Federated Semi-Supervised Learning in Image Classification

BAO HOANG* and MANH TRAN*, Michigan State University, USA

Machine learning models often require large-scale labeled datasets, which are scarce in many real-world scenarios. Semi-supervised learning (SSL) addresses this limitation by leveraging a small amount of labeled data alongside a large pool of unlabeled data, enhancing model performance. In this paper, we explore the effectiveness of SSL techniques in improving image classification tasks. Additionally, we tackle the challenges of data privacy in decentralized environments by adapting SSL algorithms to the federated learning framework. Our approach enables privacy-preserving, distributed training across multiple clients, paving the way for robust and secure semi-supervised machine learning algorithms. Our codes are provided in https://github.com/hoangcaobao/CSE847-Fall2024-FinalProject.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Distributed algorithms**.

Additional Key Words and Phrases: Federated Learning, Semi-supervised Learning, Computer Vision

ACM Reference Format:
Bao Hoang and Manh Tran. 2024. Federated Semi-Supervised Learning in Image Classification. In *Proceedings of MSU CSE 847 - Project*. ACM, New York, NY, USA, 11 pages.

1 Introduction

To train effective machine learning models, we need large-scale data. However, in most real-world scenarios, fully labeled datasets are often scarce. Semi-supervised learning (SSL) offers a promising solution to this challenge by leveraging a small portion of labeled data alongside a large pool of unlabeled data. In this paper, we investigate how semi-supervised learning techniques can enhance model performance by utilizing both labeled and unlabeled data effectively, especially in image classification task.

Furthermore, real-world applications often face strict privacy constraints, preventing the sharing of data between clients to train machine learning models. Federated learning has emerged as a crucial solution to this challenge, enabling decentralized learning without compromising data privacy. In this project, we also aim to adapt and extend existing semi-supervised learning algorithms to the federated learning setting, ensuring privacy-preserving, distributed model training across multiple data sources.

2 Related Works

Self-training [Amini et al. 2024] is one of the earliest semi-supervised learning methods. In each iteration, a supervised model is trained on the labeled data, then it is used to generate pseudo-labels for the unlabeled data. The most confident pseudo-labeled samples are added to the labeled dataset, which is then used for training in the subsequent iteration. Co-training [Rothenberger and Diochnos

*Both authors contributed equally to this research.

Authors' Contact Information: Bao Hoang, hoangbao@msu.edu; Manh Tran, tranman1@msu.edu, Michigan State University, East Lansing, Michigan, USA.

MSU CSE 847 - Project, Dec 04, 2024, East Lansing, MI

2024] is an extension of self-training where two supervised models are trained on different views of the labeled dataset. Each model adds its most confident pseudo-labels to the labeled dataset of the other models for the next iteration of training. The Mean Teacher algorithm [Tarvainen and Valpola 2018] extends Co-training by training a student model and a teacher model on the same data with different noise. The student model is trained with a classification loss on labeled data and a consistency loss that aligns the outputs of both models on unlabeled data. The teacher model's weights are updated using the Exponential Moving Average (EMA) of the student model's weights, helping the student model improve by leveraging unlabeled data and the stable teacher predictions. Label Propagation [Iscen et al. 2019] is a widely-used semi-supervised learning technique that leverages graph-based methods to propagate labels from labeled data points to unlabeled ones based on their similarities within the graph structure. [Sohn et al. 2020] proposed FixMathch, which is a simple combination of 2 common SSL methods: consistency regularization and pseudo labeling. FixMatch enforces consistency between weakly and strongly augmented views of the same unlabeled data, while pseudo labeling assigns labels to the unlabeled data based on high confidence predictions. [Berthelot et al. 2019] introduced MixMatch, a approach that combines MixUp, consitency regularization and entropy minimization. MixMatch generates augmented unlabeled examples by mixing labeled and unlabeled data, producing interpolated samples and their corresponding labels. It further sharpenes predicted labels by adjusting their confidence, encouraging more stable learning. With many existing semi-supervised learning algorithms [H. Chen et al. 2023; L. Chen et al. 2021; Duan et al. 2023; Oliver et al. 2019; Ouali et al. 2020], an unlabeled dataset can be effectively used in the training process.

3 Dataset

In this study, we evaluate three benchmark datasets to assess model performance.

