Semi-Supervised Federated Learning over Relevant Heterogeneous Data

Tahani Aladwani
tahani.aladwani@glasgow.ac.uk
Christos Anagnostopoulos
christos.anagnostopoulos@glasgow.ac.uk
School of Computing Science
University of Glasgow
Glasgow, UK

Abstract

Federated Learning (FL) aims to collaboratively train models without *clients'* data transfer. However, FL faces significant challenges, particularly concerning model performance, which heavily relies on the quality of clients' data. Clients' data quality varies due to factors like data heterogeneity and labeling strategies across clients. Not all clients have ground truth labels; client data are often unlabeled, partially labeled, or labeled with sub-optimal quality due to labeling cost (time and resources) or insufficient expertise. This paper introduces a novel approach based on Semi-Supervised FL (SSFL), which employs a Multi-Purpose pre-trained SSFL model (MP-SSFL) to leverage ground truth labels from server data for training a global pseudo-labeling model. Such model is, in turn, utilized for label prediction and distributed data selection across clients while ensuring privacy preservation. Our MP-SSFL method enables labeling of unlabeled data, correction of inaccurately labeled data, and mitigation of label overlapping issues. Compared against benchmarks, MP-SSFL significantly enhances the quality of pseudo-labels, thereby leading to improved performance of global classifier models. Such improvement is notable in cases involving non-identically labeled data across clients with non-independently and identically distributed data.

1 Introduction

With the advancement of Edge Computing, distributed devices like smartphones and vehicles have become capable of collecting and storing various forms of data including images, voice, and text [18]. As a result, edge devices are considered valuable sources of diverse data, possessing great potential to train Machine Learning (ML) models with higher accuracy and generalization compared to training over data from one source [6, 11].

However, bringing all these data into a central server is not feasible and practical due to clients' data ownership, privacy and volume [16]. Therefore, Federated Learning (FL) has been proposed as a distributed ML paradigm that aims to train high-quality global model collaboratively across clients (devices) to facilitate access to more data variety over disjoint data spaces while leveraging their computing resources to model training [4] [9]. Nonetheless, not all clients' data can improve the global model quality [1] [8]. This depends on the amount of data in each client, data quality, labels' availability, and data relevance between predictive analytics tasks and what data the clients have available [4].

Therefore, before deploying FL, we address the question: *How can we effectively mitigate the impact of data heterogeneity across clients on the overall accuracy of a global model, without disregarding important data subsets, especially in clients with many irrelevant data?* This question underscores the need to balance the integration of diverse client data while ensuring the preservation of data quality and relevance. Furthermore, FL-based tasks become significantly complex in real-world scenarios where labeling inconsistent strategies among clients need to be considered. These issues include differences in labeling experiences across clients, presence of partially labeled or unlabeled clients' data, and potentially attacked clients. Addressing such labeling challenges is essential for ensuring the robustness and reliability of supervised FL models [7]. Furthermore, clients may have a high rate of irrelevant samples that significantly deteriorates the final global model performance if they are considered in the training. Meanwhile, it is not trivial to determine a subset of clients with perfect data for a given task. Each client may consist of sets of irrelevant samples and/or unlabeled samples. In traditional FL used for supervised learning (SL), such clients cannot participate in the training process. However, ignoring these clients completely makes the global model loose the chance to train upon diverse and real-world representative data.

To address the challenge of handling irrelevant and unlabeled samples, we propose a Self-Supervised Learning (SSL) paradigm that builds a pre-trained model on the FL server. Then, this model is used to predict the sample labels for each client. We consider a sample *relevant* if only if the model generates pseudo-labels with high confidence. By employing SSL in FL, we leverage the limited labeled data available on the server to create a supervised global pseudo-labeling model. Such model is used to label clients' samples and extract knowledge from unlabeled data [13] [17], [19]. The main contributions of this paper are:

- A novel FL-based knowledge transfer mechanism that addresses the sample labeling challenge over large amounts of unlabeled samples, or untrustworthy labeled samples (samples labeled by non-expert clients or attacked clients and label flipping across clients).
- A novel probabilistic data selection mechanism based on the loss of the sample predicted pseudo-labeling.
- Comprehensive experimental evaluation and comparative assessment over widely-used benchmark datasets showcasing the effectiveness of our label prediction and data selection mechanisms. Our mechanism, coined **Multi-purpose Self-Supervised FL (MP-SSFL)**, accurately predicts pseudo-labels and efficiently filters out irrelevant samples, including those with the same label due to label overlapping.

