KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Machine Learning

Second cycle, 30 credits

Unlearn with Your Contribution

A Machine Unlearning Framework in Federated Learning

YIXIONG WANG

Stockholm, Sweden, 2024

Unlearn with Your Contribution

A Machine Unlearning Framework in Federated Learning

YIXIONG WANG

Master's Programme, Machine Learning, 120 credits
Date: January 8, 2024

Supervisors: Masoumeh (Azin) Ebrahimi, Jalil Taghia, Selim Ickin
Examiner: Amir H. Payberah
School of Electrical Engineering and Computer Science
Host company: Ericsson AB
Swedish title: Avlär dig med ditt bidrag
Swedish subtitle: Ett ramverk för maskinavlärning inom federerad inlärning

© 2024 Yixiong Wang

Abstract

Recent years have witnessed remarkable advancements in machine learning, but with these advances come concerns about data privacy. Machine learning inherently involves learning functions from data, and this process can potentially lead to information leakage through various attacks on the learned model. Additionally, the presence of malicious actors who may poison input data to manipulate the model has become a growing concern. Consequently, the ability to unlearn specific data samples on demand has become critically important.

Federated Learning (FL) has emerged as a powerful approach to address these challenges. In FL, multiple participants or clients collaborate to train a single global machine learning model without sharing their training data. However, the issue of machine unlearning is particularly pertinent in FL, especially in scenarios where clients are not fully trustworthy.

This paper delves into the investigation of the efficacy of solving machine unlearning problems within the FL framework. The central research question this work tackles is: How can we effectively unlearn the entire dataset from one or multiple clients once an FL training is completed, while maintaining privacy and without access to the data?

To address this challenge, we introduce the concept of ”contribution,” which quantifies how much each client contributes to the training of the global FL model. In our implementation, we employ an Encoder-Decoder model on the server's end to disentangle these contributions as the FL process progresses. Notably, our approach is unique in that there is no existing work that utilizes a similar concept nor similar models.

Our findings, supported by extensive experiments on datasets MNIST and FashionMNIST, demonstrate that our proposed approach successfully solves the unlearning task in FL. Remarkably, it achieves results comparable to retraining from scratch without requiring the participation of the specific client whose data needs to be unlearned. Moreover, additional ablation studies indicate the sensitivity of the proposed model to specific structural hyperparameters.

Keywords

Machine Learning, Federated Learning, Machine Unlearning, Privacy

Sammanfattning

Här har de senaste åren bevittnat enastående framsteg inom maskininlärning, men med dessa framsteg kommer bekymmer om dataskydd. Maskininlärning innebär i grunden att lära sig funktioner från data, och denna process kan potentiellt leda till läckage av information genom olika attacker mot den inlärda modellen. Dessutom har närvaron av illvilliga aktörer som kan förgifta indata för att manipulera modellen blivit en växande oro. Följaktligen har förmågan att avlära specifika datasatser på begäran blivit av avgörande betydelse.

Federerad inlärning (FL) har framträtt som en kraftfull metod för att ta itu med dessa utmaningar. I FL samarbetar flera deltagare eller klienter för att träna en enda global maskininlärningsmodell utan att dela sina träningsdata. Emellertid är problemet med maskinavlärande särskilt relevant inom FL, särskilt i situationer där klienterna inte är fullt pålitliga.

Denna artikel fördjupar sig i undersökningen av effektiviteten av att lösa problem med maskinavlärande inom FL-ramverket. Den centrala forskningsfråga som detta arbete behandlar är: Hur kan vi effektivt avlära hela datasamlingen från en eller flera klienter när FL-utbildningen är klar, samtidigt som vi bevarar integritet och inte har tillgång till datan?

För att ta itu med denna utmaning introducerar vi begreppet ”bidrag,” som kvantifierar hur mycket varje klient bidrar till träningen av den globala FL-modellen. I vår implementering använder vi en Encoder-Decoder-modell på serverns sida för att reda ut dessa bidrag när FL-processen fortskrider. Det är värt att notera att vår metod är unik eftersom det inte finns något befintligt arbete som använder ett liknande koncept eller liknande modeller.

Våra resultat, som stöds av omfattande experiment på dataseten MNIST och FashionMNIST, visar att vår föreslagna metod framgångsrikt löser avlärandeuppgiften i FL. Anmärkningsvärt uppnår den resultat som är jämförbara med att träna om från grunden utan att kräva deltagandet av den specifika klient vars data behöver avläras. Dessutom indikerar ytterligare avläggningsstudier känsligheten hos den föreslagna modellen för specifika strukturella hyperparametrar.

Nyckelord

Maskininlärning, Federerad inlärning, Maskinavlärande, Sekretess

Acknowledgments

I would like to express my deepest gratitude to my supervisor at Ericsson Stockholm, Jalil, Selim, for his invaluable guidance, patience, and expertise. His insights and suggestions have been crucial in shaping this research work. I am also immensely grateful to my manager, Andreas, for his continuous support, encouragement, and for providing me with the opportunity to work on this fascinating project.

A special word of thanks goes to my KTH supervisor, Azin, whose expertise and understanding added considerably to my experience both academically and personally. Her guidance was instrumental in navigating the complexities of this research. I also wish to extend my gratitude to my KTH examiner, Amir, for his constructive feedback and critical evaluation of this work which immensely contributed to the quality of this thesis.

Furthermore, I would like to acknowledge the support and encouragement from my colleagues at Ericsson Stockholm, who provided a stimulating and fun environment to work in. Their perspectives and insights have been invaluable throughout this journey.

Last but not least, I extend my heartfelt thanks to my family and friends for their unwavering support and belief in me throughout my academic journey. Their encouragement and understanding have been a source of strength and motivation, making this achievement possible.

Stockholm, January 2024
Yixiong Wang

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The global fascination with machine learning was ignited when the machine learning model AlphaGo triumphed over human chess player Lee Sedol. Machine learning encompasses programs or entities that possess human-like abilities in learning and questioning, with machine learning being a prominent aspect. Machine learning, in essence, refers to algorithms capable of acquiring knowledge beyond explicit programming, and it encompasses deep learning. Deep learning, a subset of machine learning, leverages vast quantities of data to train artificial neural networks. In recent years, there has been tremendous progress in the fields of machine learning and artificial intelligence. Virtually all disciplines have embraced machine learning algorithms to enhance their services and drive economic growth. For instance, Spotify leverages machine learning algorithms to provide personalized song recommendations based on users' listening history. Similarly, Ericsson employs machine learning models to optimize the configuration of communication towers, resulting in energy savings. Furthermore, the remarkable yet controversial online chatbot, ChatGPT, stands as a massive machine learning model itself, trained on extensive language data.

In modern machine learning tasks, the harmonious interplay of both data and model is crucial. A deficiency in either can lead to significant drawbacks. Firstly, inadequate or poor-quality data can result in biased models, creating fairness issues and ethical dilemmas, especially in sensitive fields like healthcare or finance. Secondly, such insufficiency often leads to overfitting, where the model excels in training scenarios but fails to generalize to new, unseen data, severely limiting its practical applicability. Finally,

limited data scopes hinder the model's ability to capture complex patterns, compromising its effectiveness in accurately representing and interpreting the real-world complexity. These challenges underscore the importance of ensuring both robust data and model design in the development of effective and ethical machine learning solutions.

However, large models tend to memorize some sensitive information from training data. Nowadays, attacks that target at large models such as the training data extraction attack [1], are capable of recovering individual training examples by querying the model. On the other hand, security and privacy has been a hot social topic. Laws and regulations enforce companies and organizations to properly use personal data [2]. For instance, the European Union's General Data Protection Regulation (GDPR) [3], and the California Consumer Privacy Act (CCPA) [4] empower people with the rights to get more control of their data, and remove their personal information from any data collection entities.

On the other hand, modern machine learning algorithms heavily rely on training data, which often falls short of expectations regarding purity. These datasets may contain errors or noises that can adversely affect the performance of the model. Unwanted information, such as racist samples, may inadvertently exist within the dataset, and it is crucial to prevent the model from learning such biases. Additionally, some samples might be intentionally crafted by malicious participants, introducing potential risks to the model. Hackers can exploit these vulnerabilities, engaging in unauthorized activities like backdoor attacks to manipulate the model and unlawfully gather information.

Ideally, thorough scrutiny of the data could mitigate these issues. However, in the realm of contemporary machine learning tasks, this approach proves impractical due to the massive scale of data and the substantial labor cost associated with meticulous inspection. As a result, deviations and problems are often identified during product tests or through user feedback. Consequently, a pertinent question arises: Is it possible to remove specific information from a trained model after its training phase?

To tackle the problem of removing data and corresponding information from machine learning models, machine unlearning was proposed by Cao & Yang [5]. Several unlearning paradigms have been brought up and have sparked a interest and motivation both in academia and industry [6, 7, 8].

Although the utilization of machine unlearning can stem from a variety of reasons such as security, usability, fidelity, and privacy, it is frequently framed as a problem of preserving privacy. In this context, users have the ability to request the erasure of their data from computer systems and machine learning

models. In order to comply with a request for data removal, the computer system must delete all data associated with the user and eliminate any influence it may have had on the models trained using that data. This entails a complete unlearning of the user's information and its influence on the system.

In traditional machine learning approaches, the model which is highly centralised, has access to all the data in the dataset, yet in real scenarios, most likely, the learning task involves many participants, and data is confidential and exclusive to its owner. For instance, several insurance company intends to train a machine learning model that predicts potential clients. One feasible approach is to feed a model with their current customers' information as positive samples. Nevertheless, due to data regulations, any disclosure of personal data is illegal, thereby jeopardizing the principle of data completeness in traditional machine learning approaches.

To mitigate this problem, Google introduced FL as a potential solution [9, 10]. In the FL framework, the machine learning task is carried out through the collaboration of multiple clients, each holding their own confidential dataset. During the learning process, data privacy is maintained. In each federation round, clients upload their local model gradients to the server. The server then aggregates these gradients and sends back the aggregated result as the initial model parameters for the next federation round. This iterative process continues until specific convergence criteria are met.

Unlearning plays a crucial role in FL and holds practical significance. In telecommunications, machine unlearning is a required feature due to the increasing level of liability issues and corresponding strictness in data protection regulations. In most cases, it is not desirable or allowed to deploy a machine learning model that is trained on a dataset collected from a plurality of network operators to serve for a particular operator. In that case, the ownership of a ML model is not well-defined and it is non-trivial to pinpoint underlying reasons if a ML model under-performs than expectations. Similarly, a base station node may consist of different network elements from different vendors, e.g., antenna and radio units from one vendor, while the baseband and power supplier unit from another. A joint model trained on dataset collected from products of different vendors may also create similar challenges with respect to business sensitivity as well as model ownership and responsibility.

Unlearning under FL setups can be inherently challenging due to the entangled contributions of each client during successive federation rounds. This complexity arises from the nature of FL, where the information from local models is exchanged among participants during the aggregation process at the server's end.

This projects aims to propose a new approach that mitigate the unlearning problem under FL setup. The core idea is to employ a machine learning model at the server's end that monitors and updates the contribution from each FL client. The contribution model takes model parameters from each client as input, and estimates the contribution of each client. The training process of the contribution model is synchronized and aligned with that of the FL task. This means that the contribution model updates the contributions as well as its own parameters in each federation round. Inspired by the idea of VAE [11], we design the contribution model in an Encoder-Decoder fashion.

The contribution of this project underlies in: 1) we propose the concept of contribution in FL for machine unlearning. 2) we employ the contribution model in the server's end to record and update the contribution of each agent, thereby addressing the challenge posed by untrusted agents. 3) the training process of the contribution model is incorporated with the FL training, which suggests that no extra training rounds are required for unlearning.

1.2 Problem Definition

Consider the case where neither the server nor agents have access to training datasets after FL, is it possible to unlearn a specific agent without revealing potential information learned from the portion of the dataset that was asked to be removed?

Given an enterprise or cross-silo setting [12], in which agents are different organizations, and the number of agents is small, each client participates in all rounds, and each agent possesses relatively large amount of data. More specifically, we have N agents, and there are F rounds of federated training. We formulate the problem as follow:

Definition 1 *(Dataset) Let $\mathcal{C}$ denote the set of clients and $|\mathcal{C}|$ denote the number of clients. As an example, $\mathcal{C} = \{a, b, c\}$ denote a federation of three clients, named "a", "b" and "c". As an example, for the client "a", the training dataset is shown as $\mathcal{D}_a := \{(\mathbf{x}_n^a, \mathbf{y}_n^a) \mid n = 1, 2, \ldots\}$ containing the pairs of inputs $\mathbf{x}_n^a \in \mathbb{R}^{d_\mathbf{x}}$ and outputs $\mathbf{y}_n^a \in \mathbb{R}^{d_\mathbf{y}}$, where $d_\mathbf{x}$ and $d_\mathbf{y}$ indicate the dimensionality of inputs and outputs respectively.*

Definition 2 *(local model) Let $h_{\boldsymbol{\theta}_k} : \mathbf{x}^k \rightarrow \mathbf{y}^k$ denote the local model from client $k \in \mathcal{C}$, where $\boldsymbol{\theta}_k \in \Omega$ denotes all the model parameters and Ω is the space of parameters for all local models. The local model $h_{\boldsymbol{\theta}_k}$ is learned from its local training data $\mathcal{D}_k$.*

Definition 3 *(Global model) Let $\mathcal{M}$ denote the global, before unlearning, and denote $\mathcal{M}_{\setminus a}$ as the global model after unlearning agent "a", and $\mathcal{M}_{b,c}$ denotes the global model that is retrained with $\mathcal{D}_b$ and $\mathcal{D}_c$ from scratch.*

Definition 4 *(Contribution) Define the latent variable as the progressive contribution of each agent in the federated training process. Denote the latent variables as $\{\boldsymbol{z}_k \in \mathbb{R}^{d_{\boldsymbol{z}}} | k = a, b, c\}$, which is an embedding of the contribution from agent 'k' to the global model $\mathcal{M}$, and $\pi_k = \sum \boldsymbol{z}_k$, where $\pi_k \in \mathbb{R}$ is the sum of all contributions from all agents.*

Given the preceding definitions, our research inquiry quite naturally evolves: Is it possible for $\mathcal{M}_{\setminus a}$, when incorporated with $\boldsymbol{z}$, to reach a level of performance that is comparable to $\mathcal{M}_{b,c}$?

