

Split Unlearning

Yanna Jiang*
University of Technology Sydney
Sydney, NSW, Australia
yanna.jiang@uts.edu.au

Guangsheng Yu*
University of Technology Sydney
Sydney, NSW, Australia
guangsheng.yu@uts.edu.au

Qin Wang
Data61, CSIRO
Sydney, NSW, Australia
University of Technology Sydney
Sydney, NSW, Australia
qin.wang@uts.edu.au

Xu Wang
University of Technology Sydney
Sydney, NSW, Australia
xu.wang@uts.edu.au

Baihe Ma
University of Technology Sydney
Sydney, NSW, Australia
baihe.ma@uts.edu.au

Caijun Sun[†]
Zhejiang Lab
Hangzhou, Zhejiang, China
sun.cj@zhejianglab.com

Wei Ni
Data61, CSIRO
Sydney, NSW, Australia
University of Technology Sydney
Sydney, NSW, Australia
wei.ni@uts.edu.au

Ren Ping Liu
University of Technology Sydney
Sydney, NSW, Australia
renping.liu@uts.edu.au

Abstract

We introduce *Split Unlearning*¹, a novel machine unlearning technology designed for Split Learning (SL), enabling the first-ever implementation of Sharded, Isolated, Sliced, and Aggregated (SISA) unlearning in SL frameworks. Particularly, the tight coupling between clients and the server in existing SL frameworks results in frequent bidirectional data flows and iterative training across all clients, violating the “Isolated” principle and making them struggle to implement SISA for independent and efficient unlearning. To address this, we propose SPLITWIPER with a new *one-way-one-off propagation* scheme, which leverages the inherently “Sharded” structure of SL and decouples neural signal propagation between clients and the server, enabling effective SISA unlearning even in scenarios with absent clients. We further design SPLITWIPER+ to enhance client label privacy, which integrates differential privacy and label expansion strategy to defend the privacy of client labels against the server and other potential adversaries. Experiments across diverse data distributions and tasks demonstrate that SPLITWIPER achieves **0%** accuracy for unlearned labels, and **8%** better accuracy for retained labels than non-SISA unlearning in SL. Moreover, the one-way-one-off propagation maintains *constant* overhead, reducing computational and communication costs by

99%. SPLITWIPER+ preserves **90%** of label privacy when sharing masked labels with the server.

CCS Concepts

• Security and privacy; • Computing methodologies → Machine learning; Distributed computing methodologies;

Keywords

Machine Unlearning, Split Learning, Label Privacy, SISA

ACM Reference Format:

Yanna Jiang, Guangsheng Yu, Qin Wang, Xu Wang, Baihe Ma, Caijun Sun, Wei Ni, and Ren Ping Liu. 2025. Split Unlearning. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*, October 13–17, 2025, Taipei, Taiwan. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3719027.3744787>

1 Introduction

Split Learning (SL) is one of the latest approaches to distributed learning, suitable for scenarios where clients with limited computational resources, such as smartphones or IoT devices, need to offload part of their training to the servers [34, 46, 47]. Existing SL frameworks include two configurations: *Vanilla SL* and *U-shaped SL* [33]. In Vanilla SL, the model is split into two segments: The clients manage the segment receiving the training data and send intermediate values to the server, while the server handles the segment producing the final outputs without accessing raw data. U-shaped SL extends this by eliminating the need for label sharing, protecting both label privacy and data privacy. The model in U-shaped SL is divided into three segments: The clients process the input data and produce output classes, while the server manages the intermediate layers.

The need for the compliance with regulatory requirements for “the right to forget” (e.g., GDPR [48]), which grants individuals the right to remove their personal data or impact of the data be erased,

*Contributed equally to this research.

[†]Corresponding author.

¹Full version: <https://arxiv.org/abs/2308.10422>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CCS '25, Taipei, Taiwan

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1525-9/2025/10
<https://doi.org/10.1145/3719027.3744787>

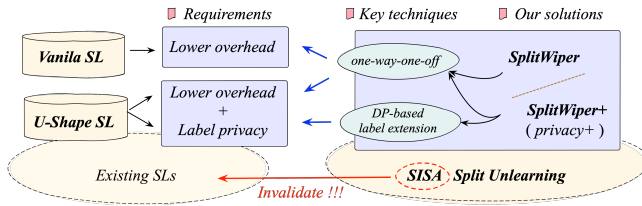


Figure 1: Why we need SPLITWIPER/SPLITWIPER+? The “Isolated” principle of SISA invalidates existing SLs, hindering efficient and effective split unlearning. Our solution that features a novel one-way-one-off design and DP-based label extension strategy enables SISA unlearning and fulfills the fundamental requirements of SLs.

necessitates the development of effective *machine unlearning* [2] techniques. However, the requirement becomes challenging within the context of SL, where the model segments across clients and servers need to be jointly updated. Current methods include re-training the model [19, 23] or using alternatives like fine-tuning, gradient ascent, and knowledge distillation [31, 35, 40]. While these alternatives are favored for their higher unlearning efficiency (i.e. lower *time consumption*) compared to retraining, which prioritizes *effectiveness* (i.e., *accuracy*) [25], they all require all data samples during each update iteration. This becomes problematic when some data samples are absent due to contextual constraints. The **Sharded, Isolated, Sliced, and Aggregated (SISA)** method [2, 4, 8] addresses this by segmenting data and training multiple models in parallel, maintaining higher effectiveness and efficiency than non-SISA unlearning methods.

Existing SL solutions [34, 46, 47, 55] are inadequate for directly applying SISA unlearning in multi-client scenarios, where multiple clients have independent models while sharing a common model on the server. Although the clients’ training data can be considered “Sharded”, SL undermines the “Isolated” requirement of SISA because: (1) *the server model is tightly coupled with the client models through frequent bidirectional propagation*, and (2) *the shared server model is iteratively trained across all clients*. Therefore, a new framework is necessary to decouple the shared server model from the client models, enabling SISA unlearning in multi-client SL environments.

Contributions. In this paper, we address these gaps in enabling SISA unlearning in multi-client SL environments where unlearning can thus be conducted smoothly even with absent clients (cf. Fig. 1).

► We introduce **SPLITWIPER** (Sec.4), a new SL framework that implements the SISA guideline, achieving SISA unlearning capabilities: (i) *selective involvement of only clients who request unlearning* and (ii) *effective unlearning while maintaining comparable utility for retained data*. SPLITWIPER features an efficient and streamlined interaction between clients and the server by incorporating a new low-effort pre-training phase on clients, after which the clients’ weights are frozen (one-off). It allows iterative updates to focus only on the server’s weights without requiring feedback loops or updates to be returned to clients (one-way). This novel *one-way-one-off* protocol architecture for propagating neural signals from clients to the server forms the foundation for enabling SISA unlearning in SPLITWIPER. Our solution removes the requirement for all clients to engage in unlearning processes by Isolating training on each

client, allowing for the formation of distinct **Shards** or **Slices** of data that are Aggregated exclusively on the server.

► We further introduce **SPLITWIPER+** (Sec.5), a privacy-enhancing variation of SPLITWIPER, to address the privacy concerns associated with label sharing during the learning and unlearning tasks. The “Isolated” principle of SISA invalidates traditional privacy-preserving methods, such as non-label-sharing U-shaped SL [47]. SPLITWIPER+ incorporates a novel *label expansion* strategy, marking the first approach to mask the quantity of training labels and their semantics. By expanding, shuffling, and anonymizing real labels, while protecting the outputs of clients’ models with differential privacy (DP), this strategy prevents adversaries from exploiting the exact number of labels and launching effective attacks with supervised learning techniques. This innovative design effectively mitigates privacy leakage when clients’ labels are exposed to the server during propagation.

► We conduct extensive experiments (Sec.7) in various settings where the clients either share the same training task with various data distributions or engage in different tasks, exploring the framework’s adaptability and performance across a range of scenarios. The results show that SPLITWIPER achieves complete unlearning accuracy (0%) while improving retained accuracy by 8%, engaging only the clients who request unlearning to minimize involvement and reduce overhead. SPLITWIPER maintains *constant* overhead, cutting off computational and communication costs by over 99% compared to existing SL frameworks. Additionally, SPLITWIPER+ preserves over 90% of client label privacy from the server.

2 Related Work

Unlearning in distributed environments. Machine unlearning has not been fully explored within current distributed learning frameworks, including SL. While efficient unlearning strategies have gained attention in Federated Learning (FL), which also involves collaboration between clients and a central server. Some researchers have worked on accelerating retraining across all clients, the most robust unlearning method in FL, using optimized algorithms like distributed Newton-type updates [26], SGA-EWC [49], and SFU [20]. Solutions like FedEraser [22] reduce the impact on non-unlearning-involved clients through calibration but still require cooperation from some clients. However, in SL, where client resources are often limited, non-requesting clients may lack the incentive or ability to contribute, especially if they are uninterested in or unable to participate in the unlearning process due to network issues, such as low bandwidth, high latency, or intermittent connectivity. This presents a significant challenge in SL compared to other distributed environments like FL, where developing unlearning methods that accommodate these constraints remains an unresolved issue.

SISA guideline for SL unlearning. SISA [2] is a state-of-the-art guideline for robust and efficient machine unlearning, enabling rapid adaptation to data removal requests without requiring full model retraining. By strategically partitioning data into distinct “shards” linked to different model versions, the SISA guideline allows for the targeted unlearning of specific data shards without requiring the participation of all shards. This method has been successfully applied in various domains, including unlearning in graph

neural networks [8] and large language models [5]. However, SISA unlearning faces significant challenges in multi-client SL scenarios due to the interdependence between client model segments and the shared server model segment, violating the Isolated requirement of SISA. Consequently, the direct application of conventional SISA is impractical for existing SL scenarios.

Label privacy under SL unlearning. Beyond unlearning, privacy concerns regarding intermediate outputs and labels in SL have been widely studied. U-shaped SL [33] avoids label propagation to the server by placing the output layer on the client side but violates SISA’s Isolated requirement by necessitating the return of intermediate values across all clients, disabling the independent unlearning and adding overhead to resource-limited clients. If labels are shared but privacy is required, existing research primarily addresses privacy risks by directly adding noise and obfuscating labels [13, 15] or further protecting intermediate outputs transmitted to the server. Wu et al. [50] introduced a DP mechanism to perturb these transmitted outputs, while Li et al. [21] developed the Marvell method to quantify privacy leakage and optimize noise structures. However, these methods overlook the exposure of sensitive information through label quantity and semantics. Xiao et al. [52] obfuscated both labels and outputs via multiple linear combinations to reduce association with original data but still failed to conceal label quantity, potentially exposing sensitive information. Protecting the quantity of labels is critical in label-sharing SL, as revealing the exact number of labels allows adversaries to exploit supervised learning techniques that achieve higher accuracy by relying on this information, significantly increasing the risk of inference attacks and unauthorized model reconstruction.

Ours. Our design overcomes those challenges (Table 1). SPLITWIPER adapts SISA for multi-client SL, starting with low-effort pre-training on the clients followed by freezing their weights. This allows for one-way-one-off propagation to the server, enabling efficient unlearning without involving non-unlearning-participating clients. Additionally, SPLITWIPER+ enhances label privacy by expanding, shuffling, and anonymizing real labels, with intermediate outputs expanded and protected using a DP-based mechanism.

Table 1: Comparison of existing label privacy methods. Our work provides a practical privacy-preserving solution beyond existing methods by protecting label semantics, label quantity, and intermediate values while enabling effective SISA split unlearning.

Method	Protection Label Semantics	Label Quantity	Intermediate Values	Fit for SISA Split Unlearning
RRWithPrior [15]	✓	✗	✗	✗
LPSC [13]	✓	✗	✗	✗
DP-based Defense [50]	✗	✗	✓	✗
Marvell [21]	✗	✗	✓	✗
MALM [52]	✓	✗	✓	✗
U-shaped SL [33]	✓	✓	✗	✗
Ours	✓	✓	✓	✓

3 Preliminaries

We outline the key background knowledge relevant to SL and machine unlearning. By reviewing existing techniques and identifying their limitations, we set the stage for the challenges addressed in our proposed frameworks.