2 Related Work & Background

There are three different learning paradigms in ML that deal with various types of data. Supervised learning (SL) is used to make predictions with labeled data. SL is widely studied in the field of FL, e.g., [14], [10], [18], while the SSL paradigm deals with partially labeled data or limited labeled data. SSL has drawn more attention in recent studies that focus on non-IID data and unlabeled samples in clients, e.g., [19], [5],[17],[12], [2],[13],[15], [20],[7]. In this work, we focus on SSL to tackle clients' labeling issues. Most studies focused on fixing one labeling problem by using SSL either by assuming clients have unlabeled data

or partially labeled data. In case of unlabeled clients, they based on a server SL model to label clients' data by employing various techniques like pseudo-labeling or teacher-student models. Some of these works use CNNs to build label prediction models. However, in distributed learning environments with heterogeneous devices, we have wider issues than just unlabeled samples across a set of clients. We meet clients with less labeling experience and knowledge, and clients with non-identically labeled samples. Such issues degrade the global model's performance. Real-world scenarios often involve these challenges when training a FL model for analytics tasks. Therefore, we adapted a multi-task SSL model to deal with these issues and bridge the gap between existing works and the issues associated with sample labeling.

2.1 Federated Learning

Consider a set $\mathcal{N}$ of distributed clients, $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$, each one having its local dataset D_k comprising m labeled samples $(X_k, Y_k) \in \mathcal{D}_k$. At each communication round in FL $t \in [\mathcal{T}]$, a subset of clients $N \subseteq \mathcal{N}$ is randomly selected by the server $\mathcal{S}$ to train a global model θ_{global} [20]. Each selected client $n_k \in N$ trains its local model $f = h(\theta_g, \mathcal{X})$ on samples $(X_k, Y_k) \in D_k$ over E epochs, updating the model parameters using stochastic gradient descent (SGD). The local loss for each client n_k is defined as: $\mathcal{L}_k(\theta) = \frac{1}{D_k} \sum_{(X_i, y_i) \in D_k} \mathcal{L}(f(X_i, \theta), y_i)$, where $\mathcal{L}$ is the loss function. After training, each client n_k sends its updated parameters to the server $\mathcal{S}$, which aggregates them to update the global model parameters θ_g such that: $\theta_g^{t+1} = \frac{1}{|N_t|} \sum_{n \in N} \theta_n^t$. The goal is to minimize the global loss $\mathcal{L}_g(\theta)$ across all clients, defined as $\mathcal{L}_g(\theta) = \frac{\sum_{n=1}^{\mathcal{N}} (D_n) | \mathcal{L}_n(\theta)}{\sum_{n=1}^{\mathcal{N}} |D_n|}$. The objective is to find the model parameters $\hat{\theta}$ that minimize $\mathcal{L}_g$, i.e., $\hat{\theta} = \arg\min_{\theta} \mathcal{L}_g(\theta)$. The updated global parameters θ_g^{t+1} are then distributed to a new set of labeled clients $N \in \mathcal{N}$ for updating their local models [3].

2.2 Irrelevant Data Filtering

We present two related approaches on filtering out irrelevant data and we adopt them as baseline methods.

Baseline 1: In [10], the selection of relevant samples is based on the probability of a sample being relevant. This probability is determined based on the output of function g_i, which yields a value between $[0,1]$. Each client $n_i \in \mathcal{N}$ learns a local relevance prediction function (RDSi) to predict the Relevance Score (RS) for each sample $(x_i, y_i) \in n_i$, where $RS \in [0,1]$. The focus is on prioritizing high values of $g_i(x_i, y_i)$ to find the best local model parameter $\hat{\theta}_i$ that is closer to the global model parameter θ_g aiming to reduce loss on unseen test samples.

