Master's Thesis of Data Science

FedSup: Teacher-Student Architecture for Federated Learning with Unlabeled Clients

FedSup: 교사-학생 구조 준지도 연합학습

February 2023

Graduate School of Data Science
Seoul National University
Data Science Major

KWANGYEON GILL

Master's Thesis of Data Science

FedSup: Teacher-Student Architecture for Federated Learning with Unlabeled Clients

FedSup: 교사-학생 구조 준지도 연합학습

February 2023

Graduate School of Data Science
Seoul National University
Data Science Major

KWANGYEON GILL

FedSup: Teacher-Student Architecture for Federated Learning with Unlabeled Clients

Hyung Sin Kim

Submitting a master's thesis of
Data Science

December 2022

Graduate School of Data Science
Seoul National University
Data Science Major

KWANGYEON GILL

Confirming the master's thesis written by
KWANGYEON GILL
January 2023

Chair _____(Seal)
Vice Chair _____(Seal)
Examiner _____(Seal)

Abstract

Federated Learning (FL) is a machine learning paradigm in which multiple heterogeneous clients train local models with their data and only share the parameters to the server to create a centralized model. This paradigm, however, is based upon an unrealistic assumption that every client has fully labeled data readily available for training. Since labeling the data generally requires domain expertise and consistency, which are difficult to attain in a federated setup, it is more pragmatic to consider a scenario where clients own completely unlabeled data, whereas the server contains a small fraction of labeled data ("Labels-At-Server")[20]. The methods to exploit unlabeled data at clients are actively being researched, which takes advantage of stochastic augmentations to improve the quality of pseudo-labels. Inspired by recent SSL methods and knowledge distillation, we propose a Semi-Supervised FL teacher-student architecture *FedSup* to tackle this problem. To demonstrate its validity, we conduct various experiments on CIFAR-10/CIFAR-100/STL-10 using naive applications of four popular SSL methods to FL and state-of-the-art Semi-Supervised FL methods, FedMatch and FedRGD. On both Independent and identically distributed (IID) and non-IID data, *FedSup* demonstrates higher accuracy on all three datasets compared to other methods under finetuning. Also, we conduct ablation studies on CIFAR-10 to explore why *FedSup* works better.

keywords: Federated Learning, Semi-Supervised Learning

student number: 2021-26031

Contents

Chapter 1

Introduction

Large-scale data is one of the most important factors in training a deep learning model; nevertheless, data are distributed across different places in practice, and thus have to be transferred to a centralized server for training, which can cause violation of privacy [13] and higher communication cost. FL handles this problem by building multiple local models at the distributed places, or clients, using their computing resources and aggregating the parameters at the server [23] [34], without sending the local data to the server.

One of the key challenges that FL faces is that many parties hold data of heterogeneous class distributions. Previous studies such as FedProx [37], FedNova [40], SCAFFOLD [21] design clients to learn effectively from non-IID data, i.e. data heterogeneity. However, a more critical challenge is the scarcity of labels; FL inevitably suffers from label deficiency, as the parties involved in FL may not be able to provide reliable labels for their own data. FedEMA [47] tackles the lack of labels at clients by applying label-agnostic SSL at the clients, such as BYOL [14], SimCLR [5] and SimSiam [8], while FedMatch [20] and FedRGD [45] seek to improve pseudo-labeling at the clients. However, as these SSL methods are proposed for centralized training, naive combinations are not as effective in a federated setup. Also, using pseudo-labeling to train the model as FedMatch and FedRGD can cause negative impacts on the model

performance, which we elaborate in Chapter 2.

Hence, in this paper, we develop a FL framework *FedSup* to improve SSL applied on unlabeled clients without pseudo-labeling. Our framework design is motivated by BYOL [14] that uses an exponential moving averaged (EMA) online network as a target network for training. Since BYOL is designed for a centralized setup, it is not suitable for FL in which stateless clients participate in each round, i.e. they do not retain the target network once they have finished training. This discontinuous target network that only persists for several local epochs provides inconsistent training signals and increases the diversity across the client networks since they are trained with different target networks, which are known to damage the performance of the model [45]. Instead, we modify its architecture for a federated setup, such that the target network is trained at the server with labeled data. With this architecture, we can achieve more reliable regression targets from the continuously trained target network, at an expense of additional transmission. Also, as all clients see one target network, this can reduce the diversity across the client networks. More importantly, the learning of the target and online networks can be separated; the idea of learning the target and online network from labeled and unlabeled data separately is inspired by the claim in FedMatch that learning a shared set of parameters from both data can cause the network to forget the knowledge learned from labeled data when being additionally trained with unlabeled data [20]. Further inspirations of our design are explained in 4.1.1.

Experiments on IID and non-IID data show that *FedSup* achieves the state-of-the-art performance under finetuning and linear evaluation for 10% labeled-unlabeled ratio. Our contributions are as follows:

- We introduce a FL framework *FedSup* to exploit unlabeled clients with labeled server.
- We conduct various experiments to validate the effectiveness of *FedSup*.
- *FedSup* outperforms other Semi-Supervised Federated Learning methods, in-

cluding the naive applications of SSL to FL and the state-of-the-art methods, verifying its ability to extract useful values from unlabeled data and preserve knowledge learned from labeled data.

Chapter 2

Related works

2.1 Federated Learning

Federated Learning (FL) leverages multiple clients with computing capabilities and local data to collaboratively train a unified model at the server. The central server sends up-to-date model parameter to all clients at each round. Each client then trains the received model for multiple local epochs, minimizing the cross-entropy between the predictions and one-hot encoded labels, and sends back the trained parameter to the server. These parameters are aggregated and updated at the server by a de facto standard approach, FedAvg [34], which takes a weighted average of them. This simple approach, however, suffers performance degradation for non-IID data. FedProx [37] proposes a simple remedy for this problem that restricts the divergence of the model parameters caused during training by heterogeneous class distributions at clients, applying L2 regularization between the received and the local model parameters. FedNova [40] normalizes cumulative gradient updates to optimize the model, so it becomes less sensitive to heterogeneous class distributions. On the other hand, SCAFFOLD [21] reduces inter-client variance to correct for client drifts caused by non-IID data. Also, FedRGD [45] resolves the divergence by group-wise averaging of client parameters and using Group Normalization instead of Batch Normalization. Likewise, we adopt group-wise

averaging to maximize the generalization, which is explained in 4.1.