3.1 Split Learning

SL is a distributed machine learning technique designed to train neural networks without sharing raw data between K clients, each owning datasets $D_o^1, D_o^2, \dots, D_o^K$, and the server that holds no data [33, 39, 51]. A typical SL solution involves two key factors:

- **Topology.** A network is divided into two parts, referred to as *client-side* and *server-side*. Consider the network is a sequence of layers $\{L_1, \dots, L_n\}$. The topology step involves splitting this sequence into two subsets: The client-side layers for K clients are denoted as $\{\{L_1^1, \dots, L_m^1\}, \dots, \{L_1^K, \dots, L_m^K\}\}$ and the server-side layers $\{L_{m+1}, \dots, L_n\}$. The clients and server initialize their respective portions of the network randomly.
- **Training.** Each client k processes its local data through the initial layers of the model, producing an intermediate output $H_k^{(t)} = \mathcal{F}_o^k(X_k; W_k^{(t)})$ through its layers $\{L_1^k, \dots, L_m^k\}$, where $W_k^{(t)}$ represents the weights of client k at iteration t . This output is sent to the server, which continues the forward propagation through its layers $\{L_{m+1}, \dots, L_n\}$ and updates its own weights $W_s^{(t)}$ based on the gradients: $W_s^{(t+1)} = W_s^{(t)} - \eta \Phi \left(\frac{\partial \mathcal{L}_1}{\partial W_s^{(t)}}, \frac{\partial \mathcal{L}_2}{\partial W_s^{(t)}}, \dots, \frac{\partial \mathcal{L}_K}{\partial W_s^{(t)}} \right)$, where $\frac{\partial \mathcal{L}_k}{\partial W_s^{(t)}}$ is the gradient contribution of the loss values from client k and $\Phi(\cdot)$ is a customized aggregation function. The server then sends the gradients to the clients, allowing them to update their weights: $W_k^{(t+1)} = W_k^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial W_k^{(t)}}$.

The above processes enable training without sharing raw data or model details between clients and the server. In multi-client, single-server scenarios, two primary training methods are used: synchronizing model weights between clients after each epoch or alternating training epochs between clients and the server without synchronization [33].

Incremental training process. This process has been utilized by many learning frameworks, including FL [29, 44] and SL, where a model is continuously updated by gradually incorporating new data over time. Each iteration of back-propagation during model training needs to be sent back to every client involved in the training process and builds incrementally on the previous ones, embodying the incremental nature of SL [6]. This process enables the model to dynamically learn and adapt as new data becomes available.

The challenges of applying SISA in multi-client SL scenarios stem from the incremental training process inherent in these setups. This iterative process undermines the isolated requirement of SISA due to the tight coupling between the server and client model segments through regular bidirectional propagation and the shared server model being incrementally trained across all clients. The knowledge to be unlearned, reflected in $W_k^{(t)}$, would have also influenced $W_{k'}^{(t')}$ for all $k' \neq k$ and $t' \geq t$, necessitating the recalculation of corresponding changes in subsequent updates, thereby making it impossible to apply SISA unlearning [24, 25].

3.2 Machine Unlearning

Machine unlearning [27, 31, 35, 37] involves removing the influence of specific data subsets from a trained model while preserving its performance on the remaining data. The goal is to eliminate the

impact of certain labels on the model's predictions without compromising accuracy on the retained data. Let $D = \{(X_i, Y_q)\}_{i=1, q=1}$ represent the dataset, where X_i is the i -th training sample and $Y_q \in \{1, \dots, Q\}$ is its label. Unlearning removes the influence of subset $D_u \subset D$ with labels Y_u while preserving performance on the remaining data $D_r = D \setminus D_u$ with labels Y_o , ensuring Y_u is forgotten and predictions Y_o remain accurate.

Retraining. The most direct and effective, though not efficient, method of implementing unlearning is retraining [19, 23]. In this method, the revoked sample is deleted, and the model is retrained on the original dataset minus the deleted sample, denoted as $D_r = D \setminus ((X_i, Y_q) \in D_u)$. While this method is effective and straightforward, the computational overhead becomes prohibitive for complex models and large datasets.

Alternatives. Referring to fine-tuning, gradient ascent, and knowledge distillation, These unlearning methods are proposed to improve unlearning efficiency compared to retraining, but they still consider retraining as the baseline for unlearning effectiveness [31, 35, 40], including:

- *Fine-tuning.* Adjusting the model parameters by further training on D_r to deliberately induce catastrophic forgetting of D_u , typically using $W^{(t+1)} = W^{(t)} - \eta \nabla \mathcal{L}_r(W^{(t)})$, where \mathcal{L}_r is the loss function computed on the retained data.
- *Gradient ascent.* Updating the weights in the direction of the gradient of a loss function to maximize the objective: $W^{(t+1)} = W^{(t)} + \eta \|\nabla \mathcal{L}(W^{(t)}) - \nabla \mathcal{L}_r(W^{(t)})\|$.
- *Knowledge distillation.* Trading-off between a preserver that aids to retain knowledge and a destroyer that aids in forgetting knowledge: $\mathcal{L} = \alpha \mathcal{L}_u + (1 - \alpha) \mathcal{L}_r$.

SISA. SISA [2] is a guideline designed to enable efficient and independent unlearning for those who request it, without affecting other participants. In SISA, the original training set D_o is divided into K disjoint shards $\{D_o^1, D_o^2, \dots, D_o^K\}$, each used to train a separate model $\{\mathcal{F}_o^1, \mathcal{F}_o^2, \dots, \mathcal{F}_o^K\}$. These models generate individual predictions, which are then aggregated to produce a global prediction. When an unlearning request is made, only the shard containing the specific data sample needs to be retrained, allowing the unlearning process to be conducted independently for the requesting party, without impacting other clients or the overall system.

Why SISA split unlearning. In multi-client SL, the ability to unlearn data despite client absence due to network issues is crucial. Existing frameworks rely on non-SISA unlearning methods, which require retraining or similar processes from the unlearning request point. While these methods prioritize either effectiveness or efficiency, SISA enables independent unlearning for requesting clients without burdening resource-limited devices in the SL network. SISA is seamlessly compatible, allowing prevalent unlearning methods to be applied individually on each client², making it the most practical solution to address limitations in current non-SISA methods.

4 SISA Split Unlearning

In this section, we define the problem and establish clear objectives for split unlearning. We present SPLITWIPER, a SISA-based

framework designed to address existing limitations, offering a feasible and scalable solution for split unlearning.

4.1 Problem Definition

In the context of SL, the training dataset D_o is distributed among K clients and a server, functioning as a training service provider. Throughout this paper, we assume a horizontally non-IID data distribution, where each client possesses a subset of D_o characterized by distinct feature distributions and imbalanced label distributions.

Multi-client settings. We focus on multi-client scenarios. For a trained model \mathcal{F}_o , we can divide it into a client-side part \mathcal{F}_o^k for client k , and a server-side part \mathcal{F}_o^s . Each client k owns its \mathcal{F}_o^k in parallel and shares a single \mathcal{F}_o^s . We define the evaluation of model performance from the perspective of client k as $\mathcal{T}(\text{Concat}(\mathcal{F}_o^k, \mathcal{F}_o^s), D_o^k)$, where \mathcal{T} represents the evaluation function. Each client functions as an individual user managing its local data, aiming to enhance the prediction accuracy of its pre-trained local model by leveraging the server's computational capabilities with the shared server-side model segment. Consequently, the performance assessment focuses on the server's perspective, which aggregates the independent evaluations, $\mathcal{T}(\text{Concat}(\mathcal{F}_o^k, \mathcal{F}_o^s), D_o^k), \forall k$.

Independent unlearning. Accordingly, any client k that raises an unlearning request should be conducted only between this particular client and the server, with no needs of other clients k' ($k' \neq k$) being involved in the unlearning process. This means the server should unlearn the unlearned features of client k from the server-side part \mathcal{F}_o^s upon the intermediate outputs of the updated client-side part \mathcal{F}_o^k . Formally, to respond to the unlearning request for a series of data samples D_u^k raised by client k , the entire SL network needs to satisfy an unlearned model \mathcal{F}_u trained on $D_u = D_o \setminus \{D_u^k\}$. **Challenges of SISA split unlearning.** The split unlearning process faces a conflict between achieving independent unlearning for each client and maintaining overall effectiveness, which we term the *Dilemma of Independence and Effectiveness*, particularly when clients operate independently with limited training resources. Non-SISA methods, depending on full client participation to update shared models, are vulnerable to client absence during unlearning and impose significant overhead on both the initiating client k and other clients k' ($k' \neq k$). This method forces disinterested clients k' to expend resources solely to meet the needs of client k .

The pursuit of SISA split unlearning is to devise an unlearning algorithm fulfilling the following goals:

- **G1: Independent unlearning.** This goal ensures that unlearning affects only the requesting client and server, leaving other clients ($k' \neq k$) and their resources unaffected.
- **G2: Effective unlearning with retained utility.** This is to ensure that an unlearning process thoroughly removes the influence of the unlearned data from the model, while also maintaining the predictive accuracy for the retained data.

4.2 SPLITWIPER

We propose SPLITWIPER with a novel one-way-one-off propagation scheme, enabling SISA split unlearning to achieve **G1** and **G2**. The Sharded training of SPLITWIPER involves multiple clients, each acting as a distinct shard with different data samples. Our

²For simplicity, this paper considers retraining is used on each client (i.e., each shard).

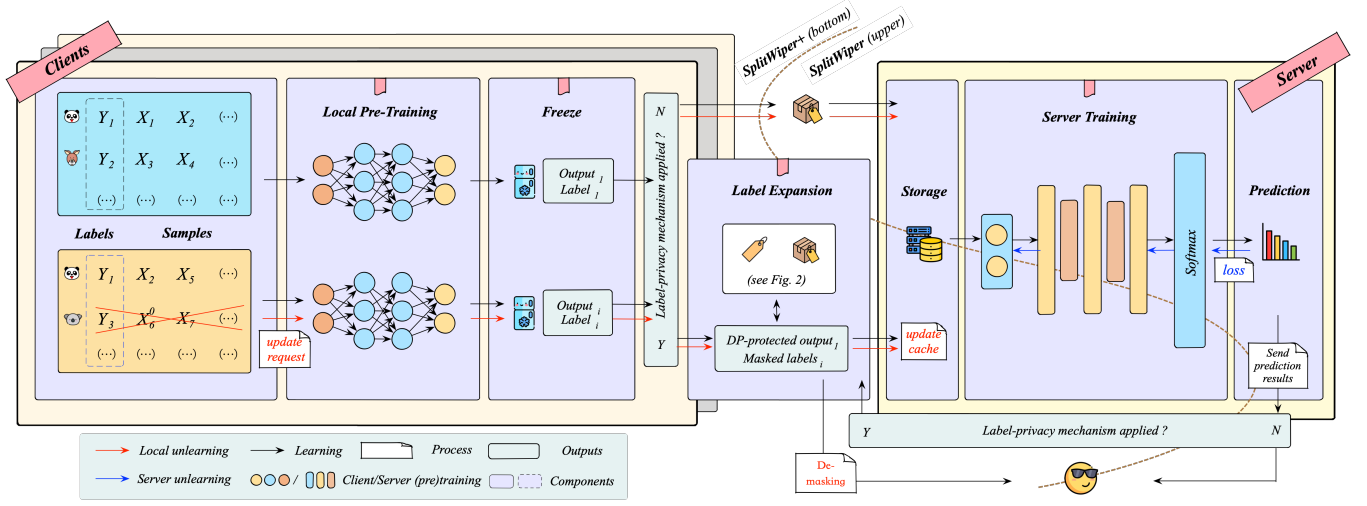


Figure 2: Architectural Design: In **SPLITWIPER** (Sec.4), each client k trains its local model \mathcal{F}_o^k on its dataset D_o^k , treating them as SISA shards without additional partitioning. After training, clients freeze the weights and send outputs with labels to the server for caching (one-off), while the server continues training \mathcal{F}_o^s using stored values without returning to the clients (one-way). **SPLITWIPER+** (Sec.5) enhances privacy by using a label expansion strategy to convert real labels into masked ones, which clients then share with the server for label-protected training.

one-way-one-off propagation ensures Isolated training by eliminating backward gradient propagation (one-way) and sending each client’s intermediate output to the server only once (one-off). This confines the impact of each shard to its corresponding client, with no information shared between different clients’ models. Within each client, data can be further divided into Slices by embedding SISA. On the server side, Aggregation combines stored intermediate outputs from clients, refining the server’s model and enabling efficient unlearning across multi-client SL.