(1) CIFAR-10 [Krizhevsky 2009]: The CIFAR-10 dataset [Krizhevsky 2009] consists of 60,000 32x32 color images, divided into 10 classes, with 6,000 images per class. The 10 classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 50,000 images for training and 10,000 images for testing. For our project, we will use 10% of the training images (5,000) with labels, while the remaining 45,000 training images will be unlabeled for semi-supervised learning algorithms. For the golden baseline, we will use the full set of 50,000 training images with labels.
(2) STL-10 [Coates et al. 2011]: The STL-10 dataset [Coates et al. 2011] consists of 113,000 96x96 color images, divided into 10 classes, with 6,000 images per class. The 10 classes are airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. There are 100,000 unlabeled images, 5,000 labeled images for training, and 8,000 labeled images for testing. For our project, we will use 50% of the unlabeled images (50,000) for training instead of all. We will **not** evaluate the golden baseline, as the unlabeled images in STL-10 are not provided with labels.
(3) Cat and Dog: The Cat and Dog dataset consists of 10,000 32x32 color images divided into 2 classes: cats and dogs, with 5,000 images per class. There are 8,000 images for training and 2,000 images for testing. For our project, we will use 10% of the training images (800) with labels, while the remaining 7,200 training images will be unlabeled for semi-supervised learning algorithms. For the golden baseline, we will use the full set of 8,000 training images with labels. The Cat and Dog dataset can be found at https://www.kaggle.com/datasets/tongpython/cat-and-dog/data.

4 Computer Vision Models

For our project, we evaluated training strategies on 3 different models:

(1) Simple CNN: The SimpleCNN architecture consists of 2 convolutional layers, each followed by ReLU activation and max-pooling for feature extraction. After feature extraction, the model flattens the output and passes it through a fully connected layer with 128 neurons, followed by a ReLU activation. Finally, a second fully connected layer produces output neurons equal to the number of classes.
(2) ResNet-18 [He et al. 2015]: The ResNet-18 architecture [He et al. 2015] consists of 18 layers. It is structured in stages, each containing convolutional blocks with skip connections. The network consists of an initial convolutional layer, followed by 4 main stages of residual blocks, each stage increasing in depth and channel size. Finally, a global average pooling layer and a fully connected layer output class predictions.
(3) DenseNet-121 [Huang et al. 2018]: The DenseNet-121 architecture [Huang et al. 2018] consists of 121 layers that connects each layer to every other layer within a dense block. It has 4 dense blocks with increasing growth rates, where each layer receives input from all preceding layers. Between dense blocks are transition layers with 1x1 convolutions and pooling layers that reduce the spatial dimensions. After the dense blocks, global average pooling is applied, followed by a fully connected layer that produces the final classification output.

5 Semi-supervised Methodology

5.1 Problem Description

Given a feature space $\mathcal{X}$ and a label space $\mathcal{Y}$, we have a labeled training dataset $D_L = \{(x_i, y_i)\}_{i=1}^n$ sampled from the joint distribution $\mathcal{P}_L$, and an unlabeled training dataset $D_U = \{x_i\}_{i=1}^m$ sampled from the marginal distribution $\mathcal{P}_U$. In our problem setting, we assume that $m \gg n$, meaning the number of unlabeled samples is much greater than the number of labeled ones. Our goal is to learn a classifier from the hypothesis space $\mathcal{H}$ that minimizes the expected loss:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathcal{P}_L} [\ell(h, (x, y))]$$

where $\ell : \mathcal{H} \times (\mathcal{X} \times \mathcal{Y}) \mapsto \mathbb{R}_+$ is the loss function. In the context of our image classification task, ℓ is the standard cross-entropy loss.

5.2 Self-training

Self-training [Amini et al. 2024] is an iterative semi-supervised learning algorithm designed to improve classifier performance by gradually incorporating pseudo-labeled samples from an unlabeled dataset. The underlying idea is to train a classifier on a small labeled dataset D_L, use the trained model to generate pseudo-labels for the unlabeled dataset D_U, and iteratively retrain the model by augmenting the labeled data with these pseudo-labeled samples.

Let D_P denote the pseudo-labeled dataset, which is initialized as an empty set and updated at the conclusion of each iteration with new pseudo-labeled samples from D_U. During each iteration, we select the classifier $h \in \mathcal{H}$ that minimizes the empirical loss, which is a weighted sum of the losses on both the labeled dataset D_L and the pseudo-labeled dataset D_P:

$$\frac{1}{n} \sum_{(x,y)\in D_L} \ell(h, (x, y)) + \lambda \frac{1}{|D_P|} \sum_{(x,y')\in D_P} \ell(h, (x, y'))$$

Once the classifier h is trained, we select unlabeled samples from D_U for which the class confidence, as estimated by h, exceeds a specified threshold ϵ. These samples, along with their pseudo-labels, are then added to the pseudo-labeled dataset D_P. The classifier is retrained using both the original labeled dataset and the updated pseudo-labeled dataset. This process is repeated for a fixed number of iterations M, with D_P growing larger at each step.