2.2 Unsupervised Representation Learning

Unsupervised Representation Learning aims to train an encoder using unlabeled data that extracts robust representations, which can be transferred to downstream tasks like classification with a small fraction of labeled data. Obtaining such representations from unlabeled data is important in a real-world setting, where labeled data is few and costly. For representation learning with visual data, there are two mainstream methods, generative or contrastive [29]. Generative approaches learn representations by reconstructing the partially cropped data, employing Generative Adversarial Networks (GAN) [6][36][15][26]. However, the computational burden of generative models makes them unappealing for FL as each client does not hold strong computing resources. Therefore, we focus on contrastive approaches, which utilize less expensive stochastic data augmentation for training. Contrastive learning trains a model by imposing the constraint that the model should output similar representations for augmented views of a same image (positive pairs) and different ones for views of different images (negative pairs). The detailed explanations of contrastive learning methods that we employ in this work are presented in 3.2. The well-known issue of contrastive learning is that the network's output can easily collapse to a single point or to a subspace, resulting in futile solutions. SimCLR [5] observes that maintaining a large number of negative pairs within each batch is crucial for mitigating this issue. Thus, MoCo [16][7] keeps a dictionary of recent batch encodings in the memory bank to increase the number of negative pairs. In contrast, SimSiam [8] prevents collapse by adopting stop-gradient. Furthermore, BYOL [14] attains a target network through EMA of an online network, which provides regression targets for the online network.

2.3 Semi-Supervised Federated Learning

Given that reliably and consistently labeling data in each client is difficult in FL, Semi-Supervised FL has been researched actively [12] [20] [30] [31] [43] [45] [47] to exploit abundant unlabeled or partially labeled data at the clients. One popular method is to use pseudo-labeling [25] that takes the high confidence model predictions on unlabeled instances to advance the model, but this encapsulates several potential issues when applied to FL. Therefore, FedMatch [20] additionally uses other clients' models to obtain more reliable pseudo-labels. Also, FedRGD [45] applies strong and weak data augmentations on the instance to extract more robust pseudo-labels and aggregates the parameters with group-wise averaging, which reduces the gradient diversity of client models and further improves the performance.

2.4 Bias in Classifier

Learning good visual representation is very important for high classification performance of the model, but it has been reported that representations can be disturbed by the bias created at the classification layer [19] [32]. UBNet [19] claims that the classification performance of the model can be improved by using the feature maps from the front layers, which are less biased than those from the rear layers. Moreover, it is suggested that the performance degradation of a classification model in FL is caused by low similarity among feature maps from the rear layers of the model [32]. This implicates that even if the standard FedAvg [34] procedure composes a good encoder, high diversity in the rear layers, especially in the classification layer, may result in decreased classification performance. To prevent this, we design our framework to train a linear classifier only with the labeled data at the server, not adopting pseudo-labeling to train multiple classifiers at the clients.

Chapter 3

Background

3.1 Supervised Federated Learning

For a standard Supervised FL, each client $k \in [1, K]$ possesses labeled dataset $D_k = \{(x_i, y_i)\}_{i=1}^{N_k}$, where x_i is an input, $y_i \in \{1, ..., C\}$ is a label and N_k is the number of data. For each training round $t \in \{1, ..., T\}$, the server selects a subset of clients $A^{(t)}$ that participate in the current round and broadcasts a model $f_\theta^{(t)}$ parametrized by θ to them. Then, the client $k \in A^{(t)}$ updates the model as $f_{\theta_k}^{(t+1)}$ through gradient descent for multiple local epochs E and sends the trained parameter to the server. The server collects these parameters and updates the global model by taking a weighted average of them with FedAvg [34]:

$$f_\theta^{(t+1)} = \sum_{k \in A^{(t)}} \frac{N_k}{\sum_{k' \in A^{(t)}} N_{k'}} \cdot f_{\theta_k}^{(t+1)}. \tag{3.1}$$

3.2 Semi-Supervised Learning

Here we enumerate SSL methods adopted for our experiments, which are depicted on Figure 3.1.

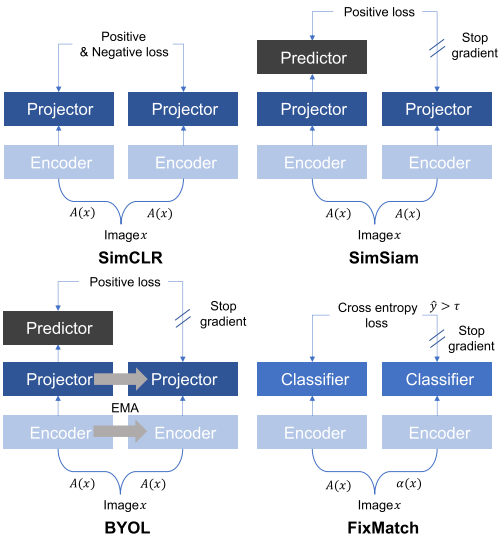


Figure 3.1: **SSL methods.** All methods except FixMatch apply a strong augmentation A twice to the image x. FixMatch applies a weak augmentation α to x to obtain pseudo-labels.

3.2.1 FixMatch

FixMatch [39] extends pseudo-labeling by using augmentations. It adds strong (Cutout [11], CTAugment [3], and RandAugment [10]) and weak (translation and horizontal flip) data augmentations on the instances to extract good pseudo-labels from unlabeled data. The loss of FixMatch is designed as:

$$\mathcal{L} = \frac{1}{B}\sum_{b=1}^{B} H(y_b, f_\theta(\alpha(x_b))) + \frac{\lambda_u}{\mu B}\sum_{b=1}^{\mu B} \mathbf{1}(\max(\hat{y}_b) > \tau)\; H(\hat{y}_b, f_\theta(A(u_b))). \quad (3.2)$$

The first term is a supervised loss, which is a mean cross-entropy H between weakly augmented inputs $\alpha(x_b)$ and labels y_b over B examples. The second is an unsupervised loss, which is also a mean cross-entropy between pseudo-labels $\hat{y}_b = f_\theta(\alpha(u_b))$ from weakly augmented unlabeled instances $\alpha(u_b)$ and predictions from strongly augmented instances $f_\theta(A(u_b))$. $\mathbf{1}(\max(\hat{y}_b) > \tau) \in \{0, 1\}$ indicates that the loss is only computed for pseudo-labels with probability greater than τ. This unsupervised loss

is modulated by a hyperparameter λ_u and the ratio of the amount of unlabeled set to labeled is controlled by μ. In our scenario, where clients hold only unlabeled data, we update the model using the supervised loss at the server and the unsupervised loss at the clients.

3.2.2 SimCLR

SimCLR [5] applies a stochastic data augmentation two times on each image x_b in the training batch of size B, resulting in x_{b1}, x_{b2}. Then, the augmented views x_{b1}, x_{b2} are fed into the encoder to obtain representations, which are again projected by a MLP (projector) to vectors z_{b1}, z_{b2} of smaller dimension. The network is trained by maximizing the cosine similarity between the projection vectors z_{b1} and z_{b2} (positive pairs) while minimizing it between the projection vectors from different images (negative pairs) in the batch. A positive pair of x_{b1} and x_{b2} composes the following loss function (*NT-Xent*)[5]:

$$l_b(1,2) = -\log \frac{\exp(sim(z_{b1}, z_{b2})/\tau)}{\sum_{b'=1}^{B} \mathbf{1}(b' \neq b)(\exp(sim(z_{b1}, z_{b'1})/\tau) + \exp(sim(z_{b1}, z_{b'2})/\tau))} \tag{3.3}$$

where τ is a hyperparameter to control the sharpness of the softmax values, $\mathbf{1}(b' \neq b) = 1$ if $b' \neq b$ otherwise 0 and $sim(\cdot, \cdot)$ is cosine similarity between two vectors. This loss function is evaluated for every positive pair in the batch and symmetrized as $\mathcal{L} = \frac{1}{2B} \sum_{b=1}^{B} l_b(1,2) + l_b(2,1)$, which trains the encoder and the projector. When training is done, the projector is thrown away and a linear layer is attached to the encoder and trained for one epoch to assess the classification performance.