4.2.1 Learning in SPLITWIPER. The **SPLITWIPER** framework (cf. Fig.2) consists of three learning phases: client model *pre-training*, client model *freezing and caching*, and server model *training*.

Client model pre-training. The concept of SISA is applied in this phase in a sense that the non-IID distributed datasets $\{D_o^1, D_o^2, \dots, D_o^k\}$ owned by any client k are inherently treated as the shards of SISA. By default, **SPLITWIPER** does not involve shard partitioning, as the server lacks access to any client’s dataset and only provides training services for outsourcing tasks. Consequently, each client k pre-trains its local model \mathcal{F}_o^k on its own dataset D_o^k independently and in parallel. This setup allows clients to use entirely different datasets and training tasks, as long as the dimensions of their outputs match the server’s input requirements to ensure smooth integration and communication.

Client model freezing and caching. The new one-way-one-off propagation is crucial by freezing clients’ weights and caching them on the server. After client model training, the outputs of the last layer (the cut or intermediate layer) and corresponding labels are shared with the server for caching, while weights are frozen to prevent further computation and communication during server training. In **SPLITWIPER**, mechanisms such as dropout layers or batch normalization are managed solely by the server, ensuring client model determinism. This allows for consistent sharing of

intermediate outputs, which the server can store without concerns about client model variability.

Server model training. As the central server that hosts the final output layer of the entire network, the server’s training process for its model, \mathcal{F}_o^s , functions similarly to the aggregation function $\Phi(\cdot)$ in SISA. By receiving and storing intermediate outputs from each client, the server accelerates the training of its model, which is typically much larger than the client models. With the client-side weights frozen, gradients are no longer back-propagated to the clients, restricting the incremental training process to the server. This ensures that subsequent updates do not propagate to all clients, implementing the one-way-one-off propagation scheme. This method lays a foundation for enabling unlearning with absent clients while significantly reducing overhead.

4.2.2 Unlearning in SPLITWIPER. Our **G1** ensures that only the clients requesting unlearning need to participate, while others remain uninvolved. We give below the workflow of SISA split unlearning in **SPLITWIPER**. The algorithm takes as input the number of epochs for client model updating N , the number of epochs for server model updating M , and the initiating client k who proposes the requirement for unlearning, and outputs the updated predicted label on the $\text{Concat}(\mathcal{F}_u^k, \mathcal{F}_u^s)$. It works in three steps as follows:

- **Step 1: Client weight unfreezing and dataset modification.** In this step, client k that initiates the retraining request unfreezes its weights and removes the intended unlearned samples from its dataset, $D_r^k = D_o^k \setminus \{D_u^k\}$. This ensures the local model is prepared for unlearning by excluding any influence from the unlearned samples.
- **Step 2: Client model unlearning and freezing.** The initiating client k re-trains its model for N epochs and then freezes the updated weights. Subsequently, it sends intermediate outputs and corresponding labels to the server for further training on the server. This step removes residual traces of the unlearned

samples from the local model of client k and ensures that the outputs provided to the server reflect the updated model state.

- **Step 3: Server model updating.** The server updates its storage with the intermediate outputs of client k , trains the server model for M epochs using the stored values, and then obtains the updated output distribution. This step integrates the updates of client k into the server model, ensuring consistent unlearning while maintaining utility for unaffected data.

Given that **G2** (effective unlearning with retained utility) is a fundamental requirement for any viable unlearning method, this approach also satisfies **G1** (independent unlearning). Clients freeze their weights and transfer intermediate values to the server, where they are securely stored. These values are selectively unfrozen only when necessary to perform the unlearning process, ensuring independent executions without involving or disrupting other clients.

5 SPLITWIPER+: Privacy-Preserving Variation via Label Expansion

The one-way-one-off propagation mechanism halts learning propagation at the server side and freezes the clients' parameters after their outputs are stored for the first time. While this design ensures efficiency and privacy for the shared data, further advancements in label privacy can be explored, particularly in scenarios where labels cannot be directly shared with the server, as seen in U-shaped SL, which is incompatible with the one-way-one-off propagation mechanism and therefore unable to support SISA split unlearning. So, this further introduces a key challenge if preserving labels private is a strict requirement:

*How to keep labels private while still enabling their sharing with the server for the unlearning process, without compromising the integrity of **G1** and **G2**?*

Player model. Two types of players are considered in this label-privacy-preserving scenario.

- **Internal users.** This includes clients (\mathbb{C}_{inter}) and servers (\mathbb{S}_{inter}) involved in training, modeled as *honest-but-curious* [21]. They adhere to the protocol without tampering, such as sending incorrect features or causing failures. Each client $C_{inter} \in \mathbb{C}_{inter}$ knows the shared labels among participants but keeps their complete label set \mathbb{Y}_{inter} private. Clients also aim to hide the quantity and semantics of their labels from the server \mathbb{S}_{inter} . This model emphasizes privacy preservation in internal label sharing.
- **External attackers.** An external attacker \mathbb{A}_{ext} is considered to know the distribution of the original dataset fed into the input layer at the targeted client C_{target} , and the posterior distribution of the model at the server side. The attacker \mathbb{A}_{ext} aims to use a shadow dataset that mimics these distributions to construct an attack model for conducting inference attacks [7]. This player model focuses on the level of privacy preservation in the unlearning capability of the framework.

Considering the player model and keeping labels private while achieving efficiency and effectiveness, we introduce an additional objective beyond **G1** and **G2** as follows.

- **G3: Privacy-preserving label sharing.** This goal emphasizes that the learning and unlearning processes should not disclose any label information from the client side to the server, including the quantity and semantics of the labels.

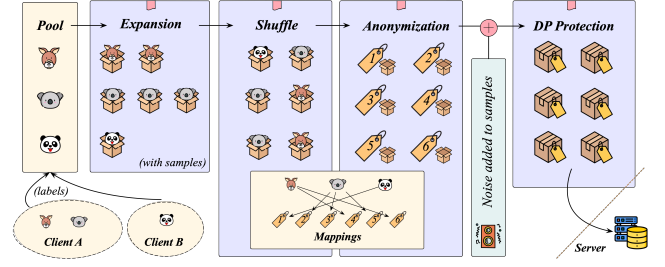


Figure 3: Label Expansion: Our proposed strategy preserves label quantity and semantics by expanding, shuffling, and anonymizing them, with intermediate values expanded via a DP mechanism.

Protecting label quantity matters. In **G3**, protecting the label privacy goes beyond protecting their semantics to include the quantity of labels, which is crucial for data privacy. By concealing the exact number of labels, adversaries are prevented from applying supervised learning methods that rely on knowing the number of classes for high accuracy. Instead, they are forced to use unsupervised learning techniques, lacking clear objectives and explicit class labels and generally resulting in lower accuracy [28, 32]. Protecting label quantity thus reduces the risk of unauthorized model reconstruction and inference attacks, strengthening the overall privacy and security of the learning process.

We introduce SPLITWIPER+ featuring a label expansion strategy combined with a DP-based masking scheme that protects the semantic information and quantity of labels shared with the server to achieve **G3**. This method is crucial when the server's trustworthiness is uncertain, as it conceals real labels, protecting client privacy and data security. By effectively masking both the quantity and specifics of the labels, clients can securely transmit masked labels to the server, mitigating privacy and security concerns.

Compared to conventional U-shaped SL tailored for label privacy, our new DP-based method in SPLITWIPER+ requires only one-way communication between clients and the server. This significantly reduces the communication frequency between clients and the server, enhancing efficiency and lowering operational costs. We show our label expansion-and-masking scheme in Fig.3.

5.1 Label Expansion Strategy in SPLITWIPER+

In SPLITWIPER+, the label expansion strategy plays a pivotal role in masking both the semantics and quantity of labels, ensuring compliance with **G3**. By incorporating secure label consensus building among all clients and implementing label expansion and anonymization, this strategy effectively obscures the true identity and count of shared labels. It provides the foundation for secure and private label sharing between clients and the server, addressing privacy concerns about label information leakage in SPLITWIPER.

We consider a scenario with Q real labels, represented as $\mathbb{Y} = Y_1, \dots, Y_Q$, where each client k holds a subset $\mathbb{Y}_k = Y_1, \dots, Y_{Q_k}$. These subsets collectively form the entire label set, such that

$$\bigcup_{k=1}^K \mathbb{Y}_k = \mathbb{Y}. \quad (1)$$

The corresponding intermediate values for label Y_q on client k are denoted as V_q^k . In Vanilla SL, where label privacy is not a concern, the data exchanged between client k and the server consists of pairs of labels and their corresponding intermediate values $\{Y_q, V_q^k\}$. To address the risks of label leakage, SPLITWIPER+ modifies this method by sharing masked labels and their DP-protected intermediate values $\{Y_q^*, V_q^{*k}\}$ instead.

To address the potential risk of sensitive label information being leaked during label sharing, we introduce a method that allows for label sharing without revealing the full set of labels \mathbb{Y}_k held by client k . Given the non-IID data distribution across clients, implementing a unified label expansion strategy in SPLITWIPER+ requires additional communication among clients. This strategy only requires clients to agree on all labels used in the SL task, without the need to exchange the actual data samples. Since internal users are assumed to be honest but curious, direct label sharing among clients could still expose sensitive information, making our method crucial for preserving privacy.

Secure label consensus building. The communication among clients facilitates consensus among all clients on the complete set of real labels \mathbb{Y} by allowing them to sequentially contribute their labels to a communal pool. The process is governed by two probabilistic thresholds:

- λ_1 : The probability that client k will select and add labels from its set \mathbb{Y}_k that are absent in the communal pool \mathbb{Y} .
- λ_2 : The probability that client k will continue to participate honestly in the sharing process after assessing whether \mathbb{Y}_k is already included in \mathbb{Y} .

These probabilistic controls over label sharing and continued participation ensure that adversaries cannot infer the complete label set \mathbb{Y}_k of any individual client from publicly shared information during the process. Even in scenarios where an untrustworthy client leaks information about the process, the adversary—potentially a curious server—remains unable to determine the specific labels associated with each client.

Label expansion and anonymization. Once the complete set \mathbb{Y} is aggregated, clients collaborate to assign an expansion factor γ_q to each label Y_q . This factor, γ_q , can be a randomly selected integer within a predefined range Γ_p for each Y_q , or be a constant integer $\gamma > 1$ and uniformly applied across all labels, i.e., $\gamma_q = \gamma, \forall q$. Each label, Y_q , is then expanded to γ_q instances, forming the initial expanded label set $\mathbb{Y}_{exp} = \{Y_1^*, \dots, Y_e^*, \dots, Y_E^*\}$, where $E = \sum_{q=1}^Q \gamma_q$. This expansion process masks the total count Q of real labels, preventing the server from inferring specific client tasks or data types based on label quantity. The elements within \mathbb{Y}_{exp} are shuffled and re-indexed to anonymize label semantics. While we use a basic pseudonymization technique, our framework can also accommodate methods like hashing [36] or encryption [12, 54] based on specific needs. The mapping between \mathbb{Y} and \mathbb{Y}_{exp} is preserved in \mathcal{G}_Y , ensuring that labels are expanded and anonymized without losing semantic integrity.

Since the information shared between client k and the server consists of pairs of labels and their corresponding intermediate values $\{Y_q, V_q^k\}$, when Y_q is expanded into γ_q masked labels, the corresponding V_q^k also need to be expanded to match the masked

labels. During this expansion, we introduce a DP mechanism to obfuscate V_q^k for further privacy protection.