Algorithm 1 illustrates the self-training procedure.

Algorithm 1 Self-training

Require: $D_L = \{(x_i, y_i)\}_{i=1}^{n}$: Labeled training dataset, $D_U = \{x_i\}_{i=1}^{m}$: Unlabeled training dataset.
Require: λ: weighting factor for pseudo-labeled loss, ϵ: confidence threshold, M: number of iterations.
Ensure: Trained classifier h.

1: Initialize an empty dataset D_P for storing pseudo-labeled samples.
2: **for** $i = 1$ to M **do**
3: Train classifier h to minimize empirical loss:

$$\frac{1}{n} \sum_{(x,y) \in D_L} \ell(h, (x, y)) + \lambda \frac{1}{|D_P|} \sum_{(x,y') \in D_P} \ell(h, (x, y'))$$

4: Use h to assign pseudo-labels $y' = h(x)$ to all samples $x \in D_U$.
5: Add unlabeled samples with pseudo-labels (x, y') to D_P if the confidence score of $h(x)$ exceeds ϵ.
6: Remove the newly added samples from D_U: $D_U = D_U \setminus \{x \in D_P\}$.
7: **end for**

The iterative nature of self-training allows the model to progressively improve as more pseudo-labeled samples are added to D_P, but careful tuning of the confidence threshold ϵ and the weighting factor λ is necessary to avoid overfitting and maintain the quality of the pseudo-labels.

5.3 Mean Teacher

The Mean Teacher algorithm [Tarvainen and Valpola 2018] leverages the concept of co-training by training a *student model* f_s with parameters θ_s and a *teacher model* f_t with parameters θ_t. The student model is optimized using a classification loss on labeled data D_L, while a consistency loss between the student and teacher models is applied on unlabeled data D_U. The total loss for training the student model can be expressed as:

$$\frac{1}{n} \sum_{(x,y) \in D_L} \ell(f_s(x), y) + \lambda \frac{1}{m} \sum_{x \in D_U} \ell_1(f_s(x), f_t(x))$$

where ℓ is the standard cross-entropy loss on the labeled data, and $\ell_1(x, y) = ||x - y||_2$ is the L2-norm measuring the distance between the student and teacher model predictions. The hyperparameter λ controls the trade-off between supervised and unsupervised losses.

After updating the student model, the teacher model is updated through an Exponential Moving Average (EMA) of the student's weights:

$$\theta_t \leftarrow \alpha \theta_t + (1 - \alpha)\theta_s$$

where α is the EMA decay rate, typically a value close to 1 (e.g., 0.99 or 0.999), ensuring that the teacher model evolves smoothly as the student model improves over time.

Algorithm 2 illustrates the mean teacher procedure.

5.4 Fixmatch

FixMatch [Sohn et al. 2020] is a combination of two approaches to SSL: Consistency regularization and pseudo-labeling. Its main novelty comes from the combination of these two ingredients as well as the use of a separate weak and strong augmentation when performing consistency regularization.

Algorithm 2 Mean Teacher

Require: $D_L = \{(x_i, y_i)\}_{i=1}^{n}$: Labeled training dataset, $D_U = \{x_i\}_{i=1}^{m}$: Unlabeled training dataset.
Require: M: number of iterations.
Ensure: Trained teacher model f_t parameterized by θ_t and student model f_s parameterized by θ_s.

1: Initialize an empty dataset D_P for storing pseudo-labeled samples.
2: **for** $i = 1$ to M **do**
3: Train classifier f_s with θ_s to minimize empirical loss:

$$\frac{1}{n} \sum_{(x,y) \in D_L} \ell(f_s(x), y) + \frac{1}{m} \sum_{x \in D_U} \ell_1(f_s(x), f_t(x))$$

4: Train classifier f_t with θ_t using Exponential Moving Average (EMA):

$$\theta_t \leftarrow \alpha\theta_t + (1 - \alpha)\theta_s$$

5: **end for**

The method starts by sampling batches of labeled data and weakly and strongly augmented versions of the unlabeled data. The model predicts logits for both the labeled and unlabeled inputs. For labeled data, the supervised loss is computed using cross-entropy between predicted and true labels. For unlabeled data, pseudo-labels are generated based on the highest confidence prediction from the weakly augmented data. A confidence threshold is applied, and only predictions exceeding the threshold are used to compute the unsupervised loss on the strongly augmented data.