Baseline 2: In [14], the authors have proposed a data relevant selection mechanism based on a model built based on the server data. They assumed the server builds a Relevant Selection Model (RSM). Then, RSM is sent to a set of clients, and each client calculates the losses for each sample and sends the loss back to the server. The server merges these losses from all clients and uses them to calculate a filtering threshold. Finally, the server sends this threshold filter to each client to filter out irrelevant samples in their own local data. However, this method is not secure as it requires a loss for each sample, which is considered data disclosure.

Both baselines are based significantly on the truth of labels $\mathcal{Y}$ to make the relevance sample decision. This means that they filter out attacked samples (label flipping attacks), unlabeled samples, and non-identically labeled samples.

3 The MP-SSFL Framework

3.1 Problem Fundamentals

A typical FL assumes that the labels on clients' datasets $D_n \in \mathcal{N}$ and server's dataset $D_\mathcal{S}$ are identical, while all have the ground truth labels. In our case, instead of considering identical labeled datasets $D_k \in D_\mathcal{N}$, we formulate a problem setting with non-IID datasets D_k.

Assumption 1 (non-identically labeled clients):
In our framework, we consider the possibility that each client n_k may exhibit a different label distribution $\mathbf{Y}$ compared to the server $\mathcal{S}$, denoted as $P(Y_k) \neq P(Y_\mathcal{S})$ [7]. If $\mathcal{S}$ has K classes, then $Y_\mathcal{S} = [K]$. Additionally, $Y_k \neq Y_\mathcal{S}$ for each client $n_i \in \mathcal{N}$, indicating that the label sets may differ between clients and the server. Each local dataset $\mathcal{D}_k$ may contain labels ranging in $[\emptyset, 1, 2, \ldots, \mathbf{L}]$, where $\emptyset$ represents samples without corresponding labels, and $\mathbf{L}$ is the last label in $\mathcal{D}_k$. Thus, we obtain that $|D_\mathcal{S}| \leq |D_\mathcal{N}|$ while ground truth labels are exclusively available at the server $\mathcal{S}$ [2]. Furthermore, while $X_\mathcal{S}$ and $X_{n_i} \in \mathcal{N}$ may share a high degree of similarity, denoted as $P(n_i(X)) \geq P(\mathcal{S}(X))$, it is essential to acknowledge that even if Y_i and $Y_\mathcal{S}$ have the same number of labels and utilize the same labeling mechanism, the labels cannot be considered entirely trustworthy. In summary, we aim to solve the following three main labels' issues across clients in FL by adopting a multi-purpose SSFL model:

1. Unlabeled samples on clients,

$$D_k = \{X_k\}_{x=1}^{|\mathcal{X}|} \in \mathcal{U}, \tag{1}$$

   where $\mathcal{U}$ refers to unlabeled data.

2. Client n_i has more labels than those on the server $\mathcal{S}$ with different labels distributions,

$$|Y_{n_i}| > |Y_\mathcal{S}|, \tag{2}$$

3. Client n_i and $\mathcal{S}$ have the same number of labels and labeling experience, but the client n_i has *poisoned* data as a result of labels flipping attacks:

$$\begin{aligned} &\{X_\mathcal{S}, Y_\mathcal{S}\} \in Y \\ &\{X_{n_i}, Y_{n_i}\} \in (\overline{Y}, Y), \end{aligned} \tag{3}$$

   where Y is the ground truth labels and $\overline{Y}$ is the flipped labels over client n_i.

Assumption 2 (relevant and irrelevant samples):
After fixing the labeling problem, each sample will present in the form (x_i, y_i), while y_i could be real or pseudo-label as we will elaborate later. As mentioned in Assumption 1, $\mathcal{S}$ and $(n_i \in \mathcal{N})$ have similar distributions but not identical. That means $n_i \in \mathcal{N}$ could have relevant and irrelevant data samples for a given task on the $\mathcal{S}$. Each node $n_i \in \mathcal{N}$ has $(D_k = \mathbb{F}_{n_i} \cup \mathcal{I}_{n_i})$, where $\mathbb{F} \subseteq D_k$ is the relevant training samples in n_k and $\mathcal{I} \subseteq D_k$ is the irrelevant training samples $\mathcal{I} \subseteq D_k$. These irrelevant samples could be a result of labels overlapping. Our goal is to train a global classification model θ_{global} collaboratively over a set of clients $n_k \in \mathcal{M}$ with respect to non-IID $D_k \in D_\mathcal{N}$ and data heterogeneity, that makes $D_k \in D_\mathcal{N}$ including relevant and irrelevant samples.