DP-based expansion-and-masking scheme. Following the pre-determined label expansion map \mathcal{G}_Y , Client k takes the pairs $\{Y_q, V_q^k\}$ obtained from its local model \mathcal{F}_o^k as input. Each Y_q is then expanded into γ_q masked labels, and V_q^k is correspondingly expanded into γ_q DP-protected intermediate values, resulting in the set $\mathbb{U}_{exp} = \bigcup \{Y_e^*, V_e^{*k}\}$. Let the total collection of DP-protected intermediate values across all clients as $\mathbb{V}_{exp} = \{V_1^*, \dots, V_e^*, \dots, V_E^*\}$, keeping consistency with \mathbb{Y}_{exp} . The expansion of the intermediate values is carefully documented in \mathcal{G}_V , establishing a mapping that pairs expanded labels with their corresponding intermediate values, so we have $(\mathcal{G}_Y, \mathcal{G}_V)$ satisfying:

$$(\mathcal{G}_Y, \mathcal{G}_V) : \{Y_q, V_q\} \rightarrow \bigcup \{Y_e^*, V_e^*\}. \quad (2)$$

The mapping $(\mathcal{G}_Y, \mathcal{G}_V)$ represents a one-to-many expansion. Without collusion between clients and the server, $(\mathcal{G}_Y, \mathcal{G}_V)$ remains confidential, preventing the server from knowing the actual quantity or specific content of labels and intermediate values used in the training task.

5.2 Masked Label Sharing in SPLITWIPER+

Building on the proposed label expansion strategy, we examine how SPLITWIPER+ implements masked label sharing to enhance privacy, distinguishing it from the SPLITWIPER framework.

Masked label sharing. In SPLITWIPER+, after freezing the pre-trained model, clients employ an expansion-and-masking scheme before sharing their data with the server. In contrast, SPLITWIPER involves clients directly transmitting the obtained data $\{Y_q, V_q^k\}$ to the server after freezing the pre-trained model, without performing any additional masking step. The transformation of $\{Y_q, V_q^k\}$ into $\{Y_e^*, V_e^{*k}\}$ enables clients to transmit masked labels and their corresponding DP-protected intermediate values to the server, ensuring that the original information remains concealed.

Theorem 1 establishes that with the DP protection, the server cannot discern whether any two received labels are expanded from the same real label based on the corresponding intermediate values. In other words, the server cannot infer the quantity and content of real labels based on the received information, ensuring the privacy protection of the genuine labels used by the client.

Theorem 1. Consider that $\mathbb{V}_{exp} = \{V_1^*, \dots, V_E^*\}$ is expanded by $\mathbb{V} = \{V_1, \dots, V_Q\}$ using a given method \mathcal{G}_V , where $\forall V_q \in \mathbb{V}$, γ_q copies with ϵ -DP-based noise are generated as $\{V_{a_1}^*, \dots, V_{a_{\gamma_q}}^*\} \in \mathbb{V}_{exp}$. The probability that any two elements in \mathbb{V}_{exp} correspond to the same element in \mathbb{V} satisfies 2ϵ -DP.

Label demasking. The server in SPLITWIPER+ proceeds with the next stage of training based on the received masked labels and intermediate values. The final training output on the server should yield masked predicted labels, which does not hinder the server-side training since it utilizes the same masked labels for learning.

We introduce the label demasking on the client side when a client needs to utilize the complete model, i.e., $\text{Concat}(\mathcal{F}_o^k, \mathcal{F}_o^s)$. This step converts the predicted masked labels received from the server back into their original.

Given that \mathcal{G}_Y is a one-to-many label expansion method, its inverse, \mathcal{G}_Y^{-1} , operates as a many-to-one function, defined as:

$$Y_q = \mathcal{G}_Y^{-1}(Y_e^*). \quad (3)$$

Each masked label corresponds to a real label, and since the mapping \mathcal{G}_Y is shared among all clients, they can easily demask the server's predicted results using the inverse mapping \mathcal{G}_Y^{-1} . This demasking process ensures the integrity and applicability of the model's predictions. The demasked labels are then used to compute the performance $\mathcal{T}(\text{Concat}(\mathcal{F}_o^k, \mathcal{F}_s^s), D_o^k)$, ensuring evaluation is based on accurate and relevant data, reflective of true model performance in real-world settings.

Label protection in unlearning process. When the initiating client k requests unlearning, only client k participates in the subsequent update process. Other clients do not receive any additional updated information. The server, upon receiving the updated masked labels and corresponding DP-protected intermediate values shared by client k , gains insight into the aspects of the unlearning process. However, since the labels are masked, the server cannot accurately infer the quantity or category of the real labels involved in the unlearning from the changes in the number of masked labels. Thus, we deem that the privacy associated with the unlearning labels remains protected, effectively safeguarding sensitive information from inadvertent disclosure.

Natural barrier against shadow inference. Shadow inference attacks involve an adversary using a shadow dataset to mimic the training and unlearning processes of the target model, ultimately allowing them to infer which data points have been unlearned, thereby compromising the privacy of the dataset [7]. The proposed label expansion strategy addresses this vulnerability and strengthens client privacy by implementing two key methods:

- **Training on private datasets.** When training on private datasets, an external attacker with partial access to a victim's data cannot construct a complete shadow dataset. While [7] assumes the attacker knows all labels in the victim's dataset, this is unrealistic unless using a well-known public dataset. Balancing positive and negative cases and diversifying shadow models is crucial for generalization in the attack model. Without full label knowledge, inferring the complete input-output mapping is challenging, restricting the attacker's ability to replicate the training environment and conduct effective inference.
- **Training on various tasks.** When clients are engaged in different tasks, such as CIFAR-10/MNIST, the attacker's challenge increases, as they cannot easily reconstruct a shadow dataset that accurately reflects the diverse tasks of the clients. This further diminishes the attacker's ability to predict the model's behavior and the impact of unlearning.

For simplicity in our experiments (cf. Sec. 7.1), we use public datasets to mimic this process and intentionally ignore the potential risk of inference posed by using public datasets.

SPLITWIPER as a specific case of SPLITWIPER+. The label expansion strategy provides inactive privacy protection when expansion factor $\gamma_q = \gamma = 1, \forall q$, in which case SPLITWIPER+ becomes SPLITWIPER with no DP applied. For ease of comparison, throughout this paper, we will refer to SPLITWIPER as the case where $\gamma = 1$ and SPLITWIPER+ as the case where $\gamma > 1$ is applied. These two

will be differentiated whenever label privacy is relevant in the experiments; otherwise, SPLITWIPER will be used by default.

Compared to SPLITWIPER, label expansion in SPLITWIPER+ introduces additional computational overhead and increases the size of the data transmitted from clients to the server, thereby increasing communication costs. Thanks to the one-way-one-off propagation scheme, these additional overheads are incurred only once, preserving the significant advantages over existing SL methods, particularly in client-side efficiency. A detailed analysis of client-side and server-side overhead complexities are shown in Sec. 6, where Table 2 presents a comparison of the client-side overhead among SPLITWIPER, SPLITWIPER+, and existing Vanilla and U-shaped SL frameworks.

6 Complexity Analysis

To further demonstrate the advantages of the SPLITWIPER and SPLITWIPER+ frameworks in reducing computational and communication overhead, we analyze its complexity in both learning and unlearning scenarios. We compare these metrics with traditional Vanilla and U-shaped SL, which adhere to the standard procedures of the usual round-robin SL processes detailed in [14, 33], as shown in Table 2.

6.1 Learning Scenarios

6.1.1 Computational Complexity. Client-side computational complexity. In SPLITWIPER framework, the primary computational cost is derived from the pre-training of local models on each client k , which is $O(N \cdot \mathcal{H}_K)$, where N is the epochs for client model pre-training and \mathcal{H}_K is the complexity of a single training iteration. While in SPLITWIPER+, beyond the pre-training computational cost, there is an additional component that arises from the computations required by the label expansion strategy implemented after the pre-training phase on the client side. This additional cost is $O(\gamma \mathcal{H}_{exp})$, where \mathcal{H}_{exp} represents the complexity of the label expansion-and-masking scheme, and γ is the average of the label expansion factors.

However, since the pre-training of client models is conducted independently of the server, it is excluded from our computational complexity analysis for the SL process. Instead, we focus on the final epoch of the pre-training process, disregarding the impact of N . Due to the one-way-one-off propagation design, clients freeze their local models after pre-training and do not participate in the subsequent M epochs of server training. Therefore, the computational complexity of the proposed frameworks is also unaffected by M . For a total of K clients, the computational complexity across all clients in SPLITWIPER is quantified as $O(K \cdot \mathcal{H}_K)$. Correspondingly, the computational complexity of SPLITWIPER+ is $O(K \cdot (\mathcal{H}_K + \gamma \mathcal{H}_{exp}))$.

In contrast, traditional SL, such as Vanilla and U-shaped SL, requires all K clients to be involved across the entire learning process for M epochs, leading to higher computational complexity. Specifically, the computational complexity for the Vanilla SL framework is $O(M \cdot K \cdot \mathcal{H}_K)$. As for the U-shaped framework, which allocates the output layer to the client side, an additional computation cost is required. Consequently, the total complexity for the U-shaped framework is $O(M \cdot K \cdot (\mathcal{H}_K + \mathcal{H}_{out}))$, with \mathcal{H}_{out} representing the complexity associated with the output layer operations.

Table 2: Client-side Privacy and Overhead in SL

	Framework	Label Privacy	Computational	Communication
Learning	Vanilla SL	Low	$O(M \cdot K \cdot \mathcal{H}_K)$	$O(M \cdot K \cdot (I_K + I_Y))$
	U-shaped SL	High	$O(M \cdot K \cdot (\mathcal{H}_K + \mathcal{H}_{out}))$	$O(M \cdot K \cdot I_K)$
	SPLITWIPER	Low	$O(K \cdot \mathcal{H}_K)$	$O(K \cdot (I_K + I_Y))$
	SPLITWIPER+	Medium	$O(K \cdot (\mathcal{H}_K + \gamma \mathcal{H}_{exp}))$	$O(K \cdot \gamma \cdot (I_K + I_Y))$
Unlearning	Vanilla SL	Low	$O(M \cdot K \cdot \mathcal{H}_K)$	$O(M \cdot K \cdot (I_K + I_Y))$
	U-shaped SL	High	$O(M \cdot K \cdot (\mathcal{H}_K + \mathcal{H}_{out}))$	$O(M \cdot K \cdot I_K)$
	SPLITWIPER	Low	$O(\mathcal{H}_K)$	$O(I_K + I_Y)$
	SPLITWIPER+	Medium	$O(\mathcal{H}_K + \gamma \mathcal{H}_{exp})$	$O(\gamma \cdot (I_K + I_Y))$

High: no label sharing; Low: real label sharing; Medium: masked label sharing; \mathcal{H}_K : client-side training complexity; \mathcal{H}_{exp} : label expansion-and-masking scheme complexity; \mathcal{H}_{out} : output layer complexity; I_K : size of intermediate outputs; I_Y : size of labels; \mathcal{H}_{out} : output layer complexity.

The complexity of the expansion-and-masking scheme \mathcal{H}_{exp} scales linearly with the number of real labels and the size of intermediate values. This complexity remains within the same order of magnitude as \mathcal{H}_K , especially given the significant size of large models today. This computational load drives clients to use SL in collaboration with a server rather than independently handling the entire learning process. For security and efficiency, the expansion factor γ is typically kept small, and training epochs M are much larger, often ranging from several dozen to hundreds. As a result, the computational overhead of the SPLITWIPER+ framework is substantially lower than that of traditional Vanilla and U-shaped SL frameworks. Moreover, SPLITWIPER has only $1/M$ of the computational complexity of Vanilla SL, further highlighting its advantages.