The drawback of SimCLR is that it requires extremely large batch size to obtain many negative pairs for faster convergence and better test accuracy, as [5] reported that the best performance is attained for batch size greater than 2048. However, in FL, the client usually has a memory of limited size, so large mini-batch may not fit into its memory. Moreover, the training items in each client are not as abundant. Therefore, we anticipate that the naive combination of SimCLR and FL will not work well for Semi-

Supervised FL, as corroborated in [43], but we still test its performance to compare with other methods.

3.2.3 SimSiam

Unlike SimCLR, SimSiam learns visual representations, neglecting the negative pairs. In addition to SimCLR, SimSiam uses a predictor that outputs the prediction vectors from the projection vectors. The projection vectors z_{b1}, z_{b2} and the prediction vectors p_{b1}, p_{b2} extracted from x_b constitute loss for a positive pair

$$l_b = \frac{1}{2}\mathcal{D}(p_{b1}, z_{b2}) + \frac{1}{2}\mathcal{D}(p_{b2}, z_{b1}) \tag{3.4}$$

where $\mathcal{D}(p_i, z_j) = -\frac{p_i.z_j}{||p_i||_2||z_j||_2}$. This is evaluated for every positive pair in the batch and averaged to form

$$\mathcal{L} = \frac{1}{B}\sum_{b=1}^{B} l_b \tag{3.5}$$

which is the loss function.

Although above loss can be easily minimized through gradient descent, simple minimization causes representation collapse [8]: the model outputs a constant vector regardless of the input. SimSiam resolves such problem by using stop-gradient, which can be implemented by preventing back-propagation through z_{b1} and z_{b2}. Reflecting the stop-gradient, the above loss function can be rewritten as:

$$\mathcal{L} = \frac{1}{2B}\sum_{b=1}^{B} \mathcal{D}(p_{b1}, z_{b2}.detach()) + \mathcal{D}(p_{b2}, z_{b1}.detach()). \tag{3.6}$$

Compared to other SSL frameworks, SimSiam avoids representation collapse with minimal cost and surpasses performances of all other SSL frameworks presented in this paper, without Momentum Encoder [16] and Online Clustering [4]. Also, unlike SimCLR, SimSiam does not require large batch to train, as it receives loss only from positive pairs, which makes it a promising candidate for Semi-Supervised FL.

3.2.4 BYOL

The architecture and the loss of BYOL (Equation 3.6) are same as SimSiam, but BYOL constructs a target network ξ from the online network θ being trained. Concretely, the EMA of the online network $\xi \leftarrow \mu\xi + (1-\mu)\theta$ is used as the target network to extract z_{b1} and z_{b2} where μ controls the weight of updated θ for computing the average. This does not involve negative pairs, but achieves surprisingly good performance, not collapsing into degenerate solutions.

3.3 Gradient Diversity

FedRGD adopts the gradient diversity introduced in [41] to measure the dissimilarity between local gradient updates at the clients in FL [45]. High gradient diversity is caused by gradient updates towards different directions, so it is problematic to perform FedAvg. For a successful distributed learning of a model, reducing the gradient diversity is important [41].

Chapter 4

Methods

4.1 Algorithm

4.1.1 FedSup

Here we describe our main algorithm FedSup and its design principles. The model architecture is depicted on Figure 4.1 and the training procedure on Figure 4.2.

Disjoint Learning of Supervised and Unsupervised Networks In deep learning, neural networks have a tendency to forget previously learned information when accepting new knowledge from unseen data. This phenomenon is called ”catastrophic forgetting” [38], which also occurs in SSL and should be avoided, especially when labeled data is limited [46]. There have been attempts to deal with this issue in Continual Learning [22] [42]. In FL, FedMatch [20] addresses this problem with disjoint learning of the supervised and unsupervised parameters, only combining them at the server. This helps preserve knowledge from labeled data, while improving representations via unlabeled data. Likewise, we segregate learning of two sets of parameters in our proposed training regime.

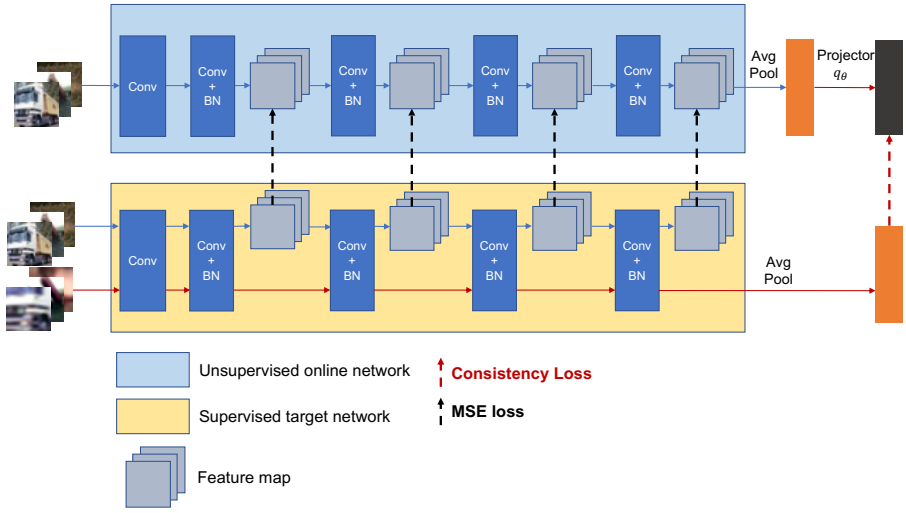


Figure 4.1: **FedSup architecture.** FedSup is trained with two losses: consistency loss $\mathcal{L}_{Con}$ and feature map MSE loss $\mathcal{L}_{MSE}$. The consistency loss is defined between the outputs of the online and target networks from the raw images and the augmented images, like in SimSiam [8] and BYOL [14]. The distinction between the target networks of FedSup and SimSiam/BYOL is that the target network in FedSup is the network supervised at the server, which is contrasted to the Siamese network in SimSiam and the EMA network in BYOL. Also, the MSE loss between the feature maps extracted by two networks from the raw images is applied.