1.3 Purpose

Machine unlearning within an FL environment is a critical area of research. The goals of this Master's thesis involve leveraging a contribution model at the server's end that monitors and updates each agent's contribution for the purpose of unlearning, and the potential benefits of this study could ripple across numerous sectors of society.

This thesis work would bring significant advantages to data privacy. The ability to unlearn specific data points can protect user privacy and promote ethical data use. Users could request their data to be unlearned from models, thereby respecting their digital rights and autonomy. This aligns with ethical considerations around personal data use and the right to digital privacy. Also as mentioned in Section 1.1, Governments and other regulatory bodies would appreciate the mechanism to unlearn data from models as it aligns with global trends towards user data protection regulations, like GDPR [3] in Europe and CCPA [4] in California. It enhances their ability to enforce data privacy laws and protect citizens.

In parallel, machine unlearning contributes to sustainability. Machine learning, especially at scale, can be resource-intensive. Consider the massive computation that is involved in training with terabytes of data, the energy assumption and the carbon emission are non-negligible when it comes to retraining from scratch. The ability to unlearn data without retraining the entire model could dramatically reduce the computational resources and energy required, thus promoting sustainability in AI practices.

By addressing these anticipated issues, this thesis offers a balanced approach to machine learning, underscoring the importance of ethical

considerations, sustainability in the development and deployment of AI systems. The anticipated benefits and problem-solving potential of the project go beyond technological advancement, underscoring the necessity of incorporating societal and ethical perspectives into technological research and development.

1.4 Goals

The overarching objective of this project is to alleviate issues related to machine unlearning in an FL setting by introducing and employing the concept of 'contribution.' The main goal has been subdivided into the following targeted sub-goals:

1. Acquire an in-depth understanding of the principles, research orientations, methodologies, and challenges encompassing machine unlearning and FL.
2. Establish the necessary framework for FL.
3. Develop a model to measure contributions and subsequently maintain and update client contributions.
4. Customize the aggregation function at the server in line with the above.
5. Cultivate the ability to design and implement experiments to evaluate unlearning performance within the FL environment.
6. Utilize evaluation methods to assess the unlearning process whilst preserving an acceptable degree of FL performance.

Upon successful completion of the project, we anticipate the following deliverables and outcomes:

1. A viable machine unlearning framework within an FL context will be developed and verified for operation.
2. The concept of 'contribution' will be rigorously defined, supported by theoretical reasoning and mathematical assurances.
3. The effectiveness of the framework from computation and communication cost aspects will be verified through strategically designed experiments. Specifically, it will be demonstrated that post-unlearning, our model achieves performance metrics comparable to a model that has undergone retraining from scratch.

1.5 Research Methodology

In this thesis, an Encoder-Decoder styled contribution model is utilized at the server end to observe and update the contribution of each participating agent, eliminating the need for retraining. Previous unlearning methodologies applied at the server's end, as noted in [13, 14], necessitate retraining, thereby introducing additional computational burdens. [15] introduces gradient ascent, and [16] extends this by incorporating the Fisher Information matrix into gradient ascent, which constrains the update magnitude of different parameters in consideration of the significance of each parameter from preceding tasks. However, this approach operates at the agent's end, potentially compromising the global model due to the issues associated with untrusted agents.

1.6 Delimitations

In the context of FL setup, machine unlearning can be broadly categorized into three tasks. The first category is Class Unlearning, where clients request the global model to unlearn information associated with specific classes. The second category is Sample Unlearning, where a client seeks to have the global model unlearn a subset of its own data. If the subset encompasses the entire client's data, then Sample Unlearning becomes equivalent to Client Unlearning.

This project specifically concentrates on Client Unlearning, and all experiments are conducted under this particular assumption.

1.7 Structure of the thesis

Chapter 2 presents relevant background information about xxx. Chapter 3 presents the methodology and method used to solve the problem. …

Chapter 2

Technical Background

In this initial section, we strive to provide foundational knowledge pertinent to the thesis. The objective is to empower readers with a thorough comprehension of the key concepts involved, which are machine learning, machine unlearning, FL, and the fusion of these concepts.

Machine learning (ML) is a subset of artificial intelligence (AI) that employs algorithms and statistical models to enable systems to perform tasks without explicit instructions, relying instead on patterns and inference derived from data [17]. This dynamic field has progressed rapidly in recent years and has underpinned many technological advancements, from autonomous vehicles to personalized recommendation systems.

Machine Unlearning (MU), an extension of the traditional machine learning model, has been introduced to address issues of data privacy and model efficiency [5]. MU aims to ”forget” or remove specific data points from a trained model when the data is no longer relevant, erroneous, or when the data owner wishes for it to be deleted for privacy reasons. This could be accomplished without completely retraining the model from scratch. MU has also been explored as a tool for ensuring compliance with regulations such as the General Data Protection Regulation (GDPR), which grants individuals ”the right to be forgotten” [3]. A seminal work by Bourtoule et al., ”Machine Unlearning,” provides a comprehensive study on this topic [6].

FL is another emergent field in machine learning. Proposed by Google in 2016 [18], it enables models to be trained across many decentralized devices or servers holding local data samples, without requiring the transfer of those samples to a central server. This has significant implications for data privacy and security, as it allows data to remain in its original location, mitigating the risks of data breaches associated with central data storage. The work ”Towards

Federated Learning at Scale: System Design" provides an excellent overview of FL [19].

The intersection of Machine Unlearning and FL is a promising research area, creating a potential for machine learning systems that safeguard user privacy by learning in a distributed manner and 'unlearning' data when necessary.

2.1 Machine Unlearning

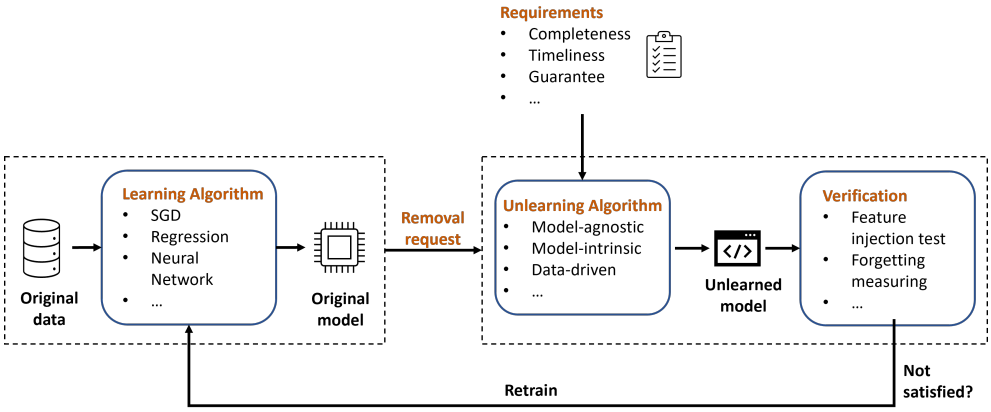


Figure 2.1: A Machine Unlearning Pipeline

In the realm of machine unlearning, training a model involves the use of a particular dataset and a learning algorithm. Once this training phase is complete, there arises a requirement to delete some elements, which could include sample(s), feature(s), or class(es), contingent on the particular unlearning task. Following this, an unlearning algorithm is executed. Moreover, machine unlearning is a process that requires cooperation of original data, learning algorithm, original model, unlearning algorithm, unlearning requirements, unlearned model, and verification schemes. Fig 2.1 shows the pipeline of machine unlearning. Before diving deep into details of each component, we give the mathematical definition of unlearning problems in general.

Definition 5 *(Exact Machine Unlearning) [20] Let $\mathcal{A}(\cdot)$ denote a learning algorithm and $U(\cdot)$ denote an unlearning algorithm. Further, let $\mathcal{D}$ denote the training dataset and $\mathcal{D}_f$ denote the dataset to be unlearned. For a hypothesis $\mathcal{T}$ in the space of all possible hypotheses $\mathcal{H}$, it is said that $\mathcal{U}(\cdot)$ exactly unlearns $\mathcal{D}_f$ if and only if:*

$$\Pr(\mathcal{A}(\mathcal{D}\backslash\mathcal{D}_f) \in \mathcal{T}) = \Pr(\mathcal{U}(\mathcal{D}, \mathcal{D}_f, A(\mathcal{D})) \in \mathcal{T}), \tag{2.1}$$

where $\backslash$ denotes set subtraction.

One thing to mention is that in Equ. 2.1, the unlearning algorithm $\mathcal{U}(.)$ has access to the unlearned dataset $\mathcal{D}_f$, however, this assumption may not be necessary, as Nguyen et al. [21] proposed an unlearning algorithm based on Bayesian Inference that does not require access to $\mathcal{D}_f$.

Def. 5 provides us with an ideal case where the unlearning algorithm has identical performance (at least identical output distribution) with the model retrained from scratch, while real scenarios often prove insufficiency of this assumption. This yields a relaxed definition, $\epsilon-$Approximate Unlearning.

Definition 6 *($\epsilon-$Approximate Machine Unlearning) Given $\epsilon > 0$, we say that $\mathcal{U}(.)$ is an $\epsilon-$Approximate machine unlearning algorithm if $\forall\mathcal{T} \in \mathcal{H}$:*

$$e^{-\epsilon} \leq \frac{\Pr(\mathcal{U}(\mathcal{D}, \mathcal{D}_f, A(\mathcal{D})) \in \mathcal{T})}{\Pr(\mathcal{A}(\mathcal{D}\backslash\mathcal{D}_f) \in \mathcal{T})} \leq e^{\epsilon} \tag{2.2}$$

The definition above provides a tolerance range concerning the deviation between the output space distributions of the unlearned and retrained models.

2.1.1 Unlearning Algorithm

Based on their specifications, unlearning algorithms can be sorted into different categories. Model-agnostic algorithms operate irrespective of the variety of learning models and offer theoretical guarantees. Conversely, model-intrinsic strategies are specifically designed for certain types of learning models. Lastly, there exist data-driven techniques that employ data segmentation and slicing to expedite the retraining process on a partitioned fraction of the original dataset.

Since machine unlearning can be classified to sample-level unlearning, feature-level unlearning, and class-level unlearning, we introduce some of the most classic algorithms within each category, in order to give readers an intuition of how unlearning algorithms work.

The goal of **sample-level** unlearning is to eliminate target sample(s) and their potential influence from the learning model. Bourtoule et al. [6] proposed a distributed framework as shown in Fig. 2.2. The global model is the aggregation of several constituent models. When it comes to machine unlearning, only the model whose shards contain the data that needs to be

unlearned is retrained with the retained data. One thing to mention is that this approach is different from FL, as the aggregation is done for once only if all the constituency models converge.

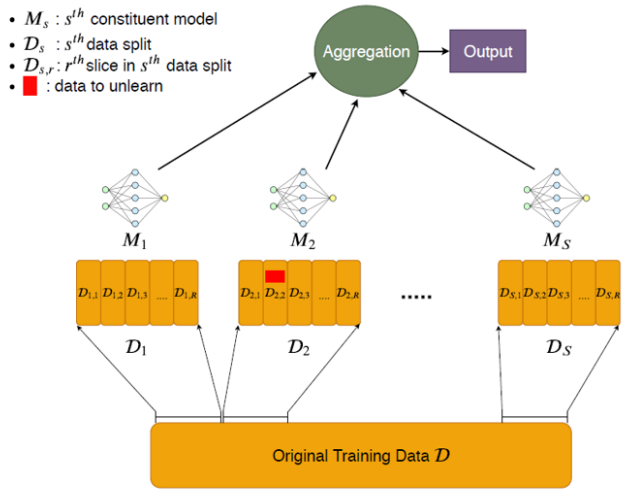


Figure 2.2: The framework of SISA, data is divided into shards and slices, once unlearning request arrives, the framework requires retraining of the target constituent model.

SISA reduces the time required for retraining by segmenting the data into shards and slices. Nonetheless, owing to the hierarchical structure, some data points not associated with the unlearning process may also be inadvertently unlearned. Additionally, it is inescapable that some level of retraining is still necessary.

When it comes to the **feature-level** unlearning, instead of forgetting the whole sample, the unlearning algorithm is asked to remove any potential influences on the learned model that can be traced back to specific features from the training data. [22] propose to replace the feature in each relevant sample with 0. Accordingly, a first-order update formula for gradient descent was proposed to make this approach learnable. This paper estimate the effect of features in training data on parameter updates, and gave a closed-form formula. However, retraining based on the 0-replacement algorithm is required.