6.1.2 Communication Overhead. Client-side communication complexity. In the SPLITWIPER and the SPLITWIPER+ frameworks, the communication overhead is determined by transmitting the (masked) labels and corresponding intermediate outputs from each client to the server in a one-way propagation manner. This leads to a complexity of $O(K \cdot (I_K + I_Y))$ for SPLITWIPER and $O(K \cdot \gamma \cdot (I_K + I_Y))$ for SPLITWIPER+, where I_K is the size of intermediate outputs and I_Y is the size of labels on client-side.

Compared to Vanilla SL, which incurs a per-epoch communication complexity of $O(K \cdot (I_K + I_Y))$ and U-shaped SL that does not require label transmission, our label expansion strategy in SPLITWIPER+ does introduce additional communication overhead. However, considering that traditional SL frameworks involve continuous communication between clients and the server across M epochs, the overall communication complexities are respectively $O(M \cdot K \cdot (I_K + I_Y))$ and $O(M \cdot K \cdot I_K)$. Given that M is significantly greater than γ , the SPLITWIPER+ framework still maintains a substantial advantage regarding communication efficiency.

Server-side communication complexity. In the proposed frameworks, the information transfer from clients to the server follows a one-way propagation manner, allowing us to consider that the server incurs no communication costs. In contrast, the Vanilla SL framework requires the server to send the loss back to each client to facilitate model training, leading to a communication complexity of $O(M \cdot I_{loss})$ over M epochs, where I_{loss} represents the size of the loss during each epoch. Similarly, our U-shaped SL necessitates

the server to return outputs back to clients for operations related to the final output layer, resulting in a communication complexity of $O(M \cdot I_S)$, where I_S is the size of the server-side outputs.

6.1.3 Server-side Storage Overhead. In addition to computational and communication overheads, our SPLITWIPER and SPLITWIPER+ frameworks introduce additional storage overhead on the server to support the one-way-one-off propagation scheme. The server needs to store the intermediate values received from clients, ensuring that these values can be readily accessed during unlearning scenarios without disrupting other clients which do no request unlearning, whereas in Vanilla SL and U-shaped SL, server-side storage overhead depends solely on \mathcal{W}_s , the size of the stored server-side model weights.

In SPLITWIPER, the storage complexity on the server is $O(K \cdot \mathcal{D}_{inter} + \mathcal{W}_s)$, where K is the number of clients and \mathcal{D}_{inter} is the average size of the intermediate values transmitted by each client. In SPLITWIPER+, the intermediate values are expanded, resulting in a storage complexity of $O(K \cdot \gamma \cdot \mathcal{D}_{inter} + \mathcal{W}_s)$ with the average of the label expansion factors γ .

6.2 Unlearning Scenarios

In the unlearning scenario, the retraining process in traditional Vanilla and U-shaped SL frameworks necessitates the participation of all clients. This implies that each client must re-initiate model updates and transmit intermediate values for an additional M epochs, replicating the computational and communication costs observed in the learning scenario.

Conversely, SPLITWIPER and SPLITWIPER+ introduce an optimized approach, wherein unlearning requests are isolated to the specific client initiating the request, significantly reducing the overall burden. This isolation avoids the need for unrelated clients to engage in redundant computations and communications, thereby enhancing efficiency and reducing client-side total resource consumption. In terms of client-side computational complexity in SPLITWIPER framework, the primary costs come from the retraining of the local model on the initiating client k , which is $O(\mathcal{H}_K)$. In contrast, in SPLITWIPER+, client k also incurs the additional cost of performing a new round of the expansion-and-masking scheme, resulting in a computational complexity of $O(\mathcal{H}_K + \gamma \mathcal{H}_{exp})$.

The communication overhead is determined by transmitting the intermediate outputs from the initiating client k to the server, leading to a complexity of $O(\gamma \cdot (I_K + I_Y))$ in the proposed frameworks, where $\gamma = 1$ represents SPLITWIPER, and $\gamma > 1$ represents SPLITWIPER+. Additionally, in unlearning scenarios, the server only updates the stored intermediate values related to client k without requiring additional storage costs, allowing us to consider server-side storage overhead during the unlearning scenarios consistent with that of the learning scenarios. It satisfies **G1** (independent unlearning), thus improving the robustness and reducing the overhead of unlearning processes.

7 Evaluation

In this section, we empirically evaluate SPLITWIPER of SPLITWIPER+, examining its compliance with key objectives: **G1** (independent unlearning), **G2** (effective unlearning with retained utility), and **G3** (privacy-preserving label sharing).

7.1 Experimental Settings

Our experiments run on NVIDIA PCIe A100, $2 \times 40\text{GB}$. We use Python 3.7.16 and PyTorch 2.3.1 to build and train our framework.

We evaluate the effectiveness, efficiency, and privacy level of SPLITWIPER and SPLITWIPER+ with $K = 5$ clients and a single server, where client data is distributed in a non-IID manner.

After completing their learning tasks, a subset of clients, denoted as \mathbb{C}_u , initiates unlearning requests. In this evaluation, we set $|\mathbb{C}_u| = 1$, where $|\cdot|$ represents the ℓ_1 norm, indicating that only one client requests unlearning. The remaining clients, who do not initiate unlearning requests, are denoted as \mathbb{C}_o .

SPLITWIPER and SPLITWIPER+ are versatile, accommodating a variety of unlearning scenarios including class-, client-, and sample-level tasks. The experiments specifically focus on class-level unlearning tasks for clarity of presentation [9]. In our settings, \mathbb{C}_u selects a subset of label, denoted as \mathbb{Y}_u for unlearning, aiming to effectively erase all related information without affecting other clients \mathbb{C}_o , or other labels \mathbb{Y}_o . We set $|\mathbb{Y}_u| = 1$, i.e., only a specific label is required for unlearning. The unlearned label set, \mathbb{Y}_u , is exclusively possessed by \mathbb{C}_u , aligned with the non-IID data distribution setting and enabling a clearer demonstration of unlearning effectiveness.

Dataset and data distribution. The experiments are conducted under various settings based on the dataset used and the degree of non-IID distribution among the clients:

- **Clients with same task.** Clients operate on the same dataset but under different conditions:
 - *Homogeneous labels but heterogeneous volumes.* All clients possess every label except Y_u , but the volume of labels varies among clients.
 - *Partially overlapping labels.* Clients have different types and quantities of labels with some overlap, i.e., all labels except Y_u are randomly distributed among 1-3 clients.
 - *Heterogeneous labels.* Each client has a unique set of labels, with no overlap.
- **Clients with different tasks.** Clients utilize different datasets for separate tasks, with distinct labels among the clients.

In each setting, the training and testing datasets are distributed among the clients using the same method to ensure consistency between the clients' training and testing data.

For experiments where clients perform the same task, we use the CIFAR-10 [18, 30] and CIFAR-100 [16, 41] datasets, both widely utilized in the fields of computer vision and neural networks. CIFAR-10 consists of 60,000 32×32 color images distributed across 10 classes, while CIFAR-100 includes 100 classes with 600 images per class. In the CIFAR-10 experiments, the designated label Y_u is set to "bird," and in CIFAR-100, it is set to "aquarium-fish." For experiments where clients perform different tasks, 3 clients utilize the CIFAR-10 dataset, while 2 clients use the MNIST dataset [11], which comprises 60,000 28×28 grayscale images of handwritten digits. In this setup, the label "bird" from CIFAR-10 remains designated as Y_u .

Model architecture. We utilize the architecture based on a simplified VGG [1] and a ResNet18 [17] to construct the client-side and server-side models for our split (un)learning experiments. In this configuration, the first convolutional layer of VGG or ResNet18

serves as the client model \mathcal{F}_o^k , while the remaining other layers comprise the server model \mathcal{F}_o^s . Given that VGG and ResNet have negligible differences in their first convolutional layer, we let the number of parameters in the client model \mathcal{F}_o^k be consistent across both architectures. The number of epochs for client-side pre-training is set to $N = 10$. The number of server-side model training and retraining epochs is set to $M = 30$ for experiments involving CIFAR-10 and $M = 50$ for experiments utilizing CIFAR-100.

In scenarios where clients train on the same task, each client's model, $\mathcal{F}_o^k, \forall k$, adheres to a uniform structure with the same input and output sizes across all clients. Conversely, in scenarios with clients engaged in different tasks, the input size of each \mathcal{F}_o^k is tailored based on the specific training dataset D_o^k used locally. However, the output size remains consistent to meet the input requirements of the server model \mathcal{F}_o^s . This design ensures that despite the diverse data types and tasks, all client models can interface correctly with the centralized server model within SL.

Benchmark and experimental groups. We use non-SISA unlearning on Vanilla and U-shaped SL [14] as benchmarks to evaluate SPLITWIPER. To ensure fairness, both benchmarks are initialized with pre-trained results from SPLITWIPER. The client and server components in these frameworks are trained for the same M epochs as SPLITWIPER during learning and retraining for unlearning requests. By standardizing initial conditions and training durations, we systematically compare the efficiency and effectiveness of SPLITWIPER against the benchmarks.

To assess the impact of various label expansion strategies in SPLITWIPER and SPLITWIPER+, we categorize experiments based on different expansion factors. Specifically, we define four experimental groups: three groups with uniform label expansions of $\gamma = 1, 2$, or 3 , and one group with a variable expansion factor, $\gamma = \Gamma_p$. For $\gamma = 1$ (SPLITWIPER), labels are shuffled and re-indexed without altering their actual count, similar to the Vanilla SL approach where real labels are shared. For $\gamma = 2$ or $\gamma = 3$ (SPLITWIPER+), clients expand labels to γ instances and apply differential privacy (ϵ -DP) with $\epsilon = 0.1$ or 0.2 to protect intermediate values.

In the variable scenario ($\gamma = \Gamma_p$), the expansion factor γ_q for each label Y_q is randomly selected from a predefined range, $\Gamma_p = [1, 3]$. This configuration enables us to examine the effects of variability in the expansion factor on both label privacy and system performance.

Performance metrics. **G1** represents a significant advancement by enabling absent clients and involving only those requesting unlearning, laying the groundwork for SISA in SL. While not easily assessed quantitatively, **G1** serves as the foundational experiment setting and premise for achieving **G2** and **G3**. We analyze the unlearning effectiveness and efficiency of existing SL frameworks, SPLITWIPER, and SPLITWIPER+. We measure label accuracy during learning and unlearning phases for effectiveness, and track training and transmission times for efficiency. We focus on client-side overhead and use a 100 Mbps bandwidth setting to simulate realistic conditions. This approach demonstrates the benefits of the one-way-one-off propagation.

