings. For unlearning, Mimir consistently achieves smaller differences from the retrained model compared to baseline methods. Specifically, with LeNet, Mimir's average difference on D_r is 3.85%, while FedMM and FedGKT show 11.76% and 19.21%, respectively. On D_f, Mimir's difference is 3.03%, compared to 9.99% and 6.19%. With ResNet9, Mimir's difference on D_r is 3.95% (versus 12.44% and 8.95%), and on D_f is 2.80% (versus 11.84% and 8.11%). For ViT-Tiny, Mimir achieves 0.84% on D_r (versus 2.32% and 6.25%) and 14.73% on D_f (versus 28.53% and 15.87%). Overall, Mimir maintains higher similarity to the retrained model, showing better performance than baseline methods.

VII. CONCLUSION

In this paper, we propose a federated unlearning framework under the pFL setting, addressing the challenges posed by stringent privacy regulations in client data-free scenarios. We present Mimir, a unified learning-unlearning framework that leverages client-specific prompts for personalized feature extraction. During the unlearning phase, Mimir executes knowledge distillation based on generative adversarial training to eliminate the feature extraction capability of the forgotten client without compromising the performance of the remain client, thus avoiding catastrophic unlearning. Incorporating personalization not only preserves strong learning performance in non-IID settings but also serves as the foundation for the unlearning process. Empirical evaluations on benchmark datasets validate the effectiveness of Mimir. Our work lays the foundation for future research in federated unlearning, particularly in increasingly stringent operational environments.

REFERENCES

[1] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "FedEraser: Enabling efficient client-level data removal from federated learning models," in *Proc. IEEE/ACM 29th Int. Symp. Qual. Serv.*, 2021, pp. 1–10.
[2] L. Wu, S. Guo, J. Wang, Z. Hong, J. Zhang, and Y. Ding, "Federated unlearning: Guarantee the right of clients to forget," *IEEE Netw.*, vol. 36, no. 5, pp. 129–135, Sep./Oct. 2022.
[3] 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.*, 2017, pp. 1273–1282.
[4] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR)," *Intersoft Consulting*, vol. 24, no. 1, pp. 1–8, 2018.
[5] L. Bourtoule et al., "Machine unlearning," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 141–159.
[6] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proc. ACM Web Conf.*, 2022, pp. 622–632.
[7] Y. Lin, Z. Gao, H. Du, D. Niyato, J. Kang, and X. Liu, "Incentive and dynamic client selection for federated unlearning," in *Proc. ACM Web Conf.*, 2024, pp. 2936–2944.
[8] L. Zhang, T. Zhu, H. Zhang, P. Xiong, and W. Zhou, "FedRecovery: Differentially private machine unlearning for federated learning frameworks," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 4732–4746, 2023.
[9] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1749–1758.
[10] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023.
[11] W. Huang, M. Ye, and B. Du, "Learn from others and be yourself in heterogeneous federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10143–10153.