Layer-wise MSE loss Layer-wise feature map loss $\mathcal{L}_{MSE}$ is adopted in knowledge distillation [1] [28] [44], in which a teacher network transfers knowledge to a student network via comparison of layer-wise feature maps. It is observed that supervising the network with softmax outputs as originally suggested in [18] can produce vastly different feature maps, not fully delivering the knowledge of the teacher model and deteriorating the generalization [1]. Such difference in feature maps can be more detrimental in FL, as it can induce higher gradient diversity that harms the generalization [45]. To alleviate this issue, we apply MSE loss between the feature maps of the unsupervised client network and the supervised target network. Specifically, the feature maps from

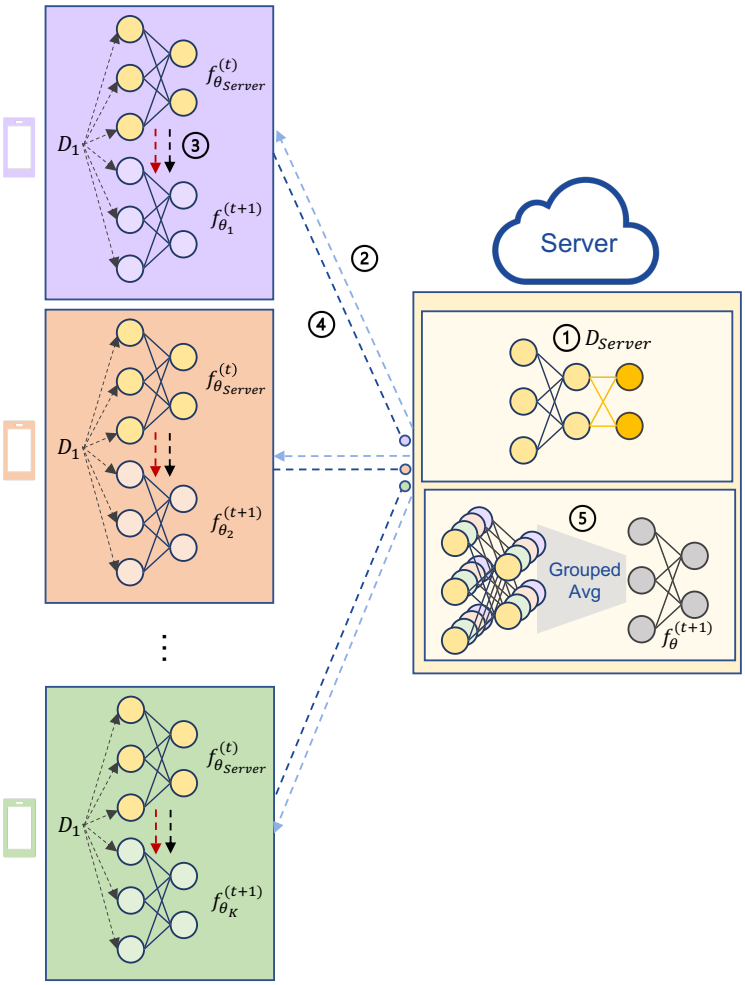


Figure 4.2: **FedSup training procedure.** ① The server trains the network with labeled data D_{Server} ② The supervised network $f_{\theta_{Server}}^{(t)}$ and the online network are distributed to the clients ③ The clients train the online network with their unlabeled dataset D_k, replacing the target network f_ξ in BYOL with $f_{\theta_{Server}}^{(t)}$ and applying layer-wise Mean Squared Error (MSE) loss between feature maps of two networks ④ The trained online networks are sent back to the server ⑤ They are aggregated using group-wise averaging [45] to update the online network only. This process is repeated for T rounds.

each of four convolutional blocks in ResNet-18 are compared. We also adopt a hyperparameter λ_s to decrease this loss by a factor of $\lambda_s^{t/T}$, where t is the current round and T is the total number of rounds, thus increasing the effect of unlabeled data towards the end of the training. We test the performance for $\lambda_s = [10^0, 10^{-2}, 10^{-4}]$ and select $\lambda_s = 10^{-4}$ that demonstrates the highest test accuracy.

Without Pseudo-Labeling For Labels-At-Server scenario, we hypothesize that pseudo-labeling based methods to learn from unlabeled data can be rather disadvantageous. For instance, FixMatch performs back-propagation based on the combination of both supervised and unsupervised losses, where the unsupervised loss is modulated by a small hyperparameter λ_u, which is described in 3.2.1. This makes the model mostly learn from the labeled data, while the unsupervised loss that improves generalization constantly being rectified by the supervised loss. In the absence of such supervised loss at the clients, the models can be drifted by a large margin mainly due to unreliable pseudo-labels, which is also discussed in [2]: since the model learns based on what it already knows, it causes a confirmation bias that harms the model. Also, as mentioned in 2.4, the classifier is easily biased, leading to poor classification performance. This hypothesis is substantiated by our experiments, in which FixMatch that utilizes pseudo-labeling achieves consistently the lowest accuracy. For these reasons, we do not employ pseudo-labeling to train the client networks in FedSup.

Group-wise Averaging FedRGD [45] proposes a method that replaces FedAvg in Labels-At-Server scenario, where the parameters from the clients are divided into S groups and group-wise averaging is performed to update the model. After receiving the client model parameters, the server randomly assigns each parameter into one of the groups from $\{G_i\}_{i=1}^{S}$ such that each group contains the same number of model parameters. Since five clients are used for our experiments, we divide them into groups of two and three to perform this averaging. The group-wise averaging algorithm is described in Algorithm 2.

4.1.2 Semi-Supervised Federated Learning

The training procedure of SSL (FixMatch, SimCLR, SimSiam, BYOL) in our FL setup is presented on Figure 4.3.

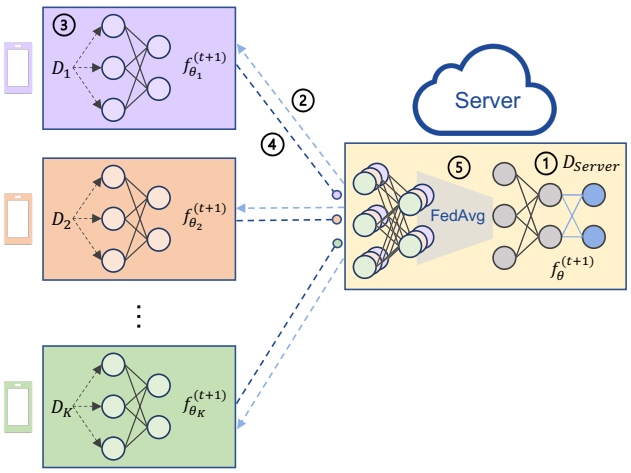


Figure 4.3: **Semi-Supervised FL training procedure.** ① The server trains the network with supervised dataset D_{Server} ② The trained parameter is broadcasted to each client ③ The network is trained at each client with unlabeled dataset D_k using SSL ④ The trained parameters $f_{\theta_k}^{(t+1)}$ are sent to the server and ⑤ aggregated with FedAvg. This process is repeated for T rounds.