3.2 Methodology

In this subsection, we present our solutions to address the problems mentioned above. We introduce our pseudo-labeling model (classifier model *CM*), which is established through SSFL utilizing a pre-trained model that has been built using the server dataset $D_{\mathcal{S}}$. This model *CM* is used to pseudo-labeling $\hat{y}$ the unlabeled samples $X \in \mathcal{U}$ to leverage the unlabeled data $\{X_k\}_{x=1}^{|\mathcal{X}|} \in \mathcal{U}$ across clients $n_i \in \mathcal{N}$ that share almost the same distribution as labeled data over the server $\mathcal{S}$ and filter out the irrelevant data (samples out of the server data distribution). In addition, it is used to label the non-trustworthy samples according to specific factors (e.g., each sample loss $\mathcal{L}$, $g(x)$), as we will elaborate in this section. This method forms a stronger generalization in the direction of exploiting the relevant samples and provide more meaningful pseudo-labels for unlabeled and non-trustworthy labeled clients. The FL model is trained on both labeled and unlabeled data based on the following:

$$\min_{\theta} f_{AT}(\theta) = \frac{1}{|D_{\mathcal{S}}|} \sum_{i=1}^{|D_{\mathcal{S}}|} \ell_s(X_i, y_i; \theta_{\mathcal{S}}) + \frac{1}{|\mathcal{N}|} \sum_{n=1}^{|\mathcal{N}|} \sum_{j=1}^{|n_k \in \mathcal{N}|} \ell_{n_k}(X_j; \theta_{\mathcal{S}}), \tag{4}$$

where *AT* points to the analytic task mode as we will elaborate later. There are three steps for our samples labeling and filtering mechanism: The SL stage on the server $\mathcal{S}$, unsupervised learning (UL) on clients (clients samples pseudo-labeling), and irrelevant samples filtering out.

Stage1: SL at the Server The server $\mathcal{S}$ builds the *CM* model based on its labeled samples $(D_{\mathcal{S}})$ to get $(X_{\mathcal{S}}, Y_{\mathcal{S}}) \rightarrow f(X_{\mathcal{S}}, \theta_{\mathcal{S}})$. This model is distributed to $n_i \in \mathcal{N}$ in order to label their data according to it.

Step2: Clients' samples Pseudo-labeling: Each client $n_i \in \mathcal{N}$ retrieves the $f(.;\theta_{\mathcal{S}})$ and applies it to its own data, to predict a pseudo-label $\hat{y}_i$ for each $x_{i \in ni}$. Here, $\hat{y}_i$ is the class with the highest predicted probability, according to:

$$\hat{y}_i = arg\max_{c} g_{\theta_{\mathcal{S}}}(x_i)_c. \tag{5}$$

In this context, $g_{\theta_{\mathcal{S}}}$ refers to the model parameterized by $g_{\theta_{\mathcal{S}}}$, and $g_{\theta_{\mathcal{S}}}(x_i)c$ represents the prediction probability of class c for the input x_i, where $c \in \mathbf{C}$ represents one of the classes within the set $\mathbf{C}$. However, leveraging the pseudo label $\hat{y}$ varies between unlabeled data and non-trustworthy labeled samples. The common practice between all these labeling issues is all of them based on a confidence threshold (hard threshold) $\tau \in [0.5, 1]$ for the pseudo label probability, i.e.,

$$\hat{y_i} = \begin{cases} 1 & g_{\theta_{\mathcal{S}}}(x_i)c > \tau \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

For the **unlabeled data**, our framework considers the label that satisfies the τ condition in (6). According to **non-trustworthy labels**, we aim to use the $\hat{y}$ but without completely ignore labels Y_{n_i}. In other word, without completely blur samples' identity. Because in some cases *CM* model could not learned enough with some classes, this results in *CM* making mistakes in labeling some samples X_i. The important question now becomes: Consider a client $n_i \in \mathcal{N}$ with the sample (x_i, y_i) and $P(y_{n_i}) \neq P(y_{\mathcal{S}})$. If $f(.;\theta_{\mathcal{S}})$ has predicted a different label, which means $y_i \neq \hat{y}$, which label should we consider: the main label y_i or the predicted label $\hat{y}$ based on the SL?. In this case, we need two factors to build the decision function.