In [2], Guo et al. propose a pipeline for the feature-level unlearning task in deep neural networks. Fig. 2.3 shows the architecture. The network model is divided into two sections. The top segment constitutes the main network, which in this case, is a VGG-based classifier. The bottom portion, the auxiliary

network, computes the required mutual information given the feature map and the labeled features from input space, and formulates a representation detachment loss function. The objective is to map the features in the latent space to input data, facilitating the removal of the target feature from the latent space using this map as a reference. The assumption is that, once this is done, the feature is also removed from the output, presuming the propagation in the main network is a Markov process. While this study uses a network to comprehend the correlation between features from different spaces for unlearning features from the output space, it presumes the feature to be unlearned is predetermined and is applicable only to feature-level unlearning with deep neural networks.

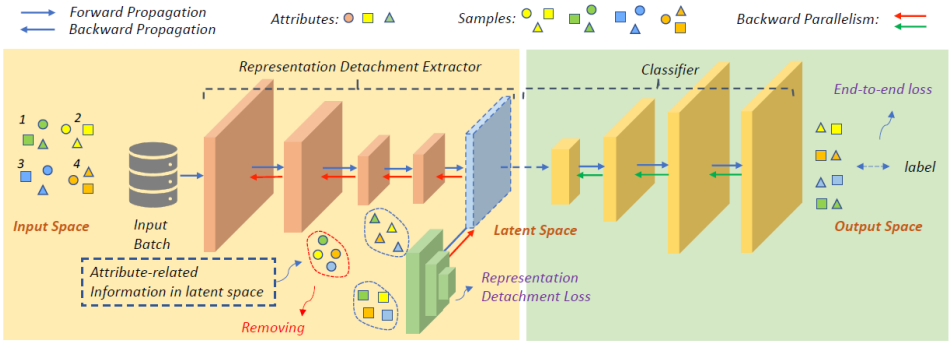


Figure 2.3: Architecture overview of feature unlearning. The top neural network is a classifier while the bottom one is the representation detachment extractor.

Class-level unlearning becomes imperative when information pertaining to an entire class needs to be forgotten. In such scenarios, Tarun et al. [23] suggest utilizing a noise matrix to modify the model weight for the unlearning process. This noise matrix aligns with the dimensionality of the input data and is devised by resolving an optimization problem where the loss of the original model with respect to data from the target unlearning class is maximized, factoring in the noise matrix. Following the acquisition of the noise matrix, a step to impair the model's parameters is needed to confirm the unlearning of the target class. An additional repair step is then executed to uphold the performance on other classes. It's important to note that the unlearning algorithm does not have access to the erased dataset, thereby offering a degree of privacy protection for the user.

An interesting consideration arises as to whether it is feasible to implement class-level unlearning devoid of any access to either the training or the retained

data. Drawing from the principles presented in [23], Chundawat et al. [24] introduce a novel approach in the form of a zero-shot unlearning algorithm. This approach diverges from the standard procedure of generating a noise matrix from actual training data. Instead, the authors suggest a unique method involving the synthesis of data through the resolution of a different optimization problem. The resulting synthetic data then forms the foundation of the noise matrix, thereby eliminating the necessity for access to the real training data. Pertaining to the crucial processes of impairing and repairing the model parameters, the technique of knowledge distillation is employed. This technique allows the preservation and transfer of knowledge from a complex model into a simpler one. In this context, it's used to modify the model's parameters effectively for unlearning and subsequently restoring them, again bypassing the requirement for the original training dataset.

2.1.2 Unlearning Requirement

To ensure the effective implementation of the unlearning algorithm, certain prerequisites must be met. The primary requirement is completeness, which infers that the performance of the model post-unlearning should be analogous to the model prior to unlearning [5]. Another critical aspect to consider is timeliness, given that the unlearning algorithm might necessitate additional time. Furthermore, it is important that machine unlearning algorithms effectively "forget" or unlearn specific data points, having minimal impact on the learning model. Aspects of efficiency and privacy should be taken into account, although assuring these can be challenging in certain practical scenarios, thus we might consider relaxing this requirement.

2.1.3 Unlearning Verification

Once an unlearning algorithm is implemented, the resultant model is referred to as the unlearned model. Nevertheless, before concluding that the unlearning process is complete, it's essential to validate the unlearned model. The general rule is that if the unlearned model and the model retrained from the start are indistinguishable, then the unlearning algorithm is deemed feasible [25]. The first verification technique is feature injection. Izzo et al. [26] put forth a method that concentrates on the weight associated with the target feature to be unlearned. The concept is that if a model unlearns a feature, the corresponding weight should drop to zero post-unlearning. However, this approach is applicable only to feature-level unlearning, which will be

discussed in subsequent sections. Relying on privacy attacks, measuring forgetting ensures that the unlearned data does not further influence the unlearned model. Jagielski et al. [27] introduced $\alpha-$forgetting, where a model $\alpha-$forgets a training sample if a privacy attack on that sample yields no more than a success rate of α. This methodology is more flexible compared to differential privacy, as it does not necessitate instant forgetting. However, when the volume of data samples to be unlearned is vast, this scheme fails in terms of verification time.

2.2 Federated Learning

2.2.1 Overview of FL

A representative case of FL is illustrated in Fig. 2.4. This diagram demonstrates the progressive update of the global model weight, θ_s, achieved by aggregating the local model parameters θ_a, θ_b, and θ_c. This procedure ensures a continuous adaptation of both the global and local model weights. Such adaptations are made possible due to the efficient interaction between the server and its associated agents, thereby upholding a system of timely and responsive updates.

2.2.2 Entanglement in FL

In the FL process, iterative training takes place over numerous rounds, with each client's initial model for a given round being sourced from the parameter updates in the preceding round. These updates inherently integrate contributions from all participating clients. As a result, parameter updates facilitate information entanglement, a phenomenon that intensifies with the progression of the training rounds.

Suppose we consider N clients, each possessing their distinct dataset denoted by $D_a, D_b, \cdots$. The initial model weights for this scenario are designated as θ_0. Further, $\theta_k^{(i)}$ signifies the model of client k after the i-th round of FL training, and $\theta_s^{(i)}$ represents the global model post-aggregation in the i-th round.

The parameter update function, represented by $G(\theta, D)$, accepts the initial model θ and the dataset D, and outputs the updated weight of this model. In the j-th round, the global model is given by:

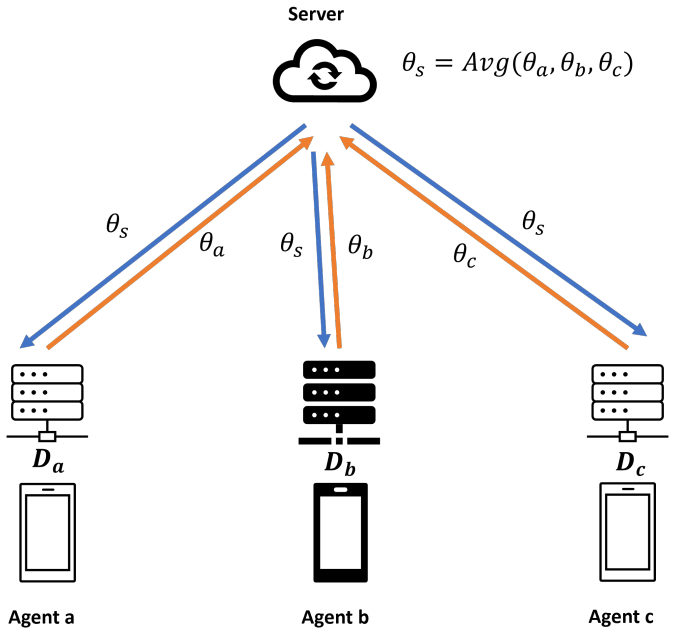


Figure 2.4: A toy case of FL. Three agents and one server participate in the FL task.

$$\theta_s^{(j)} = \frac{1}{N} \sum_{k=1}^{N} G(\theta_s^{(j-1)}, D_k) \tag{2.3}$$

where the marginal condition $\theta_s^{(0)} = \theta_0$ applies.

This iterative process is integral to FL and demonstrates the flow and updating of model parameters in a collaborative multi-client setting. Information entanglement, facilitated by parameter updates, allows all clients' contributions to be incorporated into each iteration, further enriching the model's learning process.

2.3 Machine Unlearning in FL

2.3.1 Challenge

Navigating the intersection of machine unlearning and FL introduces distinct challenges. In FL, training data remains private, belonging solely to its owner, thereby preventing the server from accessing individual data samples.

The nature of FL also necessitates a different categorization of unlearning

tasks compared to traditional machine unlearning. For instance, feature-level unlearning, which is commonplace in typical scenarios, becomes impractical in this context due to data confidentiality. Consequently, the concept of agent-level unlearning emerges, which requires the unlearning algorithm to remove all data and subsequent influences stemming from a specific agent. What's more, the distinction between each category in FL context can be vague. In the case where the distribution of data of each agent is non independent and identically distributed(non-iid), implying that every agent retains data corresponding to exclusive classes, agent-level unlearning can morph into class-level unlearning.

In addition, as discussed in Sec. 2.2.2, the occurrence of information entanglement in FL cannot be neglected. The iterative contribution of all clients to the final model's learning complicates the disentanglement of an individual agent's impact in an agent-level unlearning task. This characteristic of FL presents significant challenges to the execution and efficacy of machine unlearning.

2.3.2 Algorithm

This fact that servers lack access to training data has led many researchers in this field to implement unlearning algorithms on the agent's side, where data is directly accessible. Nevertheless, this idea brings potential reliability and security risks to the FL system. This is largely due to the commonly recognized assumption in FL that agents may manipulate their data or model weights to their advantage, thereby creating added challenges to maintaining secure and efficient unlearning in a federated environment.

In the work of Halimi et al. [15], the concept of agent-end gradient ascent is proposed as a solution to the challenge of agent-level unlearning. The central idea revolves around locating the weight that maximizes the loss function for the client intended to be erased, located within the l_2-norm ball with a radius of δ, centered around $\mathbf{w}_{ref}$. In this scenario, $\mathbf{w}_{ref}$ signifies the reference model that is aggregated from the weights of all agents, excluding the targeted agent.

Inspired by [15], Wu et al. [16] give a comprehensive examination of machine unlearning within the context of FL. It identifies three types of unlearning requests - class, client, and sample unlearning, and proposes a general pipeline to handle these requests effectively. The core of this pipeline involves the use of reverse stochastic gradient ascent (SGA) and elastic weight consolidation (EWC) as tools for federated unlearning. EWC takes the importance of each parameter to the previous old data into account. The

effectiveness of this approach was confirmed through numerous experiments that measured both unlearning efficacy and efficiency.

In the paper [13], the authors introduce an innovative unlearning technique for FL, known as FedEraser. This technique focuses on minimizing the loss function of the global model while retaining the accuracy of unlearning below a specific threshold. Model parameters are calibrated using the Frank-Wolfe algorithm to solve an optimization problem, enabling the adjustment of the remaining agents' weights to offset the influence of the unlearned agent. Crucially, the FedEraser algorithm operates on the server-side, thus ensuring a secure environment free from potential data manipulation by individual agents.

2.4 Variational Autoencoder (VAE)

VAEs are a type of generative model, which sets them apart from other types of deep learning models. First proposed by [11], VAEs learn a compressed, low-dimensional representation of the input data from a latent space. This type of algorithms have demonstrated considerable effectiveness in scenarios where the output isn't directly influenced by the input. Instead, certain latent variables, in conjunction with the input, jointly contribute to the formation of the output.

VAEs employ an encoder-decoder structure. The encoder models the posterior distribution $p(\mathbf{z}|\mathbf{x})$, where $\mathbf{z}$ is the latent variable and $\mathbf{x}$ is the observed data. The decoder then generates new data by sampling from the posterior distribution.

The objective of VAE is to maximize the evidence lower bound (ELBO) of the marginal likelihood of $\mathbf{x}$:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})] - \mathbb{D}[q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})] \tag{2.4}$$

where θ and ϕ are the parameters of the decoder and encoder, respectively, $p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$ is the likelihood of data given latent variable, and $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ is the approximate posterior. The second term is the Kullback-Leibler (KL) divergence between the approximate and prior distributions, acting as a regularization term.

To make it computationally feasible, the authors introduce the reparameterization trick for the approximate posterior, which allows backpropagation for neural nets:

$$\mathbf{z} = \mu + \sigma \odot \epsilon \tag{2.5}$$

where μ and σ are outputs of the encoder, $\odot$ is the element-wise multiplication, and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is a noise vector.

2.5 Background Summary

In this chapter, we provide a succinct overview of machine unlearning, FL, and variational autoencoders. We elaborate on the process of machine unlearning, encompassing its definition, the various algorithms used, the requirements for unlearning, and how to verify its effectiveness. In the context of FL, we explore the interactions between the server and agents, and the underlying causes of this entanglement. Further, we delve into the challenges presented by the confluence of machine unlearning and FL. To provide a comprehensive understanding, we discuss various algorithms associated with this intersection and their respective limitations. Lastly, we introduce the concept of Variational Autoencoders (VAEs), which forms the basis of the contribution model we propose in this project.

Chapter 3

Methodology & Implementation

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 outlines the research process. Section 3.2 presents a problem formulation. Section 3.3 details the contribution model's design, including its structure and the unlearning mechanism. Section 3.4 discusses the implementation of the FL-unlearning framework, covering structural parameters, training, and the unlearning algorithm.