Chapter 5

Experimental Details

5.1 Experiments

5.1.1 Setup

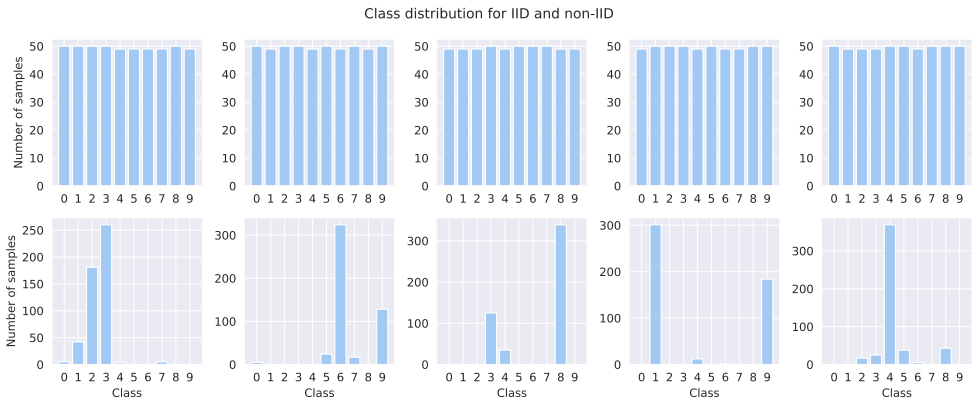


Figure 5.1: **Class distributions for IID and non-IID data.** The class distributions of five clients with 500 items for $\alpha = 10^5$ and $\alpha = 10^{-1}$. For higher α, the class distribution approaches uniform distribution and for smaller α, the clients are assigned unbalanced amounts of samples for each class.

Three datasets CIFAR-10, CIFAR-100 [24] and STL-10 [9] are used in our ex-

periments. For CIFAR-10 and CIFAR-100, we split the data into $50,000$ for training, $5,000$ for labeled and $5,000$ for test data. The training data is distributed across 100 clients by sampling class priors using Dirichlet distribution with different values of α to simulate IID and non-IID as [27] and [31] did, so each client sees 500 unlabeled instances. We define IID when $\alpha = 10^5$ and non-IID when $\alpha = 10^{-1}$. Figure 5.1 shows how two class distributions across five clients differ. For STL-10, we take the original splits as $100,000$ for training, $5,000$ for labeled and $8,000$ for test data. Since it was originally curated for SSL, STL-10 does not hold the labels for unlabeled set, preventing sampling class priors for IID and non-IID data. Therefore, we randomly assign $1,000$ unlabeled samples to each client. In all experiments, ResNet-18 [17] is used as the encoder and five clients are selected for each round from a pool of 100 clients, which are simulated on 4 NVIDIA RTX 3090 GPUs. All methods are implemented using PyTorch [35] and Torchvision [33] with multiprocessing for parallel training. We use the temperature 0.5 for SimCLR, the probability threshold 0.95 for FixMatch, the projection dimension of 512 for SimCLR, Simsiam, BYOL, and the prediction dimension of 512 for SimSiam and BYOL. We run FL for 100 global rounds on CIFAR-10, 200 rounds on CIFAR-100 and STL-10, with 5 local epochs and batch size of 128. For Centralized Supervised Learning, the model is trained for 100 epochs with the same batch size.

5.1.2 Evaluation

To evaluate the model performance, a linear classifier is attached on top of the trained encoder and the model is finetuned end-to-end with the labeled set at the server (Table 6.1). Also, for Semi-Supervised FL and FedSup, we evaluate the quality of representations by only training a linear classifier on top of the frozen encoder (Table 6.2), which is a common practice in SSL [47] [31] and is referred to as linear evaluation.

In Labels-At-Server scenario, STL-10 [9] cannot be evaluated as Fully Supervised FL, as it does not have labels for data distributed to the clients. Also, as FedMatch

[20] and FedRGD [45] did not benchmark on CIFAR-100 and STL-10, we omit their results.

We run following experiments:

- **Fully Supervised FL**: The clients with fully labeled dataset train models following the procedure described in 3.1. Also, the network is additionally trained with the labeled dataset at the server after aggregation. This is the upper bound of the methodologies.

- **Centralized Supervised Learning**: The model is trained with labeled dataset at the server. This experiment is the lower bound of the methodologies.

- **Semi-Supervised FL**: Every client trains the local model with the state-of-the-art SSL methods, SimCLR, SimSiam, FixMatch and BYOL, utilizing the unlabeled data. Then, the aggregated model is supervised with labeled data at the server. The frameworks and the training procedure are explained in 3.2 and Figure 4.3.

- **FedSup**: This is our proposed method. The training procedure is depicted on Figure 4.2 and elaborated in 4.1.1.

Chapter 6

Results and Discussions

6.1 Experimental Results

6.1.1 Main observations

We make the following observations: i) FedSup outperforms other methods on all three datasets under finetuning, and underperforms on CIFAR-100 under linear evaluation for both IID and non-IID, ii) When the label ratio is reduced to 1% for CIFAR-10, such that the server only contains 500 labeled data, FedSup shows similar performance to other methods, because FedSup relies on a highly biased server network for training, iii) FedSup outperforms both FedRGD [45] and FedMatch [20], which implies FedSup can utilize both labeled and unlabeled data more effectively than state-of-the-art methods, iv) The performance gain of FedSup is mainly due to layer-wise MSE loss, v) FedSup slows the convergence down, but reaches higher top accuracy, as shown on Figure 8.1, 8.2, 8.3 in Appendix.

Test Accuracy under Finetuning (%)					
	CIFAR-10		CIFAR-100		STL-10
Method	**IID**	**Non-IID**	**IID**	**Non-IID**	
Centralized	56.92		22.72		55.68
Fully Supervised	79.50	77.30	34.04	32.18	-
FixMatch	59.76	59.62	23.16	22.70	57.45
SimCLR	61.50	61.38	26.82	27.02	61.44
SimSiam	61.08	58.56	24.74	26.34	59.89
BYOL	64.10	62.66	28.28	28.26	64.25
FedMatch [20] [45]	46.81	47.11	-	-	-
FedRGD [45]	63.32	63.24	-	-	-
FedSup	**69.54**	**68.84**	**30.14**	**28.68**	**66.68**

Table 6.1: **Semi-Supervised FL on IID and non-IID data under finetuning.** For FedMatch and FedRGD, we take their values from the paper [45]. It should be noted that FedRGD used the measure of iid-ness $0 \leq R \leq 1$ to simulate IID and non-IID data, whereas we use a parameter α for Dirichlet distribution. FedRGD defined IID and non-IID as $R = 0$ and $R = 0.4$, and we define them as $\alpha = 10^6$ and $\alpha = 10^{-1}$.

6.1.2 Statistical Heterogeneity

Although FedSup shows higher test accuracy than FedRGD, it increases the accuracy gap between IID and non-IID data to 0.7%, as shown in Table 6.1. However, this gap is reduced to 0.1% for linear evaluation in Table 6.2, which suggests that finetuning diminishes the model's capability of dealing with heterogeneity, in exchange for better generalization.

It is also observed that SimSiam and BYOL, which are label-agnostic, are also adversely affected by the heterogeneity, which requires further analysis in the future.