To further evaluate **G3**, we assess the robustness of our privacy-preserving method. We examine the server's ability to infer real labels from masked labels and intermediate outputs. In SPLITWIPER+, the server uses clustering algorithms on received data pairs to group

Table 3: Evaluation on Efficiency and Effectiveness over VGG across Different SL Frameworks

				(Client-side) Training / Transmission Time (s)							Accuracy (%)											
				Existing SL		SPLITWIPER		SPLITWIPER+			Existing SL		SPLITWIPER		SPLITWIPER+							
				U-shaped	Vanilla	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = \Gamma_p$	U-shaped	Vanilla	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = \Gamma_p$							
Clients with same task	Homo labels Hetero volumes	CIFAR-10	L.	C_u	74.36/216.40	66.03/261.63	2.23/9.04	9.73/18.08	15.85/27.13	9.08/18.08	Y_u	95.40	92.70	99.80	98.10	98.70	97.40	97.90	99.40	99.70		
				C_o	58.88/172.98	42.65/168.02	1.42/5.90	6.16/11.80	10.25/17.70	5.72/5.90	Y_o	93.47	92.36	97.04	96.67	97.41	97.07	96.16	96.83	97.97		
				C_u	34.23/168.63	22.18/168.72	1.43/5.84	5.94/11.68	10.20/17.52	5.64/5.84	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
		CIFAR-100	U.	C_o	57.48/172.95	42.89/168.42	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	94.23	92.94	97.22	96.47	97.91	96.78	97.07	97.14	98.56		
				C_u	79.88/324.58	74.68/328.02	1.44/8.24	6.31/16.48	10.70/24.73	5.88/24.72	Y_u	91.20	92.30	97.60	97.00	97.10	97.00	98.10	97.40	97.80		
				C_o	77.21/312.58	71.84/315.24	1.41/6.22	5.98/12.44	10.21/18.66	5.48/6.22	Y_o	82.01	81.77	85.39	83.75	84.17	82.53	83.44	85.27	86.08		
	Partially overlapping labels	CIFAR-100	L.	C_u	77.03/308.95	70.93/312.04	1.36/6.13	5.92/12.25	10.17/18.38	5.43/6.13	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
				C_o	77.15/309.45	71.56/312.04	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	82.79	82.02	86.86	83.51	84.95	82.71	84.23	86.19	87.17		
			U.	C_u	87.58/313.31	76.24/314.44	2.47/10.72	11.07/21.44	18.93/32.17	9.84/21.44	Y_u	95.60	94.30	99.60	99.50	98.40	99.90	99.20	99.60	99.70		
				C_o	42.99/162.61	38.59/157.22	1.28/5.34	5.53/10.68	9.43/16.02	5.47/10.42	Y_o	94.03	92.91	97.30	96.34	97.26	95.80	96.94	96.79	97.78		
		CIFAR-100	L.	C_u	67.88/219.54	53.51/220.83	1.74/7.52	7.72/15.04	13.31/22.56	6.41/15.04	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
				C_o	43.49/163.04	38.25/157.22	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	95.08	92.35	97.72	97.61	97.19	96.11	96.64	97.81	98.07		
			U.	C_u	115.03/513.69	113.55/516.06	2.26/12.72	9.85/25.44	17.01/38.17	8.62/28.17	Y_u	91.90	91.50	98.00	97.70	98.00	97.60	97.60	98.00	98.10		
				C_o	68.54/285.63	60.44/287.24	1.08/5.28	4.43/10.56	7.56/15.84	4.19/14.32	Y_o	81.98	81.73	83.94	82.47	83.47	82.29	83.42	83.67	84.03		
	Hetero labels	CIFAR-100	L.	C_u	112.48/498.07	110.02/500.07	2.16/12.40	9.53/24.80	16.44/37.21	8.04/14.40	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
				C_o	68.82/282.50	62.86/284.04	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	83.01	82.11	85.14	83.95	84.51	83.66	84.37	84.86	85.17		
			U.	C_u	49.04/187.54	43.87/189.62	1.44/6.40	6.25/12.80	10.78/19.20	8.87/12.80	Y_u	93.20	93.90	99.40	99.80	99.80	99.70	99.50	99.90	99.90		
				C_o	50.31/187.53	44.34/189.63	1.46/6.40	6.28/12.80	10.78/19.21	7.37/12.82	Y_o	93.04	93.67	98.52	97.15	97.92	95.74	96.92	98.26	98.10		
		CIFAR-100	L.	C_u	27.80/93.77	22.18/96.01	0.73/3.20	3.12/6.41	5.39/9.57	5.57/9.60	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
				C_o	48.88/187.54	43.97/189.64	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	93.93	94.13	98.60	96.98	98.53	97.06	97.61	98.59	98.14		
			U.	C_u	79.51/312.58	76.99/316.04	1.48/6.40	6.33/12.83	10.79/19.22	6.36/19.16	Y_u	90.30	92.10	99.90	99.70	98.90	98.70	99.00	99.00	99.10		
				C_o	79.74/312.60	77.02/316.03	1.50/6.42	6.35/12.80	10.83/19.22	5.78/11.21	Y_o	81.42	80.22	87.43	85.52	86.93	84.14	85.77	86.07	87.71		
	different tasks	CIFAR-10 + MNIST	L.	C_u	78.88/296.96	76.66/300.04	1.44/6.08	6.00/12.16	10.26/18.24	5.79/12.16	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
				C_o	79.32/312.57	77.11/316.04	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	82.04	80.95	88.87	86.89	87.93	85.99	87.06	87.59	88.21		
			U.	C_u	93.61/381.11	88.92/384.05	2.95/12.80	13.50/25.61	31.64/37.29	13.08/24.76	Y_u	95.80	95.60	99.90	97.20	98.60	96.80	96.80	97.30	97.40		
				C_o	104.97/428.04	90.37/438.09	3.01/14.26	14.55/30.13	23.64/37.81	13.41/28.98	Y_o	95.13	95.73	98.64	96.71	97.15	95.09	96.47	96.31	96.71		
			U.	C_u	80.97/286.96	61.15/288.04	2.39/9.60	9.89/19.20	14.20/26.13	9.92/19.08	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
				C_o	103.79/428.09	90.35/437.94	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	95.12	95.74	99.07	96.89	97.01	95.33	96.34	96.56	96.72		
①	②	③	④							⑤							$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.1$	$\epsilon = 0.2$

Notation: ① Data distribution; ② Dataset; ③ Scenario (L. for learning, U. for unlearning); ④ Client type (C_u for clients involved in unlearning, C_o for clients excluded from unlearning); ⑤ Label type (Y_u for to-be-unlearned labels, Y_o for retained labels).

similar labels. The metric is the probability of correctly matching masked labels to real ones.

7.2 Unlearning Efficiency and Overhead

The left sides of Table 3 and Table 4 show the training and transmission times for clients in various SL frameworks over VGG and ResNet18, focusing on clients with unlearning requests (C_u) and other clients (C_o) across both learning and unlearning scenarios. Since the degree of DP-based noise addition does not affect time costs, we exclude the influence of ϵ from our analysis, focusing solely on the effects of different label expansion factors γ .

The results highlight the efficacy of SPLITWIPER in achieving **G1** (independent unlearning). Training and transmission times in SPLITWIPER are significantly lower than those in Vanilla and U-shaped SL across all data distributions and scenarios. Crucially, during unlearning, only the requesting clients, C_u , incur additional costs, while other clients, C_o , remain unaffected, as the server uses stored data from the learning phase to handle unlearning, minimizing overall client-side expenses and disruptions. The time overhead for the label expansion strategy discussion between clients is negligible, measured in milliseconds, and does not significantly impact our analysis of training and transmission times.

In SPLITWIPER and SPLITWIPER+, clients only need to train the pre-trained model over local data once. They then mask labels

according to the label expansion strategy and apply DP protection to the corresponding intermediate values before transmitting them to the server in a one-way propagation manner. Subsequent training phases focus exclusively on the server, disconnecting from client-side computations. Therefore, the training and transmission times for clients are fixed values dependent on γ and remain unaffected by the number of training epochs M . In contrast, in Vanilla and U-shaped SL, where each epoch involves client participation, the computational and transmission times increase linearly with M , as listed in Table 2 (cf. Sec. 6).

Fig. 4 shows the average training and transmission times for all clients across different SL frameworks as training epochs M increase. At $M = 1$, the times for Vanilla and U-shaped SL are nearly identical to those of SPLITWIPER with $\gamma = 1$. As γ increases (SPLITWIPER+), computational and communication costs rise, but they typically stay below 5 due to client-specified balancing. Even in practical settings with $M = 30$ for CIFAR-10 and CIFAR-10/MNIST, and $M = 50$ for CIFAR-100, SPLITWIPER retains a clear advantage, especially during unlearning, where only the requesting clients (C_u) incur costs, while others (C_o) do not.

To illustrate the impact of the label expansion factor γ on client-side overhead in SPLITWIPER+, we plot the average training and transmission times for all clients across various γ values in Fig.5(a) and Fig.5(b). Both figures, for VGG and ResNet18, show a roughly

Table 4: Evaluation on Efficiency and Effectiveness over ResNet18 across Different SL Frameworks

				(Client-side) Training / Transmission Time (s)								Accuracy (%)									
				Existing SL		SPLITWIPER		SPLITWIPER+				Existing SL		SPLITWIPER		SPLITWIPER+					
				U-shaped	Vanilla	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = \Gamma_p$	U-shaped	Vanilla	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = \Gamma_p$						
Clients with same task	Homo labels Hetero volumes	CIFAR-10	L.	C_u	67.12/241.35	63.09/245.70	2.07/8.55	9.53/16.80	12.77/25.50	8.83/24.75	Y_u	90.90	88.60	89.70	89.20	90.30	88.30	90.40	89.90	91.20	
			C_o	42.06/159.30	38.69/177.45	1.25/5.55	6.29/11.10	8.15/16.65	5.51/9.60	Y_o	76.47	77.99	80.13	80.87	81.11	80.69	80.86	80.61	81.09		
			U.	C_u	44.95/149.55	40.27/157.80	1.32/5.40	6.21/10.95	8.29/16.50	6.51/11.55	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
		CIFAR-100	L.	C_u	41.76/162.45	39.08/169.95	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_u	77.29	78.64	80.95	81.46	82.37	81.07	82.08	82.59	83.86	
			C_o	99.75/308.84	96.53/319.96	1.59/8.03	6.46/16.07	9.23/24.10	6.43/24.10	Y_u	89.30	88.90	92.25	91.70	93.20	92.00	93.10	93.00	94.60		
			U.	C_u	79.82/300.96	77.22/307.48	1.53/6.08	6.03/12.13	8.80/18.21	6.05/6.08	Y_o	66.88	66.92	70.62	69.14	70.33	67.43	68.69	70.31	71.48	
	Partially overlapping labels	CIFAR-10	L.	C_u	82.28/301.43	79.30/304.86	1.54/5.97	6.05/11.97	8.69/17.94	5.94/9.87	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
			C_o	79.75/300.46	78.04/304.32	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	66.52	67.35	71.71	70.79	71.01	67.29	69.12	71.06	72.16		
			U.	C_u	79.75/300.46	78.04/304.32	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_u	66.52	67.35	71.71	70.79	71.01	67.29	69.12	71.06	72.16	
		CIFAR-100	L.	C_u	79.72/268.65	75.56/295.35	2.43/10.05	11.66/20.25	17.80/30.30	13.69/28.05	Y_u	85.80	86.60	88.60	88.20	91.30	88.00	88.80	88.40	89.80	
			C_o	60.00/165.30	56.91/190.35	1.22/4.95	5.72/10.05	10.10/15.00	5.45/8.71	Y_o	77.15	78.56	80.86	80.44	80.84	80.26	80.53	80.61	81.62		
			U.	C_u	55.97/178.20	53.18/207.45	1.73/7.05	8.02/14.10	12.50/21.15	8.75/18.60	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	Hetero labels	CIFAR-10	L.	C_u	60.34/169.80	52.24/198.45	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	80.02	80.67	81.69	81.49	82.51	81.24	82.11	81.81	82.45	
			C_o	134.15/497.29	129.23/499.94	2.62/12.40	11.18/24.80	17.18/37.25	10.80/36.89	Y_u	85.30	84.40	89.30	91.10	90.70	87.90	89.10	90.55	91.40		
			U.	C_u	93.79/332.71	89.93/347.02	1.12/5.15	4.92/10.30	7.34/15.44	5.12/13.96	Y_o	63.88	63.03	69.25	68.55	69.09	68.02	68.86	69.29	70.29	
		CIFAR-100	L.	C_u	126.87/452.21	120.65/459.62	2.42/10.96	10.16/23.52	15.58/35.41	10.16/14.98	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
			C_o	95.28/336.73	89.05/346.28	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	64.52	63.81	70.46	69.01	69.72	68.58	68.73	70.33	71.03		
			U.	C_u	95.28/336.73	89.05/346.28	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_u	64.52	63.81	70.46	69.01	69.72	68.58	68.73	70.33	71.03	
	different tasks	CIFAR-10 + MNIST	CIFAR-10	L.	C_u	46.88/176.40	43.52/208.05	1.46/6.15	7.19/12.30	9.31/18.15	7.89/15.90	Y_u	88.60	86.60	91.80	91.60	92.40	91.30	92.10	91.60	93.70
				C_o	47.05/215.55	44.43/222.30	2.03/6.00	7.16/12.15	9.25/18.15	6.00/10.65	Y_o	85.96	85.66	86.23	87.38	88.53	86.92	87.66	88.02	88.37	
				U.	C_u	24.96/88.20	22.63/104.10	0.72/3.00	3.63/6.00	4.61/9.00	4.03/7.80	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			CIFAR-100	L.	C_u	47.11/205.95	44.40/215.25	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	86.42	87.22	87.92	87.94	88.72	87.12	88.14	88.17	89.04
				C_o	83.54/312.00	79.94/321.64	1.46/6.40	5.72/12.80	9.29/19.20	6.67/19.20	Y_u	89.20	89.10	91.60	91.10	92.80	91.20	91.80	91.60	92.70	
				U.	C_u	99.37/386.16	95.21/395.20	1.53/6.40	6.37/12.80	9.27/19.20	6.31/11.28	Y_o	69.61	70.68	73.76	71.96	72.56	70.99	71.09	72.60	72.93
different tasks		CIFAR-10	L.	C_u	74.51/295.36	71.32/300.12	1.41/6.08	6.00/12.16	8.89/18.24	6.10/12.16	Y_u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
			C_o	98.15/386.20	94.85/393.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	69.99	70.57	74.72	72.27	73.29	71.98	72.38	72.59	73.33		
			U.	C_u	98.15/386.20	94.85/393.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	Y_o	69.99	70.57	74.72	72.27	73.29	71.98	72.38	72.59	73.33	
		CIFAR-100	L.	C_u	99.94/345.75	95.03/353.85	3.17/12.00	15.86/24.15	28.47/36.15	17.46/21.45	Y_u	91.30	90.40	94.10	92.50	92.60	91.60	92.30	93.00	93.10	
			C_o	108.48/391.80	100.52/397.95	3.13/13.35	16.53/26.85	21.69/40.20	17.75/28.95	Y_o	89.41	88.75	93.77	93.46	93.75	92.76	92.98	93.22	93.49		
			U.	C_u	76.59/257.40	73.01/266.10	2.36/9.15	11.86/18.00	15.02/27.00	11.96/12.30	Y_o	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