The first one is $g(x_i) \rightarrow [0.5, 1]$ and the second one is the sample's loss $\mathcal{L}(f(x_i, \theta_{\mathcal{S}}))$ that has Υ as a tolerance threshold. This means that $\mathcal{L}$ only considers low and acceptable if $\mathcal{L} < \Upsilon$. Based on these factors, we obtain four cases to decide whether to replace the actual labely_i to the pseudo-label $\hat{y}$ ($y_i \leftarrow \hat{y}$) or not. According to Table (1), we consider the cases:

Rate	High	Low
High	$(\mathcal{L}, g(x))$	$(\mathcal{L}, g(x))$
Low	$(\mathcal{L}, g(x))$	$(\mathcal{L}, g(x))$

Table 1: Label allocation.

1. $(\mathcal{L}, g(x_i))$ are both high. This means that the predicted label $\hat{y} \neq y_i$ and the difference between them is not acceptable (i.e., y_i is *far* from $\hat{y}$), while the *CM* model has predicted x_i with high probability $g(x_i)$. In this case we consider the client n_i either had attacked (label flipping attack), or had no enough experience in data labeling, or had a different label distribution. Therefore, we cannot consider the real label y_i thus we replace y_i by $\hat{y}$ as in (7), which is considered as a relevant sample:

$$\text{If } \mathcal{L}(x_i) > \Upsilon \text{ and } g(x_i) \geq \tau \quad \rightarrow \quad (x_i, \hat{y}) \tag{7}$$

2. $\mathcal{L}$ is high and $g(x_i)$ is low. We assume that this is an irrelevant sample (outside the server data distribution). Consequently, the *CM* model is not confident about this sample. Therefore, the model will not change this sample's label and it is considered as an irrelevant sample in (8):

$$\text{If } \mathcal{L}(x_i) > \Upsilon \text{ and } g(x_i) < \tau \quad \rightarrow \quad (x_i, y_i) \text{ (irrelevant)} \tag{8}$$

3. $\mathcal{L}$ is low and $g(x_i)$ is high. This is the best case, since there is a high percent of agreement between y_i and $\hat{y}$. Therefore, we keep the sample x_i, y_i as it is, and it is considered as a relevant sample.

4. $\mathcal{L}$ is low and $g(x_i)$ is low. This mean that the sample tends to be irrelevant, since the *CM* model is not confident about the right label, even if the label that its predicted close to the real one. It still under τ. In this case we consider this sample as an irrelevant one as in (9):

$$\text{If } \mathcal{L}(x_i) \leq \Upsilon \text{ and } g(x_i) < \tau \quad \rightarrow \quad (x_i, y_i) \text{ (irrelevant)} \tag{9}$$

Step3: Irrelevant data filtering: After applying data labeling mechanism as in **Step 2**, each client $n_i \in \mathcal{N}$ has a clear idea which set of samples belongs to $\mathbb{F} \subseteq D_k$ and which set of samples belongs to $\mathcal{I} \subseteq D_k$. Then, it will train the *AT* model on $\mathbb{F} \subseteq D_k$. We define the loss for each client $n_k \in N$ with a collection of data samples belongs to $\mathbb{F} \subseteq D_k$ to minimize the objective function (10):

$$\mathcal{L}_k(\theta) = \frac{1}{\mathbb{F} \subseteq D_k} \sum_{(X_i, y_i) \in \mathbb{F} \subseteq D_k} \mathcal{L}(f(X_i, \theta), y_i). \tag{10}$$

Then, node $n_k \in N$ sends a parameter's update to the server $\mathcal{S}$ based on $\mathbb{F} \subseteq D_k$.