Test Accuracy under Linear Evaluation (%)					
	CIFAR-10		CIFAR-100		STL-10
Method	**IID**	**Non-IID**	**IID**	**Non-IID**	
Centralized	56.92		22.72		55.68
Fully Supervised	79.50	77.30	34.04	32.18	-
FixMatch	59.52	59.40	21.94	23.34	57.36
SimCLR	60.70	60.34	25.30	24.94	59.66
SimSiam	59.44	57.20	23.58	23.96	55.79
BYOL	62.96	61.68	**26.82**	**26.90**	64.03
FedSup	**66.26**	**66.10**	25.36	25.62	**65.80**

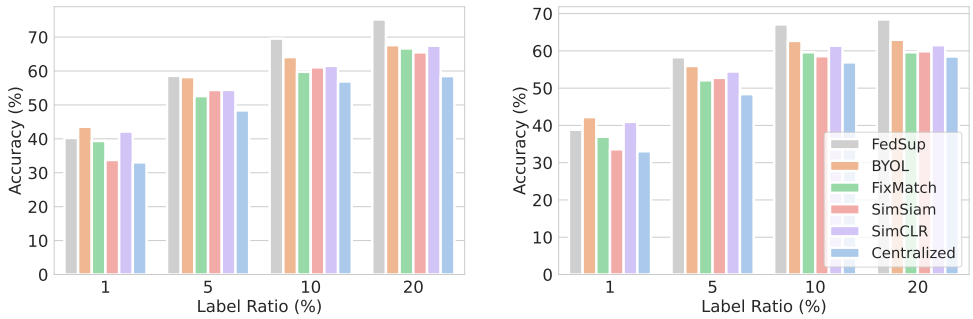
Table 6.2: **Semi-Supervised FL on non-IID and IID data under linear evaluation.** FedSup outperforms the other methods on CIFAR-10 and STL-10 under linear evaluation, but BYOL [14] shows the highest accuracy on CIFAR-100.

6.1.3 Label Ratio

The variation of the performances on CIFAR-10 with different label ratios at the server is presented on Figure 6.1. With 1% labels, FedSup underperforms $2 \sim 3\%$ compared to BYOL and SimCLR on both IID and non-IID data. However, it improves significantly for higher label ratio compared to all other methods, which demonstrates the effectiveness of FedSup for reasonable amount of labels.

6.1.4 Ablation for Loss

FedSup adopts the ideas in the state-of-the-art methods, such as group-wise averaging that replaces FedAvg in FedRGD [45] and disjoint learning of supervised and unsupervised parameters in FedMatch [20], and achieves stronger generalization performance. Through an ablation study varying the numbers of feature maps that contribute to the loss, we find that this improvement is attributed to the layer-wise MSE loss. In detail,



(a) IID (b) Non-IID

Figure 6.1: **Semi-Supervised FL on CIFAR-10 for different label ratios.** The label ratio is the proportion of the server data with respect to the training data. For instance, the server owns $50,000 \times 0.1 = 5,000$ samples for the label ratio of 10% and approximately $45,800 \times 0.2 = 9,160$ samples for 20%.

	Test Accuracy (%)	
N	IID	Non-IID
0	57.20	56.80
1	66.72	67.66
2	68.44	66.74
3	67.68	68.24
4	**69.54**	**68.84**

Table 6.3: **Performance for different numbers of feature maps N contributing to the loss.** For $N > 0$, the test accuracy significantly improves in comparison to $N = 0$. This shows that this loss is an essential factor to the improvements of FedSup.

we include the MSE loss computed with first $N \in \{0, 1, 2, 3, 4\}$ feature maps from the convolutional blocks in ResNet-18 and present the result on Table 6.3. The experiment shows significant performance gain compared to $N = 0$ that achieves almost the same accuracy as the centralized network with no layer-wise loss. This implies that this layer-wise loss is essential for FedSup.

Ablation on CIFAR-10

Loss		Test Accuracy (%)	
Consistency	MSE	IID	Non-IID
✓	✓	**69.54**	**68.84**
	✓	67.76	68.54
✓		57.20	56.80

Table 6.4: **Ablation on the loss of FedSup.** In the absence of the MSE loss, FedSup does not show any improvements compared to the centralized network, which achieves an accuracy of 56.92%. In contrast, in the absence of the consistency loss, FedSup demonstrates better accuracy on non-IID data than on IID data.

Furthermore, we study the effects of removing the consistency loss and the MSE loss from FedSup. As shown in Table 6.4, without the consistency loss, the accuracy on non-IID data drops by 0.3% while it plummets on IID data by 1.8%. This shows that the consistency loss and its associated MLP are less beneficial for non-IID data. We hypothesize that certain bias is created in the MLP for non-IID data, similar to the one in the classifier mentioned in 2.4.

6.1.5 Hyperparameter Search

To properly tune the hyperparameters, we vary the batch size and the learning rate for local training, and choose the ones that show the highest accuracy on CIFAR-10 [24]. As shown in Table 6.5 and 6.6, FedSup shows the best performance with batch size of 128 and local learning rate of 10^{-3}, which are used in all experiments.

	Test Accuracy (%)	
Batch Size	IID	Non-IID
16	68.44	67.10
32	68.72	68.42
64	68.14	67.32
128	**69.54**	**68.84**

Table 6.5: **Sensitivity to local batch size.** Although the maximum performance is attained using a large batch size of 128, FedSup achieves the highest accuracy even for smaller batch sizes.

	Test Accuracy (%)	
Local Learning Rate	IID	Non-IID
10^{-1}	28.72	26.44
10^{-2}	63.86	65.24
10^{-3}	**69.54**	**68.84**
10^{-4}	64.90	64.70

Table 6.6: **Sensitivity to local learning rate.**

6.2 Discussions

6.2.1 Semi-Supervised Learning for Federated Learning

Although we expected that SimCLR [5] would perform worse than other SSL methods, as it is known to demand large batch size for training, it performs competitively to other methods and even exceeds SimSiam [8] and FixMatch [39]. This is consistent with the result in [47], but [31] presents that its performance is 3% lower than SimSiam and 7% than BYOL. This implies that its performance cannot be generalized and may be subject to the implementation details, training environments or various hyperparameters. Also, the strengths of SSL methods can be different when applied to FL; for example, SimSiam accomplishes the best performance in centralized training [8]

but consistently shows low accuracy in FL. Though it is not inspected in this work, it would be an interesting research direction to examine the reason why SimCLR can work competitively to even the state-of-the-art SSL methods in FL.

6.2.2 Lack of Labels

There is a clear limitation in FedSup; when there is an extremely small amount of labels at the server, the supervised network becomes overfitted and thus cannot supply robust training signals to the unsupervised client network. Also, for real-world setting, the server may not hold balanced dataset and may even have missing labels, which can damage the performance of FedSup more severely than other methods. This is not investigated in this work, but it is possible that FedSup may not be scalable for larger scale FL that involves a lot more unlabeled data than labeled data. Our assumption is that if the labels at the server are just enough for centralized network to generalize to a certain degree, it can still utilize unlabeled data fairly well. This is left to future work.