3.1 Research Process

The research process in this project can be divided into four steps, shown as below:

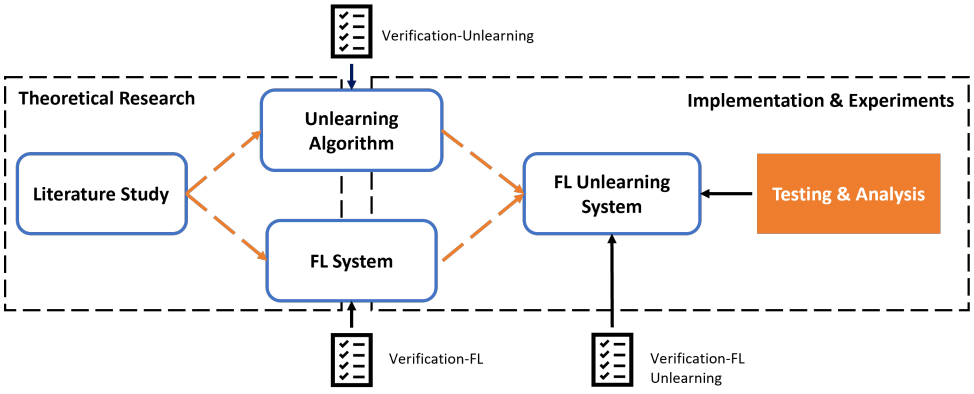


Figure 3.1: Research Process

Step 1 **Literature Study:** In this essential initial step, an in-depth review of existing literature in the fields of machine unlearning and FL is undertaken. This involves a comprehensive exploration of prevalent challenges, existing algorithms, and fundamental principles in these areas. The aim of this phase is to establish a thorough understanding, laying a robust foundation for the forthcoming stages of the project.

Step 2 **Problem Formulation and Solution Design:** This step centres around the precise definition of the problem within the unique context of FL. This involves proposing an algorithm and pipeline to handle the machine unlearning problem effectively. A significant focus in this stage is the development of a methodology capable of disentangling the contribution of each agent from the global model.

Step 3 **System Verification:** In this third phase, the integrated system along with its individual components – the FL and unlearning frameworks – is subjected to rigorous verification. This involves employing the MNIST dataset, a well-known benchmark in machine learning, to validate the efficacy and accuracy of the proposed solution.

Step 4 **Performance Testing:** The final step involves evaluating the performance of the system under various conditions, using the MNIST and FashionMNIST datasets. This phase also includes a sensitivity analysis to understand how different configurations affect the system's performance. The insights gained from this stage are crucial for refining the system and guiding future research.

Figure 3.1 shows the steps conducted to carry out this research.

3.2 Problem Formulation

In section 1.2, the background of the problem, relevant concepts, together with the goal of the project is given. With those definitions in mind, we give the problem formulation in details.

First, let's consider a FL system with N agents, each of which holds a local dataset $\mathcal{D}_i$ of size n. The global model parameterized by weights $\mathcal{M}$ is trained by aggregating the local updates from each agent. Here, we assume averaging over each agent's model parameter is used as the aggregation function. Denote the global model parameter after t rounds of updates as $\mathcal{M}^{(t)}$, which can be expressed as follows:

$$\mathcal{M}^{(t)} = \frac{1}{N} \sum_{i=1}^{N} f^{(t)}(\boldsymbol{\theta}_i^{(t-1)}; \mathcal{D}_i) \tag{3.1}$$

Following Def. 2, $f^{(t)}(\boldsymbol{\theta}_i^{(t-1)}; \mathcal{D}_i)$ is the local objective function computed by agent i at round t, which is usually an average of a loss function L over the local dataset $\mathcal{D}_i$. This can be written as:

$$f(\boldsymbol{\theta}_i; \mathcal{D}_i) = \frac{1}{n} \sum_{(x,y)\in\mathcal{D}_i} L(\boldsymbol{\theta}_i; x, y) \tag{3.2}$$

In FL settings, machine unlearning refers to the task of removing the influence of a certain data sample, agent, or class on the global model, without complete retraining. And in this project, we focus on a specific unlearning task, agent-level unlearning.

Denote the set of data to be unlearned as $\mathcal{D}_u$. The problem of machine unlearning in FL is to find a global model parameter $\mathcal{M}_{\setminus u}$ such that the distance between the output space distribution of $\mathcal{M}_{\setminus u}$, and that of $\mathcal{M}_{1,\cdots,u-1,u+1,\cdots,N}$ is minimized with a tolerance boundary.

Following Def. 6, this problem can be formally written as:

$$e^{-\epsilon} \leq \frac{\Pr(\mathcal{M}_{\setminus u} \in \mathcal{T})}{\Pr(\mathcal{M}_{1,\cdots,u-1,u+1,\cdots,N} \in \mathcal{T})} \leq e^{\epsilon} \tag{3.3}$$

where $\mathcal{M}_{\setminus u}$ is the global model after unlearning agent u, and $\mathcal{M}_{1,\cdots,u-1,u+1,\cdots,N}$ is the global model retrained from scratch with agent u erased from the FL training.

Together with the definition of contribution 4, Equ. 3.3 can be written as:

$$e^{-\epsilon} \leq \frac{\Pr((\mathcal{M}_{1,\cdots,N} - \pi_u \mathcal{M}_u) \in \mathcal{T})}{\Pr(\mathcal{M}_{1,\cdots,u-1,u+1,\cdots,N} \in \mathcal{T})} \leq e^{\epsilon} \tag{3.4}$$

where π_u is the summation of contribution of agent u, $\mathcal{M}_u$ denotes the local model of client "u" learned using local data $\mathcal{D}_a$, $\mathcal{M}_{1,\cdots,N}$ is the original global model. Note that $\mathcal{M}_{1,\cdots,N}$ is different from $\mathcal{M}$ in Equ. 3.1, we propose a aggregation scheme which shall be elaborated later.

The main challenges are (1) to define an efficient unlearning method that is suitable for the FL setting where data privacy is concerned, and (2) to find a way to disentangle the contributions of individual data samples to the global model.

3.3 Contribution Model

3.3.1 Model Architecture

The contribution model receives as the input the model parameters from the agents and learn the contributions of the agents over the course of federation. The model architecture is shown in Fig. 3.2 and is described in the following.

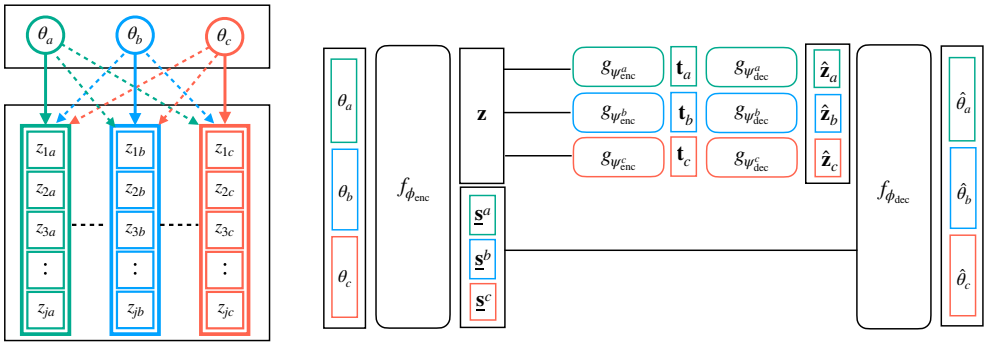


Figure 3.2: Model Architecture of three agents.

The left figure in Fig. 3.2 is a conceptual figure showing dependencies between model parameters and the latent contribution variables. Each agent model parameter is represented by a vector $(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b, \boldsymbol{\theta}_c)$ that needs to be learned. The latent contributions $(\mathbf{z}_a, \mathbf{z}_b, \mathbf{z}_c)$ are to be learned such they are disentangled from each others, where $\mathbf{z}_a \in \mathbb{R}^j$, and j is a hyper-parameter, so is $\mathbf{z}_b$ and $\mathbf{z}_c$.

The right figure in Fig. 3.2 shows the architecture of the contribution model at the server. The model receives the agents' model parameters and learns the contributions from the agents.

Let $\boldsymbol{\theta} := \text{vec}(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b, \boldsymbol{\theta}_c)$ denote the concatenation of the model parameters from all agents, and let $\mathbf{z} := \text{vec}(\mathbf{z}_a, \mathbf{z}_b, \mathbf{z}_c)$ denote the concatenation of the latent variables from the agents, named *latent contributions*. The latent contributions and the model parameters are not directly comparable. Thus, we define a second group of latent variables, named *latent support* set, that project a latent contribution into a space where it can be compared against the model parameters. As an example, for the agent "a" with the latent contribution $\mathbf{z}_a$ and model parameters $\boldsymbol{\theta}_a$, the latent support set is denoted as $\underline{\mathbf{s}}^a := \{\mathbf{s}^a_b, \mathbf{s}^a_c\}$. The product of the support variables and the latent contributions are referred to as the *projected model parameters* defined as $\tilde{\boldsymbol{\theta}}^a_b = \boldsymbol{\theta}_a(\mathbf{z}_b^\top \mathbf{s}^a_b)$ and $\tilde{\boldsymbol{\theta}}^a_c = \boldsymbol{\theta}_a(\mathbf{z}_c^\top \mathbf{s}^a_c)$ which can be compared against the agent model parameters $\boldsymbol{\theta}_a$.

The model is constructed as follows:

$$\begin{cases} \{\underline{\mathbf{s}}^a, \underline{\mathbf{s}}^b, \underline{\mathbf{s}}^c\}, \mathbf{z} = f_{\phi_{\text{enc}}}(\boldsymbol{\theta}), \\ \begin{cases} \mathbf{t}_a = g_{\psi^a_{\text{enc}}}(\mathbf{z}) \\ \hat{\mathbf{z}}_a = g_{\psi^a_{\text{dec}}}(\mathbf{t}_a) \end{cases}, \\ \begin{cases} \mathbf{t}_b = g_{\psi^b_{\text{enc}}}(\mathbf{z}) \\ \hat{\mathbf{z}}_b = g_{\psi^b_{\text{dec}}}(\mathbf{t}_b) \end{cases}, \\ \begin{cases} \mathbf{t}_c = g_{\psi^c_{\text{enc}}}(\mathbf{z}) \\ \hat{\mathbf{z}}_c = g_{\psi^c_{\text{dec}}}(\mathbf{t}_c) \end{cases}, \\ \hat{\boldsymbol{\theta}} = f_{\phi_{\text{dec}}}(\{\underline{\mathbf{s}}^a, \underline{\mathbf{s}}^b, \underline{\mathbf{s}}^c\}, \hat{\mathbf{z}}), \end{cases} \tag{3.5}$$

where $f_{\phi_{\text{enc}}}$ and $f_{\phi_{\text{dec}}}$ are the outer encoder and decoder models with the learnable parameters ϕ_{enc} and ϕ_{dec}, and $g_{\psi^a_{\text{enc}}}$ and $g_{\psi^a_{\text{dec}}}$ are the inner encoder and decoder models with the learnable parameters ψ^a_{enc} and ψ^a_{dec}, for a given agent "a". The inner encoder-decoders are responsible for generation of $\mathbf{z}$ through the latent variables $\mathbf{t}_a$, $\mathbf{t}_b$, and $\mathbf{t}_c$.