Algorithm 1 MP-SSFL Process

Input: Global model $f(.,\theta)$ **Output**: Labeled data samples

for Each client $n_i \in \mathcal{N}$ receives CM $f(.,\theta)$. **do**
  for Each (x_i, y_i) in n_i utilizes $f(.,\theta)$ to calculate $\mathcal{L}(f(x_i,y_i),\hat{y})$ and $g(x)$. **do**
    Calculate decision function ($\mathcal{L}(f(x_i,y_i),\hat{y})$ and $g(x)$).
    Use decision function based on $\mathcal{L}$ and $g(x)$ to determine the right labels.
    if (Low ($\mathcal{L}$) and High($g(x_i)$)) **then**
      $y_i \leftarrow \hat{y}$ (Relevant sample)
    else if (High ($\mathcal{L}$) and Low($g(x_i)$)) **then**
      $y_i \leftarrow y_i$ (Irrelevant sample)
    else if (Low ($\mathcal{L}$) and low($g(x_i)$)) **then**
      $y_i \leftarrow y_i$ (Irrelevant sample)
    else if (High ($\mathcal{L}$) and High($g(x_i)$)) **then**
      $y_i \leftarrow \hat{y}$ (Relevant sample)
    end if
  end for
end for=0

4 Experimental Evaluation

4.1 Experimental Setup

We evaluate the effectiveness of MP-SSFL using benchmark datasets MNIST and Fashion-MNIST. Our study addresses prevalent data labeling issues across different user scenarios, considering varying numbers of users ($\mathcal{N}$) ranging from 10 to 100 and sample sizes of 200, 500, 1000, and 5000. We employ CNNs for both the labeling and analytic task models. The setup includes comparisons with baselines to assess the performance of our proposed mechanism. Specifically, we obtain setups for each labeling issue:

- **Unlabeled Clients**: We initially considered unlabeled samples across clients, with ground-truth labels known to the server. Varying dataset sizes were examined on the server, denoted as $D_{\mathcal{S}} \in \{90, 240, 500, 1000\}$, to assess how the quantity of samples utilized in the pre-trained model could aid in labeling unlabeled data across clients. We then compared the pseudo-labels generated by our MP-SSFL approach with those from the baselines. Additionally, we employed client datasets pseudo-labeled according to our pre-trained model to construct an FedAvg model.

- **Labels Overlapping**: Initially, we conducted an analytical task using the MNIST dataset. Each client's dataset contained a mix of MNIST and FMNIST samples, with varying proportions (e.g., 200:200, 200:150, 200:100) for each user ($n_k \in [10, 50, 100]$). We then increased the sample size ($200 \rightarrow 500 \rightarrow 1000$). Utilizing a CM built on 5000 samples, we filtered out irrelevant samples to mitigate their impact on FedAvg. Subsequently, we reversed the scenario, focusing on an analytical task based on the FMNIST dataset while treating MNIST data as irrelevant samples due to label overlapping. We replicated the same steps as in the MNIST analytical task, as detailed further in the results section.

- **Labels Flipping Attack Detection**: In this phase of the experiment, we considered clients $n_k \in \mathcal{N}$ to initially have the same labels as the server $\mathcal{S}$, but they could be

vulnerable to a flipping attack. To address this labeling issue, we employed a pre-trained CNN based on Eq. (7). We conducted experiments using various attack rates of $[20\%, 50\%, 75\%, 100\%]$ for both MNIST and F-MNIST datasets. Then, comparing it to the baselines how they treated this kind of samples.

4.2 Performance Evaluation & Comparative Assessment

4.2.1 Label prediction for unlabeled datasets

In this section, we evaluate the effectiveness of the proposed data labeling mechanism by constructing *CM* models using a limited number of samples (e.g., 90, 240, 500, 1000). As presented in Table **??**, we compare the average accuracy of correctly predicted labels across (10, 50, 100) clients, each with varying amounts of unlabeled samples (200, 500, 1000), based on our mechanism against the optimal solution (the actual labels). It is evident that the model achieved an accuracy of over 60% in correctly predicting labels with different sample sizes from diverse datasets A main question that could arise is *how these predicted label*

Table 2: Accuracy of FedAvg using labels predicted by MP-SSFL compared to perfectly labeled clients and baselines' labels; MNIST.