Chapter 7

Conclusion

To run Federated Learning in real-world setting, where labeled data is scarce and expensive, it is crucial to harvest useful values from vast unlabeled data. To this end, we presented *FedSup* that operates in Labels-At-Server scenario, in which the labels only exist at the server. Built on several assumptions and claims in Semi-Supervised Learning and Federated Learning, *FedSup* achieves state-of-the-art performances in both finetuning and linear evaluation schemes, bridging the gap to Fully Supervised FL. We hope that this work brings useful insights and invokes questions for future research.

Bibliography

[1] Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 of number 05, pages 7350–7357, 2020.

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. *CoRR*, abs/1908.02983, 2019. arXiv: 1908.02983.

[3] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets, 2016.

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020.

[8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020.

[9] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[11] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. arXiv: 1708.04552.

[12] Enmao Diao, Jie Ding, and Vahid Tarokh. Semifl: communication efficient semi-supervised federated learning with unlabeled clients. *CoRR*, abs/2106.01432, 2021. arXiv: 2106.01432.

[13] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[14] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: a new approach to self-supervised learning, 2020.

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019. arXiv: `1911.05722`.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[19] Myeongho Jeon, Daekyung Kim, Woochul Lee, Myungjoo Kang, and Joonseok Lee. A conservative approach for unbiased learning on unknown biases. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16752–16760, 2022.

[20] Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Federated semi-supervised learning with inter-client consistency & disjoint learning. In *International Conference on Learning Representations*, 2021.

[21] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.

[22] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural net-

works. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017.

[23] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[25] Dong-Hyun Lee. Pseudo-label : the simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, July 2013.

[26] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning, 2021.

[27] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: an experimental study. *CoRR*, abs/2102.02079, 2021. arXiv: 2102.02079.

[28] Lingbo Liu, Jiaqi Chen, Hefeng Wu, Tianshui Chen, Guanbin Li, and Liang Lin. Efficient crowd counting via structured knowledge transfer. In *ACM International Conference on Multimedia*, 2020.

[29] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*:1–1, 2021.

[30] Nan Lu, Zhao Wang, Xiaoxiao Li, Gang Niu, Qi Dou, and Masashi Sugiyama. Federated learning from only unlabeled data with class-conditional-sharing clients. *arXiv preprint arXiv:2204.03304*, 2022.

[31] Ekdeep Singh Lubana, Chi Ian Tang, Fahim Kawsar, Robert P. Dick, and Akhil Mathur. Orchestra: unsupervised federated learning via globally consistent clustering, 2022.

[32] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: classifier calibration for federated learning with non-iid data. *CoRR*, abs/2106.05001, 2021. arXiv: 2106.05001.

[33] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1485–1488, Firenze, Italy. Association for Computing Machinery, 2010.

[34] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. arXiv: 1602.05629.

[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017.

[36] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. Ocgan: one-class novelty detection using gans with constrained latent representations, 2019.

[37] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018. arXiv: 1812.06127.

[38] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4548–4557. PMLR, October 2018.

[39] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: simplifying semi-supervised learning with consistency and confidence. *CoRR*, abs/2001.07685, 2020. arXiv: 2001.07685.

[40] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *CoRR*, abs/2007.07481, 2020. arXiv: 2007.07481.

[41] Dong Yin, Ashwin Pananjady, Maximilian Lam, Dimitris S. Papailiopoulos, Kannan Ramchandran, and Peter L. Bartlett. Gradient diversity empowers distributed learning. *CoRR*, abs/1706.05699, 2017. arXiv: 1706.05699.

[42] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2017.

[43] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *CoRR*, abs/2010.08982, 2020. arXiv: 2010.08982.

[44] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: improve the performance of convolutional neural networks via self distillation. *CoRR*, abs/1905.08094, 2019. arXiv: 1905.08094.

[45] Zhengming Zhang, Zhewei Yao, Yaoqing Yang, Yujun Yan, Joseph E. Gonzalez, and Michael W. Mahoney. Benchmarking semi-supervised federated learning. *CoRR*, abs/2008.11364, 2020. arXiv: 2008.11364.

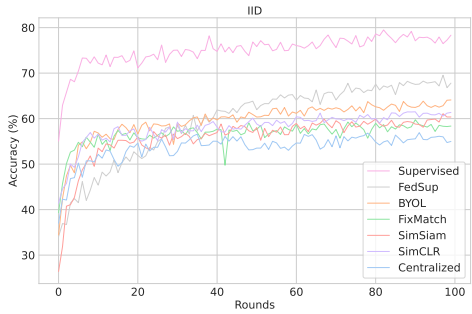
[46] Yan Zhou, Ruyi Jiao, Dongli Wang, Jinzhen Mu, and Jianxun Li. Catastrophic forgetting problem in semi-supervised semantic segmentation. *IEEE Access*, 10:48855–48864, 2022.

[47] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware federated self-supervised learning. *arXiv preprint arXiv:2204.04385*, 2022.