①

②

③

④

⑤

$\epsilon = 0.1$

$\epsilon = 0.2$

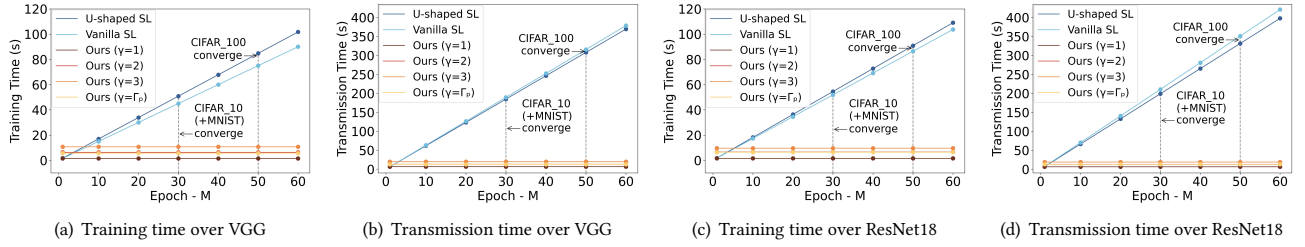
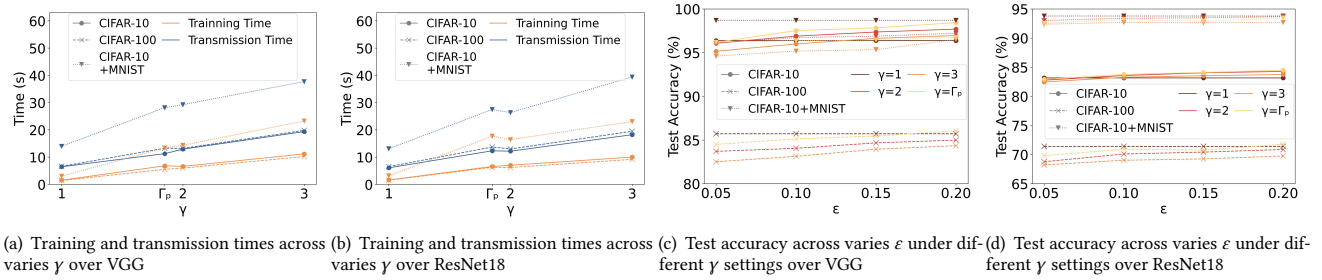
$\epsilon = 0.1$

$\epsilon = 0.2$

$\epsilon = 0.1$

$\epsilon = 0.2$

Notation: ① Data distribution; ② Dataset; ③ Scenario (L. for learning, U. for unlearning); ④ Client type (C_u for clients involved in unlearning, C_o for clients excluded from unlearning); ⑤ Label type (Y_u for to-be-unlearned labels, Y_o for retained labels).

**Figure 4: Client-side time consumption across different SL frameworks.****Figure 5: Impacts of the expansion factor γ and privacy budget ϵ on SPLITWIPER and SPLITWIPER+ in terms of client-side training and transmission time consumption and test accuracy.**

linear increase in overhead as γ grows. The case $\gamma = \Gamma_p$ represents scenarios where γ_q is randomly selected as an integer from $[1, 3]$, with an average γ_q of 1.9. This introduces slightly higher overhead due to variability in γ_q compared to a uniform expansion factor. Since CIFAR-100 and CIFAR-10 are similar in size (~ 177 MB), their computational and transmission overheads are nearly identical. However, using CIFAR-10 and MNIST in scenarios with differing client tasks significantly increases training and transmission times.

Takeaway (faster). SPLITWIPER/SPLITWIPER+ mitigate overhead growth with one-way-one-off propagation, ensuring unrelated clients remain unaffected during unlearning. While SPLITWIPER+ introduces label expansion overhead, it matches SPLITWIPER in reducing computational and communication costs, achieving a 97-99% reduction in unlearning overhead compared to existing SLs.

7.3 Unlearning Effectiveness and Accuracy

The right sides of Tables 3–4 show the accuracy of the labels to be unlearned (Y_u) and other labels (Y_o) under both learning and unlearning scenarios across the test set. All frameworks, including existing ones and ours with varying γ and ε , effectively satisfy **G2** (effective unlearning with retained utility). They achieve **zero** accuracy for Y_u during unlearning (consistent with retraining results [9, 10, 45]) while maintaining the accuracy of Y_o . CIFAR-100, with more labels and fewer training examples per label, presents a greater learning challenge than CIFAR-10, resulting in lower accuracy across all frameworks. Additionally, ResNet18 performs worse than VGG on CIFAR-100, as VGG proves better suited for this task in our experiments.

Compared to Vanilla and U-shaped SL, SPLITWIPER achieves notable accuracy improvements for both Y_u and Y_o in the learning scenario. It surpasses Vanilla SL by 4.77% on CIFAR-10, 4.37% on CIFAR-100, and 2.98% in CIFAR-10 and MNIST scenarios. Against U-shaped SL, it shows gains of 4.18%, 3.82%, and 3.54% in the same respective scenarios. SPLITWIPER's key innovation is freezing client parameters after pre-training, preventing them from being influenced during server-side training. Shared server model back-propagation (\mathcal{F}_o^s) can cause overfitting or hinder learning in a client's model (\mathcal{F}_o^k). Freezing client parameters enhances efficiency, reduces overhead, and minimizes cross-client interference, improving the accuracy of $\text{Concat}(\mathcal{F}_o^k, \mathcal{F}_o^s)$.

Figs.5(c) and 5(d) compare SPLITWIPER and SPLITWIPER+, showing the effects of ε and γ on accuracy. While increasing γ slightly reduces accuracy, it remains acceptable. In SPLITWIPER+ with $\gamma > 1$, higher ε reduces noise and improves test accuracy. With $\varepsilon = 0.2$, SPLITWIPER+ can even outperform SPLITWIPER ($\gamma = 1$) due to minimal noise and label expansion functioning as data augmentation[38, 43], enhancing accuracy and generalization. Specifically, SPLITWIPER+ with $\varepsilon = 0.2$ and $\gamma = \Gamma_p$ achieves an average accuracy gain of 0.37% on CIFAR-10 and 0.12% on CIFAR-100, particularly improving \mathbb{Y}_o while maintaining comparable performance on \mathbb{Y}_u . This highlights SPLITWIPER+'s ability to strengthen privacy protection without compromising utility.

Takeaway (more effective). SPLITWIPER/SPLITWIPER+ effectively unlearn upon request while preserving the accuracy of retained data. The unlearning process in SPLITWIPER is over 8% more effective than existing SL, with SPLITWIPER+ further enhancing accuracy by an additional 0.37%.

7.4 Label-related Privacy Analysis

Considering **G3** (privacy-preserving label sharing), we evaluate the server's ability to infer real label characteristics, such as semantic information and label quantity, from the received masked labels and corresponding intermediate outputs. Given that the server lacks access to a local dataset and the shared labels are anonymized, it faces significant constraints in conducting semantic inference attacks on labels. We focus our analysis on the server's capability to deduce the actual number of labels used by clients, as the protection of label quantity is a pivotal advancement in our approach, effectively preventing attackers from leveraging the exact number of labels to perform high-accuracy supervised learning attacks. The inference of the label quantity requires unraveling the critical mapping mechanism $(\mathcal{G}_Y, \mathcal{G}_V) : \{Y_q, V_q^k\} \rightarrow \{Y_x^*, V_x^{*k}\}$ —the key difference between SPLITWIPER+ and SPLITWIPER. This highlights the enhanced privacy features of the former in protecting label information against server-based inference attacks.

In SPLITWIPER+, the server, in its attempt to infer the actual label quantity, is limited to employing clustering algorithms on the received data pairs $\{Y_x^*, V_x^{*k}\}$. These algorithms group the labels with closer corresponding intermediate outputs into the same cluster, hypothesizing that these labels are expanded from the same original label. We assess the effectiveness of this clustering-based label matching attack using two metrics:

- **Inclusive clustering accuracy.** This accuracy indicates the probability that labels derived from the same real label are inclusively grouped within the same cluster formed by the clustering algorithm. It underscores the containment relationship where a single cluster may encompass an entire real label group, but not necessarily align exactly with it.
- **Perfect clustering accuracy.** This accuracy indicates the proportion of instances where the clustered groups formed by the clustering algorithm match exactly with the true label expansion groups, reflecting a flawless alignment.

Inclusive clustering accuracy evaluates whether masked labels originating from the same label are grouped correctly based on their proximity in feature space. In contrast, perfect clustering accuracy assesses both the separation between groups and the algorithm's ability to distinguish them. These metrics provide insights into the effectiveness of SPLITWIPER+ in preserving label-related privacy while maintaining the structure of masked labels during clustering.

We employ the KMeans clustering algorithm [3, 42], a standard method renowned for its robustness and widespread use in detecting patterns in ungrouped data. Given that the real number of labels, Q , is confidential to the server, we utilize the elbow method [53] to determine an appropriate number of clusters K . Additionally, to illustrate the potential privacy breach in SPLITWIPER+, we also conduct experiments where Q is used directly as the parameter K for KMeans. This scenario simulates the impact of a hypothetical leakage of Q , revealing how such exposure could compromise the

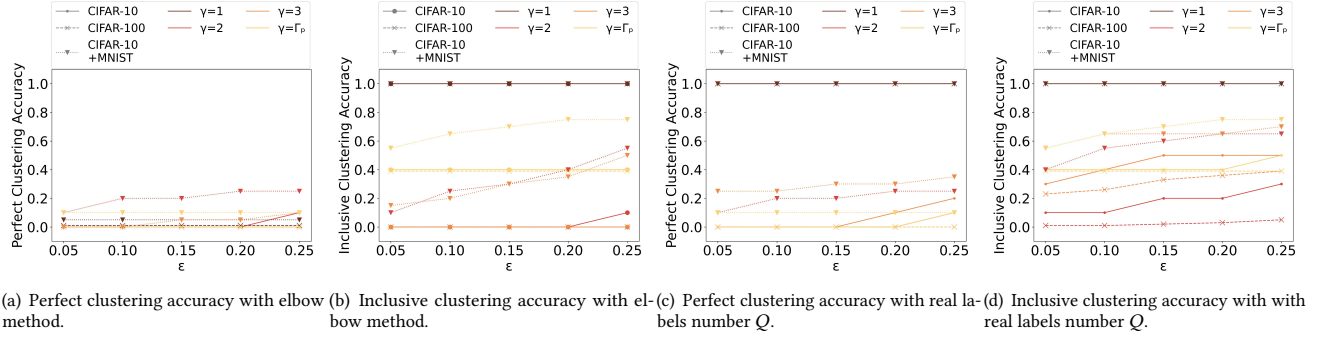


Figure 6: KMeans inclusive clustering accuracy and perfect clustering accuracy with Variable γ and ϵ , using the elbow method or the real label number Q as the cluster parameter. The low perfect clustering accuracy with both the elbow method and using Q as the parameter indicates the effective preservation of label-related privacy in SPLITWIPER+.

privacy measures implemented within SPLITWIPER+ and further emphasizing the critical importance and value of concealing the explicit label quantity.