3.3.2 Optimization loss

We need to learn latent contributions from the agents such that they are generated from disentangled processes. Taking agent "a" as an example, it means $\mathbf{t}_a \perp \mathbf{t}_b$ and $\mathbf{t}_a \perp \mathbf{t}_c$, where $\perp$ denotes the disentanglement symbol. Furthermore, we also require to disentangle a given agent model parameters from its projected model parameters. Taking agent "a" as an example, that means $\boldsymbol{\theta}_a \perp \tilde{\boldsymbol{\theta}}^a_b$ and $\boldsymbol{\theta}_a \perp \tilde{\boldsymbol{\theta}}^a_c$. With these in mind, our loss function is constructed as:

$$\begin{aligned} l = {} & \ell_{\text{rec}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \ell_{\text{rec}}(\mathbf{z}, \hat{\mathbf{z}}) \\ & + \frac{1}{3}(\ell_{\text{reg}}(\boldsymbol{\theta}_a, \{\tilde{\boldsymbol{\theta}}^a_b, \tilde{\boldsymbol{\theta}}^a_c\}) + \ell_{\text{reg}}(\boldsymbol{\theta}_b, \{\tilde{\boldsymbol{\theta}}^b_a, \tilde{\boldsymbol{\theta}}^b_c\}) + \ell_{\text{reg}}(\boldsymbol{\theta}_c, \{\tilde{\boldsymbol{\theta}}^c_a, \tilde{\boldsymbol{\theta}}^c_b\})) \\ & + \frac{1}{3}(\ell_{\text{reg}}(\mathbf{t}_a, \{\mathbf{t}_b, \mathbf{t}_c\}) + \ell_{\text{reg}}(\mathbf{t}_b, \{\mathbf{t}_a, \mathbf{t}_c\}) + \ell_{\text{reg}}(\mathbf{t}_c, \{\mathbf{t}_a, \mathbf{t}_b\})), \end{aligned} \tag{3.6}$$

where ℓ_{rec} and ℓ_{reg} are reconstruction loss and regularization loss functions, and taking agent "a" as an example $\tilde{\boldsymbol{\theta}}^a_b = \hat{\boldsymbol{\theta}}_a(\hat{\mathbf{z}}_a^\top \mathbf{s}^a_b)$ and $\tilde{\boldsymbol{\theta}}^a_c = \hat{\boldsymbol{\theta}}_a(\hat{\mathbf{z}}_a^\top \mathbf{s}^a_c)$. The reconstruction loss functions are defined as:

$$\ell_{\text{rec}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \frac{\left\| \boldsymbol{\theta} - \hat{\boldsymbol{\theta}} \right\|^2}{\|\boldsymbol{\theta}\|^2}, \tag{3.7}$$

and

$$\ell_{\text{rec}}\left(\mathbf{z}, \hat{\mathbf{z}}\right) = \frac{\left\|\mathbf{z} - \hat{\mathbf{z}}\right\|^2}{\left\|\mathbf{z}\right\|^2}, \tag{3.8}$$

where we have chosen a normalized loss. The regularization losses are designed to encourage disentanglement among the variables. The first regularization loss, for agent "a" as an example, is designed to encourage disentanglement between $\boldsymbol{\theta}_a$ and the projected variables $\tilde{\boldsymbol{\theta}}_a^b$ and $\tilde{\boldsymbol{\theta}}_a^c$,

$$\ell_{\text{reg}}(\boldsymbol{\theta}_a, \{\tilde{\boldsymbol{\theta}}_b^a, \tilde{\boldsymbol{\theta}}_c^a\}) = -\frac{1}{2}\left(\frac{\left\|\boldsymbol{\theta}_a - \cos(\angle(\boldsymbol{\theta}_a, \tilde{\boldsymbol{\theta}}_a^b))\boldsymbol{\theta}_a\right\|^2}{\left\|\boldsymbol{\theta}_a\right\|^2} + \frac{\left\|\boldsymbol{\theta}_a - \cos(\angle(\boldsymbol{\theta}_a, \tilde{\boldsymbol{\theta}}_a^c))\boldsymbol{\theta}_a\right\|^2}{\left\|\boldsymbol{\theta}_a\right\|^2}\right) \tag{3.9a}$$

$$= -1 + \frac{1}{2}\left(\cos(\angle(\boldsymbol{\theta}_a, \tilde{\boldsymbol{\theta}}_a^b)) + \cos(\angle(\boldsymbol{\theta}_a, \tilde{\boldsymbol{\theta}}_a^c)\right), \tag{3.9b}$$

where $\angle$ computes the angle between the two vectors and cos is the cosine function. The loss is designed to encourage disentanglement between $\boldsymbol{\theta}_a$ and the projected variants $\tilde{\boldsymbol{\theta}}_a^b$ and $\tilde{\boldsymbol{\theta}}_a^c$.

Similarly, the second regularization loss, for agent "a" as an example, is designed to encourage disentanglement between $\mathbf{t}_a$ and the pair of $\mathbf{t}_b$ and $\mathbf{t}_c$,

$$\ell_{\text{reg}}(\mathbf{t}_a, \{\mathbf{t}_b, \mathbf{t}_c\}) = -1 + \frac{1}{2}\Big(\cos(\angle(\mathbf{t}_a, \mathbf{t}_b)) + \cos(\angle(\mathbf{t}_a, \mathbf{t}_c))\Big). \tag{3.10}$$

The loss function above shows an ideal scenario. If we think about the optimization problem behind, the reconstruction terms are the objectives that our contribution model aims to minimize, while the regularization terms add constraints to eliminate independencies between model parameters and latent variables. The idea sounds plausible in theory, however, the algorithm may not converge due to two main reasons. 1) the regularization on θ (Equ. 3.9) can be too strong, as the nature of FL makes local models homogeneous as the federation round goes on, complete orthogonality of local parameters indicates low performance of the prediction model. 2) the values of the two regularization terms are not in the same range, one may go up to infinity, which may manipulate the total loss.

To mitigate the problem above, we add temperature t_1 and t_2 to the regularization terms in Equ. 3.6, where t_1 and t_2 are hyper-parameters

indicating how important each terms is. Now Equ.3.6 can be rewriten as:

$$
\begin{aligned}
l = & \ell_{\text{rec}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \ell_{\text{rec}}\left(\mathbf{z}, \hat{\mathbf{z}}\right) \\
& + \frac{1}{3} t_1 (\ell_{\text{reg}}(\boldsymbol{\theta}_a, \{\tilde{\boldsymbol{\theta}}_b^a, \tilde{\boldsymbol{\theta}}_c^a\}) + \ell_{\text{reg}}(\boldsymbol{\theta}_b, \{\tilde{\boldsymbol{\theta}}_a^b, \tilde{\boldsymbol{\theta}}_c^b\}) + \ell_{\text{reg}}(\boldsymbol{\theta}_c, \{\tilde{\boldsymbol{\theta}}_a^c, \tilde{\boldsymbol{\theta}}_b^c\})) \\
& + \frac{1}{3} t_2 (\ell_{\text{reg}}\left(\mathbf{t}_a, \{\mathbf{t}_b, \mathbf{t}_c\}\right) + \ell_{\text{reg}}\left(\mathbf{t}_b, \{\mathbf{t}_a, \mathbf{t}_c\}\right) + \ell_{\text{reg}}\left(\mathbf{t}_c, \{\mathbf{t}_a, \mathbf{t}_b\}\right)), \quad (3.11)
\end{aligned}
$$

When $t_1 = t_2 = 0$, the loss function is relaxed to reconstruction loss only, and when $t_1 = t_2 = 1$, Equ. 3.11 is relaxed to Equ.3.6.

3.3.3 Aggregation

Given the inferred latent contributions from the agents, the aggregated model is computed as:

$$
\bar{\boldsymbol{\theta}} = \frac{\hat{\boldsymbol{\theta}}_a \pi_a + \hat{\boldsymbol{\theta}}_b \pi_b + \hat{\boldsymbol{\theta}}_c \pi_c}{\pi_a + \pi_b + \pi_c} \tag{3.12}
$$

where $\pi_i = \sum \hat{\mathbf{z}}_i, \forall i \in \mathcal{C}$. Notice that in computation of (3.12), we compute a weighted average using estimated model parameters $\hat{\boldsymbol{\theta}}_i$.

3.3.4 Unlearning

Once an unlearning request arrives, we subtract the product of the target agent's contribution and its latest local model's parameters from the global model. Consider a simple scenario where only three agents participate, namely A, B, and C. If an unlearning request for agent C arrives, the server processes the operation as described in Equation 3.13.

$$
\bar{\boldsymbol{\theta}}_{\setminus c} = \frac{\hat{\boldsymbol{\theta}}_a \pi_a + \hat{\boldsymbol{\theta}}_b \pi_b}{\pi_a + \pi_b} \tag{3.13}
$$

It's worth noting that our unlearning algorithm is designed to disentangle each agent's contribution from the federation history. Consequently, unlearning can be initiated anytime a request is made, irrespective of the FL's progress stage. However, considering that the contribution model needs several learning rounds to stabilize, we suggest invoking the unlearning function only after the completion of the FL.

3.4 Implementation

When implementing the contribution model, I followed Fig. 3.2 and build the model with PyTorch, I referred to Equ. 3.11 to implement the loss function, and Equ. 3.12 for the calibrated aggregation function. Tab. 3.1 shows structural parameters that either contribute to the contribution model directly or have an impact on the model configuration, and their corresponding usages as well as purposes. One thing to mention is that these parameters are also hyper-parameters, one can play with them to explore the influence of each parameter. In later chapters we will discuss the importance of these parameters and how they contribute to the performance of the unlearning system.

Table 3.1: Structural Parameters

Parameter	**Usage**	**Purpose**
in_dim	Dimensionality of the input for the local model at the agent's end.	Determines the dimensionality of the input for the contribution model.
n_agents	Number of agents in the FL task. This is equivalent to *in_dim*.	Has the same purpose as *in_dim*.
class_out	Number of labeled classes in the dataset.	Has the same purpose as *in_dim*.
unlearn_agents	Specifies agent(s) to be unlearned.	Enables direct unlearning or retraining while retaining specific agents.
z_dim	Dimensionality of the latent variable z for each agent.	Represents agent contributions in latent space and provides input to ψ_{enc}.
hidden_dim_encdec	Dimensionality of hidden layers in ϕ_{enc}, ϕ_{dec}, ψ_{enc}, and ψ_{dec}.	Enhances the generalization capability of the contribution model through hidden layers.

The pseudocode below shows the algorithm.

Algorithm 1 Learning to Unlearn Algorithm

Require: $\boldsymbol{\theta}_k, \quad k \in \{a, b, c\}$
Ensure: $\mathbf{z}_k, k \in \{a, b, c\}$
$\phi_{M.enc}, \phi_{M.dec}, \psi_{M.enc}, \psi_{M.dec} \leftarrow M.Initialization$
for round $i = 1, \cdots, F$ **do**
Agent's End
for agent $k \in \{a, b, c\}$ **do**
$\hat{y}_k = f(\mathcal{D}_k; \bar{\boldsymbol{\theta}})$
$l_k = l(y_k, \hat{y}_k)$
$\boldsymbol{\theta}_k \leftarrow M.optim(\nabla_{\boldsymbol{\theta}_k} l_k; \bar{\boldsymbol{\theta}})$
Update($\boldsymbol{\theta}_k$)
end for
Server's End
$\boldsymbol{\theta} \leftarrow M.concatenate(\boldsymbol{\theta}_a, \boldsymbol{\theta}_b, \boldsymbol{\theta}_c)$
$\mathbf{z}, \mathbf{s} \leftarrow f_{\phi_{M.enc}}(\boldsymbol{\theta})$
$\mathbf{t} \leftarrow g_{\psi_{M.enc}}(\boldsymbol{z})$
$\hat{\boldsymbol{z}}_a, \hat{\boldsymbol{z}}_b, \hat{\boldsymbol{z}}_c \leftarrow g_{\psi_{M.dec}}(\boldsymbol{t})$
$\hat{\boldsymbol{\theta}} \leftarrow f_{\phi_{M.dec}}(\hat{\boldsymbol{z}})$

$$
\begin{aligned}
l &= \ell_{M.rec}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \ell_{M.rec}(\mathbf{z}, \hat{\mathbf{z}}) \\
&+ \frac{1}{3}(\ell_{M.reg}(\boldsymbol{\theta}_a, \{\tilde{\boldsymbol{\theta}}_b^a, \tilde{\boldsymbol{\theta}}_c^a\}) + \ell_{M.reg}(\boldsymbol{\theta}_b, \{\tilde{\boldsymbol{\theta}}_a^b, \tilde{\boldsymbol{\theta}}_c^b\}) + \ell_{M.reg}(\boldsymbol{\theta}_c, \{\tilde{\boldsymbol{\theta}}_a^c, \tilde{\boldsymbol{\theta}}_b^c\})) \\
&+ \frac{1}{3}(\ell_{M.reg}(\mathbf{t}_a, \{\mathbf{t}_b, \mathbf{t}_c\}) + \ell_{M.reg}(\mathbf{t}_b, \{\mathbf{t}_a, \mathbf{t}_c\}) + \ell_{M.reg}(\mathbf{t}_c, \{\mathbf{t}_a, \mathbf{t}_b\})),
\end{aligned}
$$

$\phi_{M.enc}, \phi_{M.dec}, \psi_{M.enc}, \psi_{M.dec} \leftarrow M.Optimizer(\nabla_{\phi_{M.enc}, \phi_{M.dec}, \psi_{M.enc}, \psi_{M.dec}} l)$
Global Model: $\bar{\boldsymbol{\theta}} \leftarrow \frac{\boldsymbol{\theta}_a \pi_a + \boldsymbol{\theta}_b \pi_b + \boldsymbol{\theta}_c \pi_c}{\pi_a + \pi_b + \pi_c}$, $\pi_i = \sum \boldsymbol{z}_i, \forall i \in \mathcal{C}$.
end for
Unlearn (agent c for instance): $\bar{\theta}_{\setminus c} \leftarrow \frac{\boldsymbol{\theta}_a \pi_a + \boldsymbol{\theta}_b \pi_b}{\pi_a + \pi_b}$.

Chapter 4

Experiment

In this project, we carry out a series of experiments with several objectives in mind. Primarily, we aim to validate our FL system and our unlearning algorithm. Furthermore, we aim to assess the performance of our integrated FL-Unlearning system under diverse settings. By examining these different aspects, we seek to identify optimal configurations and pinpoint the most fitting use cases. Our ultimate goal is to facilitate the maximum exploitation of our system in terms of its potential applications and benefits. This chapter elaborates experiments we do in this project.

4.1 Experimental Environment

Since the implementation as well as all the experiments are conducted in Python, one should simply create a virtual environment with libaries and dependencies listed below:

- Python $\geq$ 3.10
- scikit-learn $\geq$ 1.2.1
- numpy == 1.23
- matplotlib == 3.7.0
- torch == 1.12.1
- seaborn == 0.12.2
- scipy == 1.10.0

- torchvision == 0.13.1
- pandas == 1.5.3

4.2 Dataset

In order to evaluate the performance and robustness of the devised FL-Unlearning system, a series of tests are carried out using two well-regarded and commonly used datasets: MNIST and FashionMNIST.