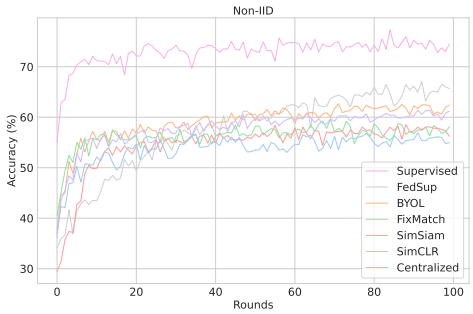
Chapter 8

Appendix

8.1 Detailed Experimental Results

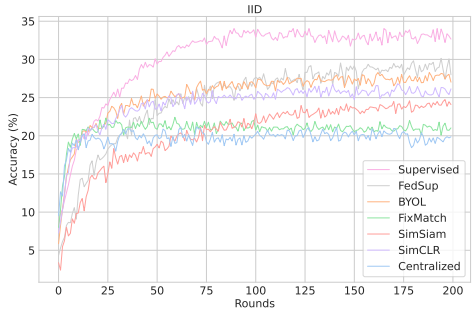


(a) IID

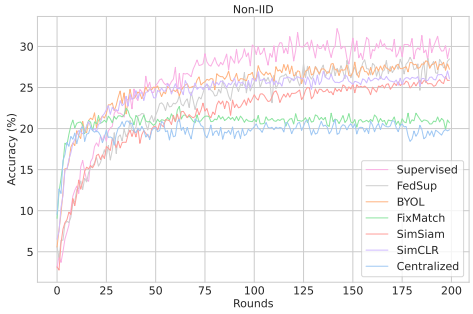


(a) Non-IID

Figure 8.1: **Training curve on CIFAR-10.**



(a) IID



(b) Non-IID

Figure 8.2: **Training curve on CIFAR-100.**

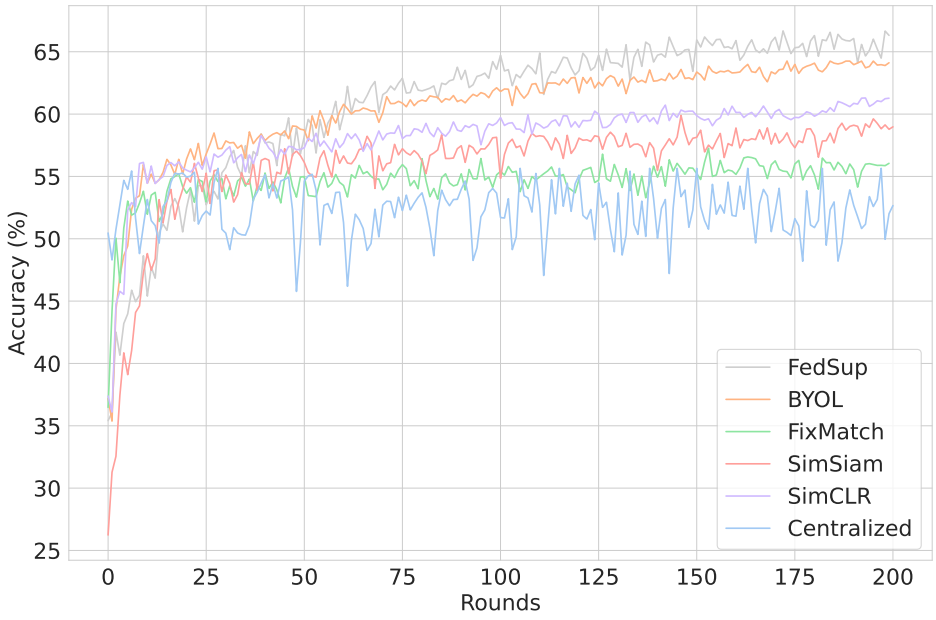


Figure 8.3: **Training curve on STL-10.**

8.2 Algorithms

Algorithm 1 Semi-Supervised Federated Learning with FixMatch/SimCLR/SimSiam/ BYOL. Each client is indexed k and the model is parametrized by θ.

Server Executes

Initialize $f_\theta^{(1)}$

for each round $t = 1, 2, ...T$ **do**

  Train $f_\theta^{(t)}$ with D_{Server}

  Randomly select K clients $A^{(t)}$

  for each client $k \in A^{(t)}$ **run parallel**

    $f_{\theta_k}^{(t+1)} \leftarrow$ **ClientUpdate**$(f_\theta^{(t)}, D_k)$

  $f_\theta^{(t+1)} \leftarrow$**FedAvg**$(f_{\theta_{[1...K]}}^{(t+1)})$

  Finetune $f_\theta^{(t+1)}$ with D_{Server}

ClientUpdate(f_θ, D)

  $\mathcal{B} \leftarrow$ Split D into batches of size B

  for local epoch $e = 1, 2, ...E$ **do**

    for batch $b \in \mathcal{B}$ **do**

      for $x \in b$ **do**

        $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(x; \theta)$

  return f_θ

Algorithm 2 Semi-Supervised Federated Learning with FedSup. Each client is indexed k and the model is parametrized by θ.

Server Executes

Initialize $f_\theta^{(1)}, f_{Server}^{(1)}$
for each round $t = 1, 2, ...T$ **do**
  Train $f_{Server}^{(t)}$ with D_{Server}
  Randomly select K clients $A^{(t)}$
  for each client $k \in A^{(t)}$ **run parallel**
    $f_{\theta_k}^{(t+1)} \leftarrow$ **ClientUpdate**$(f_\theta^{(t)}, f_{Server}^{(t)}, D_k)$
  $f_\theta^{(t+1)} \leftarrow$**GroupwiseAverage**$(f_{\theta_{[1...K]}}^{(t+1)}, f_{Server}^{(t)})$

GroupwiseAverage$(f_{\theta_{[1...K]}}^{(t+1)}, f_{Server}^{(t)})$
  $G_i \leftarrow$ Randomly divide $f_{\theta_{[1...K]}}^{(t+1)}$ into S groups
  for each G_i **do**
    $f_{\theta_i}^{(t+1)} \leftarrow$ **FedAvg**$(f_{\theta_{[j \in G_i]}}^{(t+1)}, f_{Server}^{(t)})$
  return FedAvg$(f_{\theta_{[1...S]}}^{(t+1)})$
ClientUpdate$(f_\theta, f_{Server}^{(t)}, D)$
  $\mathcal{B} \leftarrow$ Split D into batches of size B
  for local epoch $e = 1, 2, ...E$ **do**
    for batch $b \in \mathcal{B}$ **do**
      for $x \in b$ **do**
        $\theta \leftarrow \theta - \eta \nabla_\theta (\mathcal{L}_{Con}(x; \theta, f_{Server}^{(t)}) + \mathcal{L}_{MSE}(x; \theta, f_{Server}^{(t)}))$
  return f_θ

ACKNOWLEDGEMENT

I thank my academic supervisor Hyung Sin Kim for guidance on this work with useful discussions and comments.

초 록

연합 학습(FL)은 여러 클라이언트가 로컬 데이터로 모델을 훈련하고 매개 변수만 서버에 공유하여 중앙 집중식 모델을 만드는 머신 러닝 패러다임이다. 그러나 이 패러다임은 모든 데이터에 레이블이 완전히 지정되어 있다는 비현실적인 가정에 기초한다. 데이터에 레이블을 지정하려면 일반적으로 도메인 전문성과 일관성이 필요한데, 이는 연합 학습에서는 달성하기 어렵다. 그래서, 클라이언트가 레이블이 없는 데이터를 소유하는 반면, 서버에는 레이블이 지정된 데이터(”Labels-At-Server” [20])가 포함되어 있는 시나리오를 고려하는 것이 더 실용적이다. 클라이언트에서 레이블이 지정되지 않은 데이터를 활용하는 방법이 활발히 연구되고 있으며, 이는 확률적 데이터 증강을 활용하여 의사 라벨 (pseudo label)의 품질을 향상시킨다. 최근의 SSL 방법론들과 지식 증류에서 영감을 받아, 우리는 이 문제를 해결하기 위해 준지도 연합학습을 위한 교사-학생 아키텍처 *FedSup*을 제안한다. *FedSup*의 타당성을 입증하기 위해, 우리는 최근 준지도 연합학습 방법론인 FedMatch, FedRGD와 네 가지의 SSL 방법론을 연합학습에 적용하여 CIFAR-10/CIFAR-100/STL-10에 대한 다양한 실험을 수행한다. 독립 항등 분산(IID) 데이터와 비 IID 데이터 모두에서 *FedSup*은 미세 조정 중인 다른 방법에 비해 세 가지 데이터 모두에서 더 높은 정확도를 보여준다. 또한, 우리는 *FedSup*이 잘 작동하는 이유를 탐구하기 위해 CIFAR-10에 대한 절제 연구를 수행하였다.

주요어: 연합학습, 준지도 학습

학번: 2021-26031