Given that the pre-trained models on each client are consistent across both VGG and ResNet18 architectures—owing to the limited number of layers on resource-constrained clients and the application of training techniques such as dropout and batch normalization on the server side—the intermediate values exhibit negligible variation. Consequently, we do not differentiate between VGG and ResNet18 in our experiments, focusing instead on the impact of intermediate values on label privacy.

As illustrated in Fig.6(a), when Q (the true number of clusters) is unknown and the server estimates the optimal number of clusters using the elbow method, perfect clustering accuracy remains low, not exceeding 10% for $\gamma = 3$. This indicates that SPLITWIPER+, employing a label expansion strategy with $\gamma = 3$, preserves over 90% of label-related privacy, effectively protecting sensitive information against inference attacks. Furthermore, as γ increases or with non-uniform expansion factors ($\gamma = \Gamma_p$), perfect clustering accuracy decreases, providing enhanced privacy protection.

In contrast, the inclusive clustering accuracy (Fig.6(b)), is notably higher than the perfect clustering accuracy, due to its less stringent criteria. As more masked labels are clustered together by KMeans, the likelihood of satisfying the inclusive clustering accuracy increases. Under conditions of higher γ or with $\gamma = 1$, when more masked labels are grouped into the same cluster, the inclusive clustering accuracy rises. However, this scenario primarily indicates that masked labels derived from the same original label are grouped together, without providing the means to differentiate between distinct groups of labels. While this may reveal minimal related information, it is insufficient for practical use.

When the number of real labels Q is disclosed to the server (Fig.6(c)), clustering-based label matching attacks achieve higher perfect clustering accuracy, increasing by an average of 16% compared to when Q remains confidential. The increase highlights the additional privacy risks posed by the exposure of label counts, underscoring the necessity of implementing privacy-preserving measures in SPLITWIPER+. Furthermore, a larger γ amplifies the risks associated with the leakage of Q . Higher expansion factors result in an increased number of masked labels derived from each

original label, thereby providing additional information that facilitates clustering attacks when Q is known.

Takeaway (label privacy retained with one-way-one-off). SPLITWIPER+ protects label privacy and intermediate values through expansion and DP-based obfuscation, preventing server inference attacks via clustering techniques. It achieves over 90% privacy preservation when the true label count is concealed and approximately 70% even when the label count is exposed.

8 Conclusion

We achieved the first practical *Split Unlearning* framework with SPLITWIPER and SPLITWIPER+, which designed a one-way-one-off propagation to decouple propagation between clients and the server, enabling SISA unlearning with absent clients. SPLITWIPER+ further enhanced client label privacy against the server by employing DP. Experiments across diverse data distributions and tasks demonstrated that SPLITWIPER effectively achieved unlearning by involving only the requesting clients, resulting in **0%** unlearning accuracy, and **8%** better accuracy on retained data than non-SISA unlearning on existing SL frameworks. SPLITWIPER maintained *constant* overhead and reduced computational and communication costs by over **99%**. SPLITWIPER+ preserved over **90%** of label privacy, effectively protecting sensitive information from the server.

References

- [1] Yanhao Bao, Tatsukichi Shibuya, Ikuro Sato, Rei Kawakami, and Nakamasa Inoue. 2024. Efficient Target Propagation by Deriving Analytical Solution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 11016–11023.
- [2] Lucas Bourtole, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine Unlearning. In *IEEE Symposium on Security and Privacy (SP)*. 141–159. doi: 10.1109/SP40001.2021.00019
- [3] Paul S Bradley and Usama M Fayyad. 1998. Refining initial points for k-means clustering. In *ICML*, Vol. 98. Citeseer, 91–99.
- [4] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation unlearning. In *Proceedings of the ACM Web Conference (WWW)*. 2768–2777.
- [5] Jiaao Chen and Diyi Yang. 2024. Unlearn what you want to forget: Efficient unlearning for LLMs. *The Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2024).
- [6] Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H. Vincent Poor. 2021. Distributed Learning in Wireless Networks: Recent Progress and Future Challenges. *IEEE Journal on Selected Areas in Communications (JSAC)* 39, 12 (2021), 3579–3605.
- [7] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2021. When Machine Unlearning Jeopardizes Privacy. In *Proceedings*

- of the ACM SIGSAC Conference on Computer and Communications Security (CCS). 896–911.
- [8] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 499–513.
 - [9] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2023. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 37. 7210–7217.
 - [10] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. 2023. Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security (TIFS)* 18 (2023), 2345–2354.
 - [11] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2921–2926.
 - [12] Amol Deshpande. 2021. Syapse: Privacy-first Data Management through Pseudonymization and Partitioning. In *CIDR*.
 - [13] Dashan Gao, Sheng Wan, Hanlin Gu, Lixin Fan, Xin Yao, and Qiang Yang. 2024. Label Privacy Source Coding in Vertical Federated Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Springer, 313–331.
 - [14] Xinben Gao and Lan Zhang. 2023. {PCAT}: Functionality and data stealing from split learning by {Pseudo-Client} attack. In *32nd USENIX Security Symposium (USENIX Security 23)*. 5271–5288.
 - [15] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. 2021. Deep learning with label differential privacy. *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), 27131–27145.
 - [16] Sven Gowal, Sylvester-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. 2021. Improving robustness using generated data. *Advances in Neural Information Processing Systems* 34 (2021), 4218–4233.
 - [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.
 - [18] Zhibo Jin, Zhiyu Zhu, Hongsheng Hu, Minhui Xue, and Huaming Chen. 2023. POSTER: ML-Compass: A Comprehensive Assessment Framework for Machine Learning Models. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. 1031–1033.
 - [19] Junyaup Kim and Simon S Woo. 2022. Efficient two-stage model retraining for machine unlearning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4361–4369.
 - [20] Guanghao Li, Li Shen, Yan Sun, Yue Hu, Han Hu, and Dacheng Tao. 2023. Subspace based federated unlearning. *arXiv preprint arXiv:2302.12448* (2023).
 - [21] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2022. Label leakage and protection in two-party split learning. *International Conference on Learning Representations (ICLR)* (2022).
 - [22] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *IEEE/ACM International Symposium on Quality of Service (IWQOS)*. IEEE, 1–10.
 - [23] Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, PRANAY SHARMA, Sijia Liu, et al. 2024. Model sparsity can simplify machine unlearning. *Advances in Neural Information Processing Systems (NIPS)* 36 (2024).
 - [24] Xiao Liu, Mingyuan Li, Xu Wang, Guangsheng Yu, Wei Ni, Lixiang Li, Haipeng Peng, and Renping Liu. 2024. Decentralized Federated Unlearning on Blockchain. *arXiv preprint arXiv:2402.16294* (2024).
 - [25] Xiao Liu, Mingyuan Li, Xu Wang, Guangsheng Yu, Wei Ni, Lixiang Li, Haipeng Peng, and Renping Liu. 2024. Fishers Harvest Parallel Unlearning in Inherited Model Networks. *arXiv preprint arXiv:2408.08493* (2024).
 - [26] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. 2022. The Right to be Forgotten in Federated Learning: An Efficient Realization with Rapid Retraining. In *IEEE Conference on Computer Communications (INFOCOM)*. 1749–1758.
 - [27] Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. 2024. A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing Surveys (CUSR)* 57, 1 (2024), 1–38.
 - [28] Bradley C Love. 2002. Comparing supervised and unsupervised category learning. *Psychonomic bulletin & review* 9, 4 (2002), 829–835.
 - [29] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. 2021. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1622–1658.
 - [30] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. 2021. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems* 34 (2021), 5186–5198.
 - [31] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299* (2022).
 - [32] FY Osisanwo, JET Akinsola, O Awodele, JO Hinmikaiye, O Olakanmi, J Akinjobi, et al. 2017. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)* 48, 3 (2017), 128–138.
 - [33] Gupta Otkrist and Raskar Ramesh. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.
 - [34] Maarten G Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. 2019. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115* (2019).
 - [35] Youyang Qu, Xin Yuan, Ming Ding, Wei Ni, Thierry Rakotoarivelo, and David Smith. 2023. Learn to unlearn: A survey on machine unlearning. *arXiv preprint arXiv:2305.07512* (2023).
 - [36] Sérgio Luis Ribeiro and Emilio Tassato Nakamura. 2019. Privacy protection with pseudonymization and anonymization in a health IoT system: results from ocariot. In *IEEE International Conference on Bioinformatics and Bioengineering (BIBE)*. IEEE, 904–908.
 - [37] Thanveer Shaik, Xiaohui Tao, Haoran Xie, Lin Li, Xiaofeng Zhu, and Qing Li. 2024. Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* (2024).
 - [38] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6, 1 (2019), 1–48.
 - [39] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. 2019. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145* (2019).
 - [40] Yash Sinha, Murari Mandal, and Mohan Kankanhalli. 2023. Distill to Delete: Unlearning in Graph Networks with Knowledge Distillation. *arXiv preprint arXiv:2309.16173* (2023).
 - [41] Liwei Song, Reza Shokri, and Prateek Mittal. 2019. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 241–257.
 - [42] Muhammad Ali Syakur, B Khusnul Khotimah, EMS Rochman, and Budi Dwi Satoto. 2018. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In *IOP Conference Series: Materials Science and Engineering*, Vol. 336. IOP Publishing, 012017.
 - [43] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. 2019. Data augmentation using random image cropping and patching for deep CNNs. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 9 (2019), 2917–2931.
 - [44] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. 2022. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems* 35 (2022), 19332–19344.
 - [45] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. 2023. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* (2023).
 - [46] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. 2022. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 36. 8485–8493.
 - [47] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564* (2018).
 - [48] Paul Voigt and Axel Von dem Bussche. 2017. The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.
 - [49] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. 2022. Federated Unlearning: Guarantee the Right of Clients to Forget. *IEEE Network* 36, 5 (2022), 129–135.
 - [50] Maoqiang Wu, Guoliang Cheng, Dongdong Ye, Jiawen Kang, Rong Yu, Yuan Wu, and Miao Pan. 2023. Federated split learning with data and label privacy preservation in vehicular networks. *IEEE Transactions on Vehicular Technology (TVT)* (2023).
 - [51] Wen Wu, Mushu Li, Kaige Qu, Conghao Zhou, Xuemin Shen, Weihua Zhuang, Xu Li, and Weisen Shi. 2023. Split learning over wireless networks: Parallel design and resource management. *IEEE Journal on Selected Areas in Communications* 41, 4 (2023), 1051–1066.
 - [52] Danyang Xiao, Chengang Yang, and Weigang Wu. 2021. Mixing activations and labels in distributed training for split learning. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 33, 11 (2021), 3165–3177.
 - [53] Chunhui Yuan and Haitao Yang. 2019. Research on K-value selection method of K-means clustering algorithm. *J* 2, 2 (2019), 226–235.
 - [54] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*. 493–506.
 - [55] Zongshun Zhang, Andrea Pinto, Valeria Turina, Flavio Esposito, and Ibrahim Matta. 2023. Privacy and efficiency of communications in federated split learning. *IEEE Transactions on Big Data* 9, 5 (2023), 1380–1391.