The MNIST dataset, also known as Modified National Institute of Standards and Technology dataset, is a classic in machine learning and computer vision. Its establishment by LeCun et al. [28] ushered in a rich collection of handwritten digits, broadly utilized as a performance measure for various machine learning algorithms. This dataset is composed of 60,000 training images and an additional 10,000 images for testing. Every image is presented in grayscale and of the dimension 28x28 pixels.

As an alternative to MNIST, the FashionMNIST dataset was developed to pose a more demanding problem while preserving the identical image size and the structure of training and testing splits as the original MNIST. Introduced by Xiao et al. [29], FashionMNIST is made up of 70,000 grayscale images spanning ten categories. Each category represents a distinct type of clothing item such as shirts, dresses, and footwear. The images, akin to those in the MNIST dataset, are of 28x28 pixels, making FashionMNIST an exemplary dataset for comparing and benchmarking machine learning algorithms.

Both of these datasets, given their clear-cut structure and relative simplicity, serve as excellent foundations for machine learning projects. Moreover, they provide a reliable framework for verifying the effectiveness of new algorithms and models, thus making them ideal choices for our experimental evaluation of the proposed FL-Unlearning system.

4.3 Experiment Design

The experiments in this project are designed with several specific objectives:

1. **Verification of the FL System:** The performance of the FL system is assessed by conducting a machine learning task under different environments, such as centralized learning, FL, and local learning. If the FL system is working as intended, it is expected that the performance order would be: centralized learning > FL > local training.

2. **Validation of the Unlearning Algorithm:** The regularization terms in Equation 3.11 are introduced with the aim of disentangling the contributions of individual agents. Cosine similarity, which can be interpreted as a measure of entanglement between inputs, is used to test this aspect. In this experiment, the cosine similarity between the vectors $\mathbf{z}$ is compared both with and without the application of the regularization terms. The validity of the unlearning algorithm is demonstrated if the cosine similarity decreases with the regularization as the federation round progresses.

3. **Assessing the Impact of Data Homogeneity in FL tasks:** This experiment evaluates how the distribution of data amongst agents in the FL task, particularly the homogeneity of data assigned to each agent, affects the performance of the FL-Unlearning system. It aims to find out the use case under which our FL-Unlearning system performs the best.

4. **Sensitivity Analysis:** The performance of the system under different configurations, such as the dimensionality of latent variables $\mathbf{z}$, the temperature of the regularization terms in the loss function, and the effect of random seeds during initialization, is investigated. This test allows us to understand the robustness of the system under different settings.

4.4 Verification of the FL System

To ensure the robustness of the FL system, we conduct a preliminary verification test comparing FL, local learning, and centralized learning approaches. For the FL and local learning, five agents participate. The dataset utilized is the widely-known MNIST, but with a specially tailored distribution in the training set.

Tab. 4.1 displays the class distribution for each agent's training dataset. A ✓symbol denotes the presence of a specific class in an agent's training data, while an ✗indicates its absence. Notably, every agent lacks data for four classes. Despite this, the frequency of each class across all five agents remains consistent, with every ✓representing 50 instances. This distribution results in a total of 300 training samples per agent. Based on this arrangement, the expected performance hierarchy is evident: centralized learning (CL) should surpass FL, which in turn should outperform local learning (LL). For the FL

and LL, we executed 100 rounds of training, each consisting of 200 epochs. To ensure convergence, the centralized learning approach was also trained for a total of 200 epochs.

Table 4.1: Class Distribution in Training Dataset for the Sanity Check

Agent\Class	0	1	2	3	4	5	6	7	8	9
0	✓	✗	✓	✗	✓	✓	✗	✓	✗	✓
1	✗	✓	✓	✓	✓	✗	✓	✗	✓	✗
2	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
3	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓
4	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗

For the testing phase, the dataset comprises a balanced selection from each class, resulting in a total of 600 samples—60 samples from each of the 10 classes. For both FL and LL, every agent undergoes testing using this shared dataset. The final performance metric is the average accuracy across all agents. The accuracy for centralized learning is computed directly. Tab. 4.2 illustrates these results, confirming our anticipated performance ranking: CL excels, followed by FL, with LL trailing. These findings validate the integrity of the FL system, laying a solid foundation for subsequent experiments.

Table 4.2: FL System Sanity Check Results

FL	**Local Learning**	**Centralized Learning**
0.8451	0.546 7	0.990 7

4.5 Validation of Unlearning Algorithm

In modern distributed learning architectures, understanding the distinct contributions of participating agents is paramount. As outlined in Sec. 4.3, our objective with the contribution model is to effectively disentangle these distinct contributions. Before advancing with more complex experiments, it's essential to verify if the current model meets this objective.

A notable feature of our model is the inclusion of regularization terms in Eq. 3.11. These terms are designed to enforce the orthogonality of the latent variables, denoted as $\mathbf{t}$s. Historically, orthogonality in hyper-space is associated with the notion of independence in probability theory. This allows us to utilize the average cosine similarity between $\mathbf{t}$s as a metric to indicate the independence of contributions.

For the purposes of our experiment, several parameters were defined in line with the specifics of the MNIST data. Specifically, *in_dim* to 784, and *class_out* to 10. For the contribution model, *z_dim* was set to 5, and *hidden_dim_encdec* was 50, . Additionally, the agent parameters were set as *n_agents* to 5, and *unlearn_agents* to 4 (denoting the agent identified as number 4). It's noteworthy to mention that the data configuration adopted in this experiment aligns with those detailed in Sec. 4.4.

Fig. 4.1 provides a visual representation of the evolution of average cosine similarity under two conditions: with and without the aforementioned regularization terms. The blue trajectory signifies the cosine score in the absence of regularization, where both temperature coefficients in Eq. 3.11 are set to 0. The orange trajectory, however, denotes the score with the temperature coefficient in Eq. 3.11 set to 1. From the plotted data, it becomes evident that the cosine score diminishes when regularization is applied, indicating an enhanced orthogonality between agents' contributions. Conversely, in the absence of regularization, the average cosine similarity shows an increasing trend, stabilizing eventually around a value of 0.6. This behavior can be interpreted as an evidence of increased entanglement in FL processes. A salient observation here is that the average cosine similarity, even with the application of regularization, doesn't converge to a perfect zero. While this might seem counterintuitive, it's crucial to interpret this not as a flaw, but as a realistic reflection of the inherent complexities in distributed learning. Nonetheless, the observable trend underscores the potential of our algorithm to achieve meaningful decoupling in FL dependencies.

4.6 Impacts of Data Homogeneity in FL tasks

In our experiment, we examine the influence of data distribution in the training set on the performance of our FL-Unlearning system. Our goal is to determine the optimal conditions for the system's peak performance, which is crucial for its practical application.

Our experimental design aligns with the classification task of MNIST. With ten classes available, we can tailor the training dataset for each agent. If each agent has samples exclusively from unique classes, the data distribution among agents is entirely heterogeneous. In this scenario, the FL task becomes almost as challenging as local training. Conversely, when all agents hold data from identical classes, the distribution is wholly homogeneous, causing the

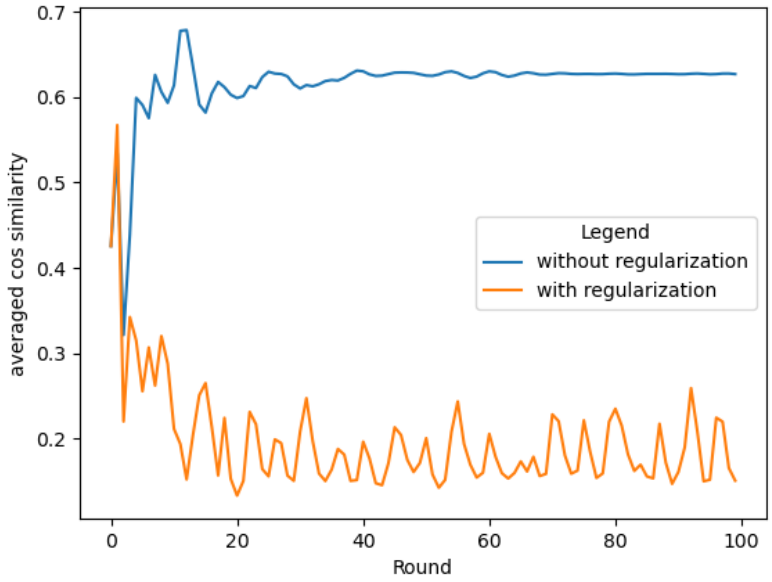


Figure 4.1: Cosine Similarity between latent variables **t**, with and without regularization.

FL task to closely resemble central training. However, the above perspective mainly sheds light on the FL system's performance. To truly understand the FL-Unlearning system's efficiency, it's vital to pinpoint its most favorable use case.

Homogeneous Data

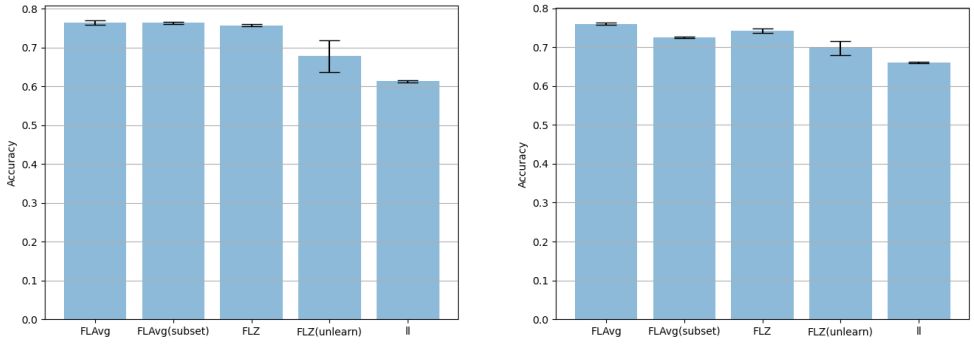
Starting with the homogeneous scenario: each agent receives an even distribution of data samples across the ten classes, maintaining uniformity in class samples for every agent. Specifically, each agent collects 30 samples from each class, totaling 300 training samples.

This experiment was performed on both the MNIST and FMNIST datasets. Except for the assignment of training data, all configurations remain consistent with those outlined in Sec. 4.5. We utilized multiple learning frameworks for each dataset to compare performance and derive deeper insights. These frameworks include:

- **FLAvg**: FL using average as the aggregation function.
- **FLAvg(subset)**: FL with agents retained retrained from the beginning.

- **FLZ**: Our proposed FL-Unlearning framework.
- **FLZ(unlearn)**: The unlearned version of our FLZ model, aligned with our proposed unlearning mechanism.
- **ll**: Localized learning.

For each framework, experiments were conducted five times with varying random initializations. This approach allowed us to compute the mean and variance and to assess the stability of each framework. The consolidated results can be seen in Fig. 4.2.



(a) MNIST (b) FMNIST

Figure 4.2: Error Bar Plots Comparing Performance of Different Learning Frameworks Using Training Datasets with iid Distribution.

In Fig. 4.2, the error bar plot distinctly presents the accuracy performance of various learning frameworks on both the MNIST and FMNIST datasets.

The x-axis enumerates the five selected learning frameworks: FLAvg, FLAvg(subset), FLZ, FLZ(unlearn), and ll, while the y-axis measures their respective accuracies.

Conceptually, the performance trends of FLZ and FLZ(unlearn) should resemble those of FLAvg and FLAvg(subset). Ideally, FLAvg(subset) should represent the performance ceiling for FLZ(unlearn). The narrower the performance divergence between them, the better the unlearning efficiency. The 'll' framework is included for a sanity check; we anticipate its performance to be the lowest among the frameworks.

Zooming into Fig. 4.2a, FLAvg and FLAvg(subset) performances are nearly identical, both gravitating towards an accuracy of approximately 0.76 with minimal deviation. Within the FLZ category, while FLZ displays robust accuracy with a slight deviation, FLZ(unlearn) manifests a significant

performance dip and increased deviation, hinting at diminished stability. These observations underscore the relative success of FLZ, with FLZ(unlearn) indicating potential areas of enhancement.

Conversely, the results from Fig. 4.2b align favorably with our expectations. All the represented bars and their corresponding minimal deviations indicate consistency and stability.

The homogeneity of the training dataset, paired with the intrinsic difficulty gradient between MNIST and FMNIST, may account for these outcomes. Given the uniformity across agents, deviations in local models are largely absent. When a local model slated for unlearning (and its associated contribution) is subtracted, it can exert significant perturbations on the global model. Further, MNIST's simplicity relative to FMNIST—translating to reduced model complexity—may negatively sway performance.

Heterogeneous Data

This experiment emphasizes data that is entirely heterogeneous in distribution. The design principle is to allocate distinct class data exclusively to individual agents. The distribution of class data across the agents is depicted in Tab. 4.3.

Table 4.3: Distribution of Training Data for Agents with Complete Heterogeneity

Agent\Class	0	1	2	3	4	5	6	7	8	9
0	316	284	✗	✗	✗	✗	✗	✗	✗	✗
1	✗	✗	308	292	✗	✗	✗	✗	✗	✗
2	✗	✗	✗	✗	314	286	✗	✗	✗	✗
3	✗	✗	✗	✗	✗	✗	295	305	✗	✗
4	✗	✗	✗	✗	✗	✗	✗	✗	292	308

Figure 4.3 displays the performance metrics for the given frameworks. Notably, all five frameworks exhibit relatively low accuracies, ranging from 0.2 to 0.25. One observation that stands out is the superior accuracy of FLZ(unlearn), even though it also showcases the highest deviation. This outcome deviates from our initial expectations.