Clients	Samples	Models accuracy			
		Perfect Labels	MP-SSFL	Baseline1	Baseline2
10 clients	200 sample	69.49%	68.11%	61.59%	63.36%
	500 sample	70.04%	70.99%	65.57%	66.91%
	1000 sample	67.58%	66.82%	62.76%	61.46%
50 clients	200 sample	69.55%	68.48%	62.86%	62.32%
	500 sample	69.27%	68.79%	64.19%	63.86%
	1000 sample	65.22%	64.82%	61.37%	59.82%
100 clients	200 sample	67.57%	66.20%	65.55%	65.67%
	500 sample	69.63%	68.87%	62.37%	64.34%
	1000 sample	69.63%	69.45%	63.04%	63.66%

samples could help improve FedAvg performance? The answer of this question is in Table 3. As we can see, building FL models based on samples that have been labeled according to MP-SSFL mechanism yields accuracy very close to models that have been built based on samples with real labels.

Table 3: Accuracy of FedAvg using labels predicted by MP-SSFL compared to perfectly labeled clients and baselines' labels; MNIST.

Clients	Samples	Models accuracy			
		Perfect Labels	MP-SSFL	Baseline1	Baseline2
10 clients	200 sample	69.49%	68.11%	61.59%	63.36%
	500 sample	70.04%	70.99%	65.57%	66.91%
	1000 sample	67.58%	66.82%	62.76%	61.46%
50 clients	200 sample	69.55%	68.48%	62.86%	62.32%
	500 sample	69.27%	68.79%	64.19%	63.86%
	1000 sample	65.22%	64.82%	61.37%	59.82%
100 clients	200 sample	67.57%	66.20%	65.55%	65.67%
	500 sample	69.63%	68.87%	62.37%	64.34%
	1000 sample	69.63%	69.45%	63.04%	63.66%

4.2.2 Impact of Label Flipping

We assumed the following flipping rates (200:20, 200:50, 200:100). Our mechanism exhibits nearly the same accuracy as the baselines in terms of label flipping detection. The

key question here is: **How does each mechanism handle these attacked samples?** In the case of the baselines, they consider the attacked samples as noise data and filter them out. However, in our MP-SSFL, we aim to rectify them. Rather than, loss o(20%, 25%, 50%) of data based on the assumed attack rates. Consequently, **how does this impact the performance of the *AT* model**? The answer to this question can be found in Table 4, which shows that removing the attacked samples instead of rectifying them leads to a decrease in model performance. This effect becomes more clear when the number of attacked samples exceeds that of non-attacked samples.

Table 4: Accuracy of baselines with MP-SSF

Dataset	Clients	Required Sample	Accuracy (%)		
			MP-SSF	Baseline1	Baseline2
MNIST	**10 clients**	200+(200 irrelevant)	94.28	52.01	87.55
		200+(100 irrelevant)	96.95	68.25	91.97
		200+(50 irrelevant)	98.33	81.31	95.91
	50 clients	200+(200 irrelevant)	93.98	49.58	83.50
		200+(100 irrelevant)	95.30	66.30	85.15
		200+(50 irrelevant)	97.31	79.75	91.77
	100 clients	200+(200 irrelevant)	91.30	51.96	84.89
		200+(100 irrelevant)	95.57	68.35	91.80
		200+(50 irrelevant)	97.55	81.20	95.35
F-mnist	**10 clients**	200+(200 irrelevant)	88.21	50.41	85.87
		200+(100 irrelevant)	94.42	66.96	92.51
		200+(50 irrelevant)	96.97	80.14	95.45
	50 clients	200+(200 irrelevant)	88.13	50.42	87.32
		200+(100 irrelevant)	93.83	67.01	93.38
		200+(50 irrelevant)	96.99	80.26	96.68
	100 clients	200+(200 irrelevant)	88.28	50.41	87.67
		200+(100 irrelevant)	93.73	67.02	93.04
		200+(50 irrelevant)	96.80	80.23	96.32

5 Conclusions

We propose the MP-SSFL framework that addresses labeling issues across clients in FL over non-IID data. MP-SSFL efficiently tackles unlabeled data, and non-IID labeled data without requiring modifications to the model for each labeling issue individually. Through extensive experiments using common datasets that incorporate these labeling issues, with variations in the number of users, samples, and the balance between relevant and irrelevant data, as well as different scenarios of label flipping attacks and non-identically labeled clients, we demonstrate the effectiveness of MP-SSFL compared to baselines. The results showcase that MP-SSFL outperforms the baselines w.r.t. accuracy, reliability, robustness, and ability to address multiple issues simultaneously.