[12] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.
[13] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Zero-shot machine unlearning," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 2345–2354, 2023.
[14] C. Zhang, S. Shen, Y. Zhao, W. T. Chen, and M. Xu, "GENIU: A restricted data access unlearning for imbalanced data," 2024, *arXiv:2406.07885*.
[15] Y. Li, C. Chen, X. Zheng, and J. Zhang, "Federated unlearning via active forgetting," 2023, *arXiv:2307.03363*.
[16] Z. Liu et al., "A survey on federated unlearning: Challenges, methods, and future directions," *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–38, 2024.
[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[18] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16816–16825.
[19] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *Int. J. Comput. Vis.*, vol. 130, no. 9, pp. 2337–2348, 2022.
[20] F.-E. Yang, C.-Y. Wang, and Y.-C. F. Wang, "Efficient model personalization in federated learning via client-specific prompt generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 19159–19168.
[21] Y. Xu, Y. Liao, L. Wang, H. Xu, Z. Jiang, and W. Zhang, "Overcoming noisy labels and non-IID data in edge federated learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11406–11421, Dec. 2024.
[22] R. Zhang, Y. Chen, C. Wu, and F. Wang, "Multi-level personalized federated learning on heterogeneous and long-tailed data," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 12396–12409, Dec. 2024.
[23] J. Zhang et al., "GPFL: Simultaneously learning global and personalized feature information for personalized federated learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 5041–5051.
[24] J. Zhang et al., "FedCP: Separating feature information for personalized federated learning via conditional policy," in *Proc. ACM Conf. Knowl. Discov. Data Mining*, 2023, pp. 3249–3261.
[25] S. Fu, Z. Yang, C. Hu, and W. Bao, "Personalized federated learning with contrastive momentum," *IEEE Trans. Big Data*, vol. 1, pp. 1–11, 2024.
[26] D. Chen, L. Yao, D. Gao, B. Ding, and Y. Li, "Efficient personalized federated learning via sparse model-adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 5234–5256.
[27] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 3557–3568.
[28] Y. Tan et al., "FedProto: Federated prototype learning across heterogeneous clients," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8432–8440.
[29] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Fast yet effective machine unlearning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 13046–13055, Sep. 2024.
[30] N. Su and B. Li, "Asynchronous federated unlearning," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[31] Y. Zhao, J. Yang, Y. Tao, L. Wang, X. Li, and D. Niyato, "A survey of federated unlearning: A taxonomy, challenges and future directions," 2023, *arXiv:2310.19218*.
[32] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[33] 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.
[34] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 2089–2099.
[35] P. Micaelli and A. J. Storkey, "Zero-shot knowledge transfer via adversarial belief matching," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9547–9557.
[36] M. Jia et al., "Visual prompt tuning," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2022, pp. 709–727.
[37] R. Gao et al., "Zero-VAE-GAN: Generating unseen features for generalized and transductive zero-shot learning," *IEEE Trans. Image Process.*, vol. 29, pp. 3665–3680, 2020.
[38] H. Gao, J. Li, and H. Huang, "On the convergence of local stochastic compositional gradient descent with momentum," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 7017–7035.
[39] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," 2019, *arXiv: 1910.14425*.
[40] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 630–641, Mar. 2022.
[41] A. Xiong et al., "A multi-task based clustering personalized federated learning method," *Big Data Mining Analytics*, vol. 7, no. 4, pp. 1017–1030, 2024.
[42] J. Zhang et al., "FedALA: Adaptive local aggregation for personalized federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 11237–11244.
[43] X. Ma, J. Zhang, S. Guo, and W. Xu, "Layer-wised model aggregation for personalized federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10092–10101.
[44] F. Hanzely, B. Zhao, and M. Kolar, "Personalized federated learning: A unified framework and universal optimization techniques," 2021, *arXiv:2102.09743*.
[45] Y. Tao, C.-L. Wang, M. Pan, D. Yu, X. Cheng, and D. Wang, "Communication efficient and provable federated unlearning," 2024, *arXiv:2401.11018*.
[46] Y. Deng and M. Mahdavi, "Local stochastic gradient descent ascent: Convergence analysis and communication efficiency," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 1387–1395.
[47] H. Yan et al., "Membership inference attacks against deep learning models via logits distribution," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 3799–3808, Sep./Oct. 2023.
[48] M. Chen, W. Gao, G. Liu, K. Peng, and C. Wang, "Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 7766–7775.
[49] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," 2022, *arXiv:2209.02299*.
[50] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
[51] Z. Xie, Z. Gao, Y. Lin, C. Zhao, X. Yu, and Z. Chai, "Adaptive clipping and distillation enabled federated unlearning," in *Proc. IEEE Int. Conf. Web Serv.*, 2024, pp. 748–756.
[52] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.
[53] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv: 1912.00818*.
[54] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4385–4395, Mar. 2022.
[55] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3805–3822, Jul. 2023.
[56] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," *Comput. Sci. Rev.*, vol. 50, 2023, Art. no. 100595.
[57] Z. Ma, Y. Lu, W. Li, and S. Cui, "Asynchronous personalized federated learning with irregular clients," in *Proc. Asian Conf. Mach. Learn.*, 2023, pp. 706–721.
[58] Y. LeCun et al., "Handwritten digit recognition with a back-propagation network," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1989, pp. 396–404.
[59] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your ViT? Data, augmentation, and regularization in vision transformers," 2021, *arXiv:2106.10270*.

Wenhan Wu (Student Member, IEEE) received the BS degree in computer science from Wuhan University, in 2023. He is currently working toward the MS degree in computer science with Wuhan University. His research interests include machine unlearning, federated learning, and edge computing.

Huanghuang Liang received the BS degree in automation engineering from the Anhui University of Technology, in 2016, and the MS degree in automation engineering from the University of Electronic Science and Technology of China, in 2019. He is currently working toward the PhD degree in computer science with Wuhan University. His research interests include cloud computing and Big Data systems.

Tianyu Tu received the BS degree in software engineering from Wuhan University, in 2022. He is currently working toward the MS degree in computer science and technology with Computer School, Wuhan University. His research interests include federated unsupervised learning and game theory.

Jiawei Jiang (Member, IEEE) received the PhD degree in computer science from Peking University, China, in 2018. He is currently a professor with the School of Computer Science, Wuhan University, China. His research interests include database, Big Data management and analytics, and machine learning systems.

Chuang Hu (Member, IEEE) received the BS and MS degrees in computer science from Wuhan University, in 2013 and 2016, respectively, and the PhD degree from the Hong Kong Polytechnic University, in 2019. He is an associate researcher with the School of Computer Science, Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.

Dazhao Cheng (Senior Member, IEEE) received the BS degree in electrical engineering from the Hefei University of Technology, in 2006, the MS degree in electrical engineering from the University of Science and Technology of China, in 2009, and the PhD degree from the University of Colorado at Colorado Springs, in 2016. He was an AP with the University of North Carolina at Charlotte in 2016–2020. He is currently a professor with the School of Computer Science, Wuhan University. His research interests include Big Data and cloud computing.