Reflecting upon the data distribution, the underlying reason for such results becomes evident. The inherently heterogeneous distribution among agents in this scenario implies a significant lack of shared information. Such limited overlap in data often complicates FL tasks, leading to divergence in the learning process. Moreover, the impurity in FLZ further amplifies the

challenges, causing the performance of FLZ(unlearn) to deteriorate.

Contrasting this with the earlier homogeneous data distribution experiment, it's evident how data distribution intricacies profoundly affect the learning outcomes. While homogeneous distribution aids in maintaining consistency across federated agents, heterogeneity tends to scatter the learning focus, leading to unpredictable results.

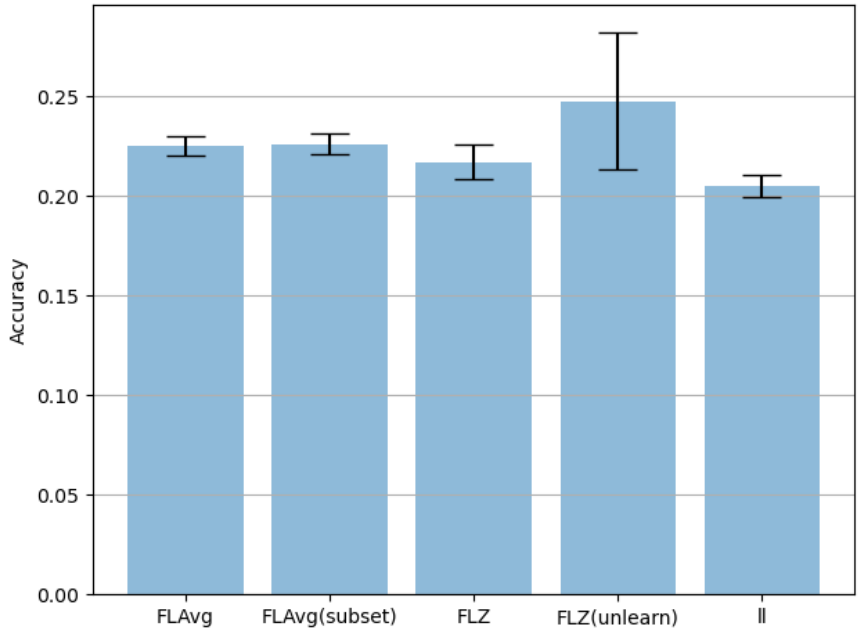


Figure 4.3: Error Bar Plots Comparing Performance of Different Learning Frameworks Using Training Dataset with heterogeneous Distribution.

Slightly Heterogeneous Data

Distributed learning often encounters scenarios where data isn't entirely homogeneous across agents, yet isn't completely divided by distinct classes either. This section examines this middle ground: the realm of slightly heterogeneous data distribution. The motivation behind this is grounded in our prior observations: purely homogeneous data distribution simplifies the FL task, as agents share a consistent data realm, making consensus relatively straightforward, as seen in our previous homogeneous experiment. In contrast, a completely heterogeneous setup complicates the task, creating a scenario where agents lack shared information, leading to fragmented learning outcomes, as evidenced by the performance metrics in our heterogeneous experiments.

A slightly heterogeneous setup aims to bridge this gap, introducing just enough variability among agents to make the task challenging yet not insurmountable. It represents a more realistic scenario that many real-world FL systems may face.

Tab. 4.4 provides a snapshot of this data distribution among the agents. As expected, the results shown in Fig. 4.4 from this setup show a promising trend. Both FLAvg and FLAvg(subset) produce high accuracies, hovering around 0.8. Interestingly, both FLZ and FLZ(unlearn) follow a similar performance trajectory, implying a coherent learning process even under slightly uneven data distribution. The stability of these frameworks is further underscored by the small deviations observed, suggesting the models are reliable under this data distribution scenario. Notably, the higher accuracy of FLZ(unlearn) compared to FLZ, mirroring the trend observed with FLAvg and FLAvg(subset), suggests that the agent intended for unlearning was detrimental to the global model, highlighting the potential benefits of targeted unlearning.

In essence, this experiment strikes a balance, offering a lens into the performance of FL frameworks under realistic data distribution conditions while also highlighting the nuances and challenges posed by data heterogeneity.

Table 4.4: Assignment of Training Data to Agents under Uneven Distribution

Agent\Class	0	1	2	3	4	5	6	7	8	9
0	50	10	50	10	50	50	10	50	10	50
1	10	50	50	50	50	10	50	10	50	10
2	50	10	50	10	10	10	50	50	50	50
3	10	50	10	50	50	50	10	10	50	50
4	50	50	10	50	10	50	50	50	10	10

Conclusion

Across our extensive experimentation, spanning three distinct data distributions - homogeneous, completely heterogeneous, and slightly heterogeneous - we've gleaned profound insights into the behavior of FL frameworks and their adaptability to varying data landscapes.

Homogeneous Data: In the case of homogeneous distribution, while the framework exhibited consistent outcomes across federated agents, it arguably rendered the task overly simplistic. The commonality in data across agents reduced the complexity and challenges typically faced in real-world scenarios.

Heterogeneous Data: At the opposite end of the spectrum, the completely heterogeneous data posed severe complications. The lack of shared

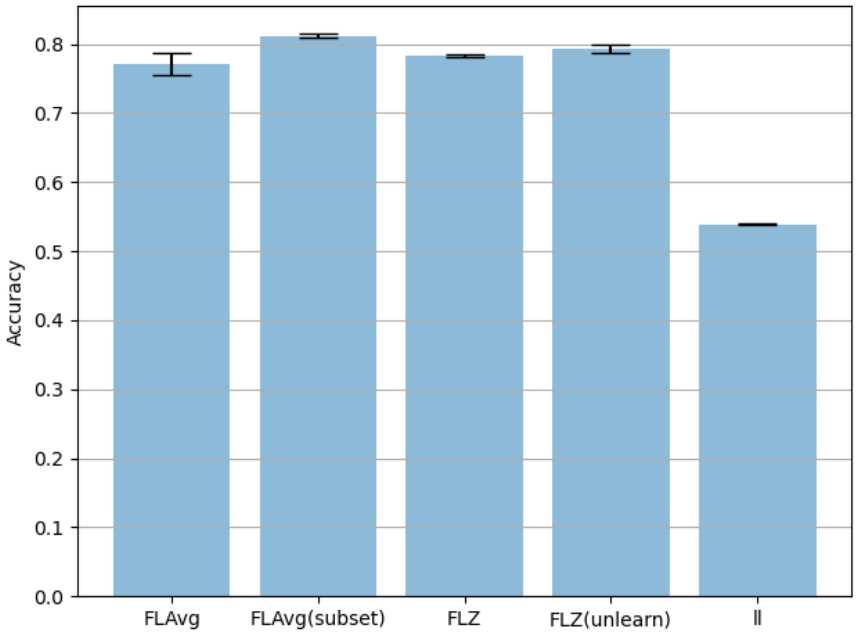


Figure 4.4: Error Bar Plots Comparing Performance of Different Learning Frameworks Using Training Datasets with Uneven Distribution.

information among agents led to disparate learning paths, resulting in fragmented and often unpredictable results. This distribution, though challenging, is also a less common scenario in practical applications of FL.

Slightly Heterogeneous Data: Nestled between the two extremes, the slightly heterogeneous data distribution struck a harmonious balance. It closely mirrored realistic scenarios, making the task challenging yet achievable. This setup proved most beneficial, yielding promising results and stability across the frameworks. Most notably, our FL-Unlearning model showcased its peak performance under this distribution, reiterating its potential for practical application.

In summation, while each data distribution provided valuable learnings, it's the slightly heterogeneous setup that stood out as the most conducive environment for FL, especially for our FL-Unlearning model. The balance it offers between challenge and achievability makes it the ideal benchmark for real-world applications, suggesting that FL models, and in particular our FL-Unlearning model, are best positioned to thrive in environments that present a moderate degree of data variability across agents.

4.7 Sensitivity Analysis

Dimensionality of latent Variable z

Recall the design of our contribution model. The latent variable $\mathbf{z} \in \mathbb{R}^{n \times d}$ serves as input for the internal Encoder-Decoder (ψ). The summation of $\mathbf{z}$ over dimension d represents each agent's contribution. While n (the number of participating agents) is determined by the specific task, the embedding dimension d is treated as a hyper-parameter. In this study, we investigate the sensitivity of our model to various d values.

We designate the agent count (n) as 5 and test the embedding dimension d at 2, 5, and 10. The framework undergoes training and testing on MNIST, maintaining a consistent data distribution as presented in Tab. 4.4. For the unlearning process, agent 4 is specifically targeted. In each d scenario, we run experiments using FLAvg, FLAvg(subset), FLZ, and FLZ(unlearn). Ideally, the accuracy trend under FLAvg should mirror that of FLZ. Meanwhile, FLZ(unlearn) should correspond with FLAvg(subset), with FLAvg(subset) acting as a theoretical performance ceiling for FLZ(unlearn). All models undergo training until convergence.

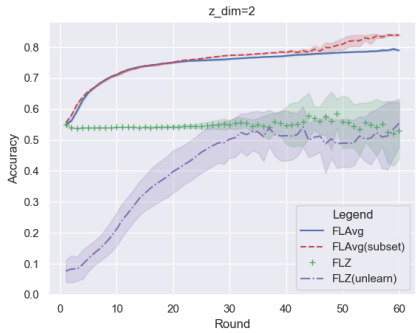
As illustrated in Fig. 4.5, optimal performance for FLZ(unlearn) is achieved when $d = 5$. The FLZ curve closely follows the FLAvg trajectory, and FLZ(unlearn) closely matches FLAvg(subset), with only a marginal performance gap. For $d = 2$, the performance curve for FLZ(unlearn) displays a significant decline in accuracy; rather than paralleling FLAvg(subset), it aligns more with FLZ. At $d = 10$, the performance of FLZ(unlearn) is comparable to FLAvg(subset), but the trend deviates.

Worth noting is the subpar performance of FLZ across all d settings. Instead of reflecting FLAvg's behavior, its results stagnate at a low point, suggesting model instability. Possible explanations include the poor performance of the test set from the removed client or that the removed client's contribution negatively impacted prediction capabilities on other client testsets.

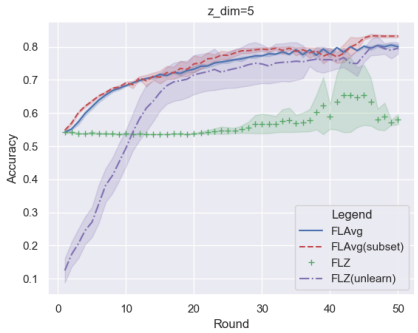
Conclusion: Our exploration underscores the significance of the embedding dimension d in the effectiveness and stability of our contribution model. While some settings, like $d = 5$, yielded promising results, others highlighted potential challenges and areas for further refinement. Continuous tuning and understanding of the contribution model are essential for optimal FL and unlearning processes.

Temperature of Regularization Terms

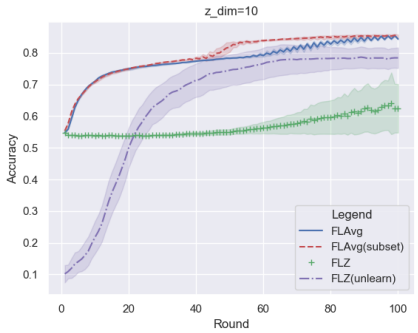
Recall that in Sec. 3.3.2, the concept of temperature in the loss function



(a) Accuracy of Different Learning Frameworks when *z_dim*=2



(b) Accuracy of Different Learning Frameworks when *z_dim*=5



(c) Accuracy of Different Learning Frameworks when *z_dim*=10

Figure 4.5: Sensitivity Study of Dimensionality of Latent Variable **z**.

was introduced to constrain the magnitudes of different loss terms, ensuring that they remain within a similar range. The temperature coefficients, denoted as Reg $\boldsymbol{\theta}$ and Reg $\mathbf{t}$, are pivotal in stabilizing and regulating the model's training process. This section provides an examination of these temperature settings' impact on the model performance, specifically focusing on the model's accuracy.

The experiment sets Reg $\boldsymbol{\theta}$ and Reg $\mathbf{t}$ to 0, 0.5, and 1, respectively, and assesses the system performance through training and testing on the MNIST dataset, maintaining a consistent data distribution as delineated in Tab. 4.4. The unlearning process targets agent 4, and five experimental runs are conducted under each configuration, yielding the mean and standard deviations presented in Tabs. 4.5 and 4.6.