References

[1] Aiguo Chen, Yang Fu, Zexin Sha, and Guoming Lu. An emd-based adaptive client selection algorithm for federated learning in heterogeneous data scenarios. *Frontiers in Plant Science*, 13, 2022.

[2] Siwei Feng, Boyang Li, Han Yu, Yang Liu, and Qiang Yang. Semi-supervised federated heterogeneous transfer learning. *Knowledge-Based Systems*, 252:109384, 2022.

[3] Martin Isaksson, Edvin Listo Zec, Rickard Cöster, Daniel Gillblad, and Sarunas Girdzijauskas. Adaptive expert models for federated learning. In *International Workshop on Trustworthy Federated Learning*, pages 1–16. Springer, 2022.

[4] Jiawei Jiang, Lukas Burkhalter, Fangcheng Fu, Bolin Ding, Bo Du, Anwar Hithnawi, Bo Li, and Ce Zhang. Vf-ps: How to select important participants in vertical federated learning, efficiently and securely? In *Advances in Neural Information Processing Systems*.

[5] Yilun Jin, Yang Liu, Kai Chen, and Qiang Yang. Federated learning without full labels: A survey. *arXiv preprint arXiv:2303.14453*, 2023.

[6] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pages 19–35, 2021.

[7] Cobbinah B Mawuli, Jay Kumar, Ebenezer Nanor, Shangxuan Fu, Liangxu Pan, Qinli Yang, Wei Zhang, and Junming Shao. Semi-supervised federated learning on evolving data streams. *Information Sciences*, page 119235, 2023.

[8] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[9] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[10] Lokesh Nagalapatti, Ruhi Sharma Mittal, and Ramasuri Narayanam. Is your data relevant?: Dynamic selection of relevant data for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7859–7867, 2022.

[11] TV Nguyen, MA Dakka, SM Diakiw, MD VerMilyea, M Perugini, JMM Hall, and D Perugini. A novel decentralized federated learning approach to train on globally distributed, poor quality, and protected private medical data. *Scientific Reports*, 12(1): 8888, 2022.

[12] Xinjun Pei, Xiaoheng Deng, Shengwei Tian, Lan Zhang, and Kaiping Xue. A knowledge transfer-based semi-supervised federated learning for iot malware detection. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[13] Liang Qiu, Jierong Cheng, Huxin Gao, Wei Xiong, and Hongliang Ren. Federated semi-supervised learning for medical image segmentation via pseudo-label denoising. *IEEE Journal of Biomedical and Health Informatics*, 2023.

[14] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K Leung. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5020–5027. IEEE, 2021.

[15] Shuai Wang, Yanqing Xu, Yanli Yuan, and Tony QS Quek. Towards fast personalized semi-supervised federated learning in edge networks: Algorithm design and theoretical guarantee. *IEEE Transactions on Wireless Communications*, 2023.

[16] Hongda Wu and Ping Wang. Probabilistic node selection for federated learning with heterogeneous data in mobile edge. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2453–2458. IEEE, 2022.

[17] Shuyi Yang, Mattia Cerrato, Dino Ienco, Ruggero G Pensa, and Roberto Esposito. Fair swirl: fair semi-supervised classification with representation learning. *Machine Learning*, pages 1–26, 2023.

[18] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. Fedmix: Approximation of mixup under mean augmented federated learning. *arXiv preprint arXiv:2107.00233*, 2021.

[19] Zhengyi Zhong, Ji Wang, Weidong Bao, Jingxuan Zhou, Xiaomin Zhu, and Xiongtao Zhang. Semi-hfl: semi-supervised federated learning for heterogeneous devices. *Complex & Intelligent Systems*, 9(2):1995–2017, 2023.

[20] Yuanshao Zhu, Yi Liu, JQ James, and Xingliang Yuan. Semi-supervised federated learning for travel mode identification from gps trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2380–2391, 2021.