Table 4.5: Accuracy under Different Regularization Temperatures (FLZ)

Reg $\boldsymbol{\theta}$	Reg $\mathbf{t}$		
	0	0.5	1
0	**0.7599(0.0562)**	0.5645(0.0028)	0.5697(0.0164)
0.5	0.7052(0.0528)	0.7480(0.0883)	0.6860(0.0359)
1	0.7003(0.0705)	0.7322(0.0403)	0.7743(0.0399)

Table 4.6: Accuracy under Different Regularization Temperatures (FLZ(unlearn))

Reg $\boldsymbol{\theta}$	Reg $\mathbf{t}$		
	0	0.5	1
0	0.7888(0.0295)	0.8133(0.0207)	0.8125(0.0163)
0.5	0.7990(0.0189)	**0.8259(0.0216)**	0.8088(0.0187)
1	0.7763(0.0410)	0.7480(0.0501)	0.7807(0.0425)

Diving into Tabs. 4.5 and 4.6, clear patterns emerge about the accuracy of both FLZ and FLZ(unlearn) when tweaking the regularization temperatures. FLZ hits its best accuracy, 0.7599, with a standard deviation of 0.0562, when neither regularization coefficients are applied (both set to 0). On the flip side, FLZ(unlearn) sees its top accuracy at 0.8259 (standard deviation of 0.0216) when both Reg $\boldsymbol{\theta}$ and Reg $\mathbf{t}$ are set at 0.5. It's also worth noting that FLZ(unlearn) generally performs better than FLZ, which is in line with

the sensitivity study regarding the dimensionality of latent variable **z**, hinting that the unlearning process could be giving model accuracy a boost.

Random Seeds

In our past tests, we observed that our contribution model reacts differently depending on how it's initially set up with random values. The differences can sometimes be quite noticeable. This experiment is designed to see how changes in random starting points can influence our model's outcomes. For this purpose, we chose the the specific random initialization method and picked distinct random seeds: 235, 172, 51, 179, and 184. The results of this experiment can be seen in Fig. 4.6.

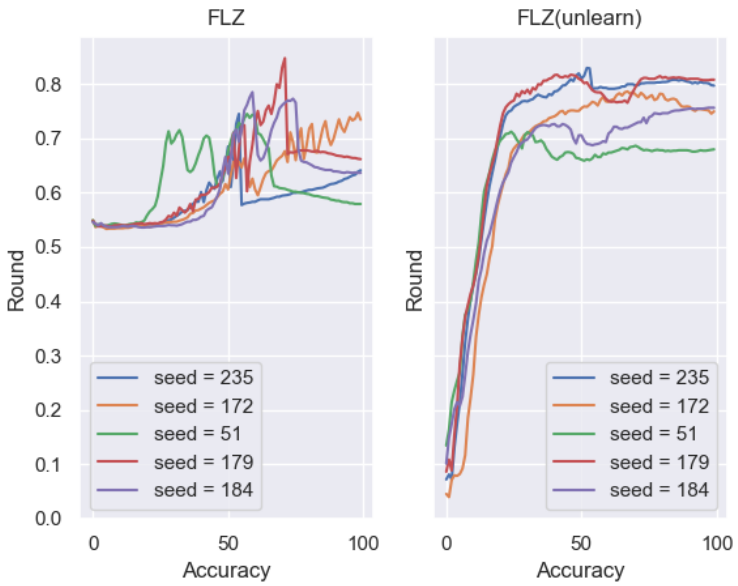


Figure 4.6: Sensitivity Study on Randomness

The graphs show that the model's performance varies quite a bit depending on the seed. For instance, the seed 235 in both FLZ and FLZ(unlearn) seems to have a more stable increase in accuracy compared to other seeds, especially in the initial rounds. Also notice that FLZ(unlearn) generally has a more consistent performance across different seeds compared to FLZ. In the latter rounds, most of the FLZ(unlearn) seeds converge to similar accuracy levels, which is not the case for FLZ. Overall, the initial setup or seed does play a significant role in the model's performance, emphasizing the need to consider and possibly average over multiple initializations for a more robust evaluation.

Chapter 5

Conclusions and Future work

In this final chapter, I wrap up the main points of my thesis, highlight its innovative aspects, discuss its limitations, and suggest directions for upcoming research. In Section 5.1, I provide a summary, revisiting the aims of this study, its strengths and weaknesses, the outcomes, and offer guidance for those interested in pursuing this area further. Section 5.2 addresses the constraints of my approach and the areas where my results might fall short. In Section 5.3, I outline possible paths and questions that future researchers might want to explore within this subject. Lastly, in Section 5.4, I share my reflections on the entire thesis journey.

5.1 Conclusions

This paper brings up the topic of machine unlearning in FL. It introduces the concept of contribution in FL, which quantifies the level of effort each client contributes to the learning of the global model, and once an unlearning request regarding certain client(s) arrive, the server merely subtracts the portion of the target client(s) from the global model. To achieve this, we propose **FLZ**, an unlearning method that learns to disentangle contributions of each client in an FL task. The fundamental idea here is to consider orthogonality as an indicator of disengagement. In an ideal scenario, we would be able to find latent representations of contributions that are mutually orthogonal. However, given the inherent heterogeneity of FL, enforcing orthogonality can potentially lead to a decline in FL task performance. The core component of **FLZ**, the contribution model is employed on the server side, and learns to disentangle the contributions as the FL training progresses, while contribution is modeled as vectors whose summation are real numbers that indicate percentage of

participation of clients. We also customize the aggregation function in FL. Instead of simply averaging local model parameters, our aggregation function takes into account the contributions, aligning with the contribution model. Consisting of reconstruction loss and regularization loss, the overall loss function aims to disentangle contributions while preserve the global model's performance to the most extent. When an unlearning request is received, our unlearning mechanism ensures that data access becomes unnecessary, which contrasts with traditional methods where data access is typically essential.

In our MNIST experiments, we observed that the implementation of **FLZ** effectively increased the orthogonality between latent variables, aligning with our initial expectations. Furthermore, in a series of experiments conducted on both MNIST and FMNIST datasets to explore the impact of data distribution in the training dataset, **FLZ** demonstrated its highest performance levels in both FL and unlearning tasks when clients' data exhibited a slight degree of heterogeneity. The performance achieved under this configuration was notably close to the theoretical upper performance bound. Additionally, a sensitivity analysis revealed that **FLZ** exhibits some degree of sensitivity to specific hyperparameter configurations, an aspect that may warrant further investigation in future work.

5.2 Limitations

This study, while pioneering in its approach to machine unlearning in FL using the **FLZ** method, encounters several limitations. The most apparent constraint is the task domain, narrowly focused on image classification tasks, which raises questions about the model's applicability to other domains. The use of a simple multi-layer perceptron as the prediction model further limits the generalizability of the findings, as this model may not capture the complexity or nuances of more complicated datasets and tasks. Additionally, the nature of training data collection for contribution model in **FLZ**, which accumulates data progressively with each federation round, results in a limited number of training samples. This shortage could reduce the model's ability to learn robust, generalizable patterns and might not represent the diverse scenarios encountered in real-world applications.

Furthermore, the current model design primarily addresses unlearning from a single agent at a time. This aspect is somewhat restrictive, as practical applications often require handling multiple agents simultaneously, particularly in complex federated environments. The sensitivity of **FLZ** to specific hyperparameter configurations is another limitation. It suggests that

optimal performance is highly dependent on precise tuning, which may not be feasible or straightforward in all scenarios. Moreover, the study does not explore the impact of various types of data, such as unstructured or non-image data, which could provide insights into the model's versatility and robustness. The lack of comprehensive testing across different FL settings, such as varying numbers of clients or network topologies, also constitutes a gap in the current research. This limitation points to the need for more extensive experiments to fully understand the capabilities and limitations of the **FLZ** approach.

5.3 Future Work

The current study opens several promising directions for future research in machine unlearning and FL. Expanding the application of the **FLZ** framework beyond image classification to encompass a wider range of tasks, such as regression, natural language processing, or even more complex analytical tasks, could significantly enhance the model's applicability and utility. Investigating the integration of model compression techniques to optimize inputs for the contribution model, especially in scenarios involving large-scale prediction models, is another crucial area. This approach could lead to more efficient and scalable unlearning processes in federated settings.

Modifying the training strategy to incorporate data collected after each epoch rather than each federation round may enrich the dataset and improve the robustness and accuracy of the learning outcomes. Conducting experiments with a larger number of agents, particularly focusing on scenarios involving simultaneous unlearning from multiple agents, would more accurately reflect real-world FL environments. Additionally, exploring different data heterogeneity levels among clients and their impact on the performance of **FLZ** could yield valuable insights, especially in balancing model accuracy with unlearning capabilities.

There is also an opportunity to delve deeper into the effects of various hyperparameter settings on the performance of **FLZ**, potentially leading to the development of adaptive or self-tuning mechanisms that could optimize performance across diverse scenarios. Further, examining the implementation of **FLZ** in different FL architectures, such as hierarchical or decentralized models, could provide a broader understanding of its adaptability and effectiveness. Finally, expanding the research to include more diverse datasets, including non-image and unstructured data, would offer a comprehensive view of the model's capabilities and limitations in various contexts.

5.4 Reflections

This research delves into the realm of Federated Learning (FL) and machine unlearning, a topic of significant relevance in today's data-driven world. From an economic perspective, the ability to unlearn data efficiently in FL models offers a cost-effective solution for businesses. It eliminates the need for resource-intensive retraining processes, thereby reducing computational costs and time. This efficiency is critical in a fast-paced market where data privacy concerns can arise spontaneously.

Socially, our approach to machine unlearning contributes to the increasing need for privacy and data protection. In an era where data breaches and misuse are prevalent, providing a mechanism for data unlearning helps build public trust in machine learning systems. It empowers users with control over their data, aligning with societal demands for greater data sovereignty and ethical data management practices.

Environmentally, the reduced computational requirement for unlearning, as opposed to full model retraining, translates to lower energy consumption. This aspect is crucial in the context of sustainable computing, as it aligns with global efforts to reduce the carbon footprint of large-scale data processing tasks.

Ethically, the ability to unlearn data addresses critical concerns about the right to be forgotten, a fundamental aspect of data privacy regulations like GDPR. Our method enables compliance with such regulations, ensuring that individuals' data can be removed from machine learning models upon request. This aspect not only adheres to legal requirements but also fosters an ethical approach to data handling, respecting individual privacy and data rights.

In summary, this work on machine unlearning in FL not only advances the field technically but also addresses key economic, social, environmental, and ethical dimensions that are increasingly pertinent in the realm of artificial intelligence and machine learning.

References

[1] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models." in *USENIX Security Symposium*, vol. 6, 2021. [Page 2.]

[2] T. Guo, S. Guo, J. Zhang, W. Xu, and J. Wang, "Efficient attribute unlearning: Towards selective removal of input attributes from feature representations," *arXiv preprint arXiv:2202.13295*, 2022. [Pages 2 and 12.]

[3] A. Mantelero, "The eu proposal for a general data protection regulation and the roots of the 'right to be forgotten'," *Computer Law & Security Review*, vol. 29, no. 3, pp. 229–235, 2013. [Pages 2, 5, and 9.]

[4] S. L. Pardau, "The california consumer privacy act: Towards a european-style privacy regime in the united states," *J. Tech. L. & Pol'y*, vol. 23, p. 68, 2018. [Pages 2 and 5.]

[5] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480. [Pages 2, 9, and 14.]

[6] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159. [Pages 2, 9, and 11.]

[7] H. Huang, X. Ma, S. M. Erfani, J. Bailey, and Y. Wang, "Unlearnable examples: Making personal data unexploitable," *arXiv preprint arXiv:2101.04898*, 2021. [Page 2.]

[8] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," *Advances*

in Neural Information Processing Systems, vol. 34, pp. 18 075–18 086, 2021. [Page 2.]

[9] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016. [Page 3.]

[10] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016. [Page 3.]

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013. [Pages 4 and 18.]

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021. [Page 4.]

[13] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "Federaser: Enabling efficient client-level data removal from federated learning models," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 2021, pp. 1–10. [Pages 7 and 18.]

[14] G. Wang, C. X. Dang, and Z. Zhou, "Measure contribution of participants in federated learning," in *2019 IEEE international conference on big data (Big Data)*. IEEE, 2019, pp. 2597–2604. [Page 7.]

[15] A. Halimi, S. Kadhe, A. Rawat, and N. Baracaldo, "Federated unlearning: How to efficiently erase a client in fl?" *arXiv preprint arXiv:2207.05521*, 2022. [Pages 7 and 17.]

[16] C. Wu, S. Zhu, and P. Mitra, "Federated unlearning with knowledge distillation," *arXiv preprint arXiv:2201.09441*, 2022. [Pages 7 and 17.]

[17] T. M. Mitchell and M. Learning, "Mcgraw-hill science," *Engineering/Math*, vol. 1, p. 27, 1997. [Page 9.]

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized

data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282. [Page 9.]

[19] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019. [Page 10.]

[20] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," *arXiv preprint arXiv:2209.02299*, 2022. [Page 10.]

[21] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational bayesian unlearning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 025–16 036, 2020. [Page 11.]

[22] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck, "Machine unlearning of features and labels," *arXiv preprint arXiv:2108.11577*, 2021. [Page 12.]

[23] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Fast yet effective machine unlearning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023. [Pages 13 and 14.]

[24] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, "Zero-shot machine unlearning," *IEEE Transactions on Information Forensics and Security*, 2023. [Page 14.]

[25] A. Thudi, H. Jia, I. Shumailov, and N. Papernot, "On the necessity of auditable algorithmic definitions for machine unlearning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4007–4022. [Page 14.]

[26] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2008–2016. [Page 14.]

[27] M. Jagielski, O. Thakkar, F. Tramer, D. Ippolito, K. Lee, N. Carlini, E. Wallace, S. Song, A. Thakurta, N. Papernot *et al.*, "Measuring forgetting of memorized training examples," *arXiv preprint arXiv:2207.00099*, 2022. [Page 15.]

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Page 32.]

[29] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017. [Page 32.]