The loss function for FixMatch consists of two cross-entropy loss terms: a supervised loss ℓ_s applied to labeled data and an unsupervised loss ℓ_u. Specifically, ℓ_s is just the standard cross-entropy loss on weakly augmented labeled examples:

$$\ell_s = \frac{1}{B} \sum_{b=1}^{B} \ell(M(\alpha(x_b)), y_b)$$

where ℓ is the standard cross-entropy loss, M is the computer vision model, and α is the augmentation applied to the labeled dataset.

FixMatch computes a pseudo-label for each unlabeled example which is then used in a standard cross-entropy loss. To obtain a pseudo-label, we first compute the model's predicted class distribution given a weakly-augmented version of a given unlabeled image: $q_b = M(\alpha(u_b))$. Then, $\hat{q}_b = \arg\max(q_b)$ is applied to produce one-hot probability distribution as a pseudo-label, except enforcing the cross-entropy loss against the model's output for a strongly-augmented version of u_b:

$$\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbf{1}(\max(q_b) \geq \tau) \ell(M(\mathcal{A}(u_b)), \hat{q}_b)$$

where τ is a scalar hyperparameter denoting the threshold which we retain a pseudo label, $\mathcal{A}$ is the augmentation function used for unlabeled images, and μ is a multiplier hyperparameter that determines the ratio between supervised and unsupervised data.

FixMatch leverages two kinds of augmentations: "weak" and "strong". In our experiments, weak augmentation is a standard flip-and-shift augmentation strategy. For "strong" augmentation, we leveraged RandAugment, which all heavily-distorted version of a given image. Following the approach of pseudo-labeling, we only retain an artificial label if the model assign a high probability to one of the possible classes. Following the approach of pseudo-labeling, artificial label only be retained if the model assigns a high probability to one of the possible classes.

Algorithm 3 illustrates the FixMatch training procedure.

Algorithm 3 FixMatch Training

Require: $D_L = \{(x_i, y_i)\}_{i=1}^{n}$: Labeled training dataset, $D_U = \{x_i\}_{i=1}^{m}$: Unlabeled training dataset.
Require: τ: confidence threshold, T: total epochs, η: learning rate.
Ensure: Trained model M.

for epoch = 1 to T **do**
  Sample batch $(x_l, y_l) \in D_L$, $(x_u^w, x_u^s) \in D_U$ ▷ Weakly and strongly augmented
  Forward pass through model M: $\hat{y}_l, \hat{y}_u^w, \hat{y}_u^s \leftarrow M(x_l, x_u^w, x_u^s)$
  Compute supervised loss $L_x \leftarrow \ell(\hat{y}_l, y_l)$
  Generate pseudo-labels $\hat{y}_u \leftarrow \text{argmax}(\hat{y}_u^w)$
  Apply mask for high confidence: mask $\leftarrow \mathbb{1}(\max(\hat{y}_u^w) \geq \tau)$
  Compute unsupervised loss $L_u \leftarrow \ell(\hat{y}_u^s, \hat{y}_u) \cdot \text{mask}$
  Total loss $L \leftarrow L_x + L_u$
  Update model M using gradients of L
end for

5.5 MixMatch

MixMatch [Berthelot et al. 2019] is a method that combines ideas and components from the current dominant paradigm for semi-supervised learning. Given a batch $\mathcal{X} \subset D_L$ of labeled examples $(x, y) \in \mathcal{X}$ with one-hot targets (representing one of L possible labels) and an equally-sized batch $\mathcal{U} \subset D_U$ of unlabeled examples $(x_u, x'_u) \in \mathcal{U}$ where x_u and x'_u is two type of augmentation from unlabeled samples, MixMatch produces a processed batch of augmented labeled examples and a batch of augmented unlabeled examples with "guessed" labels. The augmented labeled batch and unlabeled batch are then used in computing separate labeled and unlabeled loss terms. More formally, the combined loss L for semi-supervised learning is defined as

$$L_x = \frac{1}{|\mathcal{X}|} \sum_{(x,y)\in\mathcal{X}} \ell(M(x), y)$$

$$L_u = \frac{1}{|\mathcal{U}|} \sum_{(u,q)\in\mathcal{U}} \|q - M(u)\|_2^2$$

$$L = L_x + \lambda L_u$$

where ℓ is the standard cross-entropy loss, M is the computer vision model, and q is the pseudo-label generated using the MixUp approach, which will be described in detail in Algorithm 4.

In the algorithm, MixUp is adopted as a data augmentataion technique used to encourage generalization beyond the exact training samples. By providing convex combinations of samples and their labels, Mixup forces the network to behave smoothly between different classes. Algorithm 4 illustrates the MixMatch training procedure.

6 Federated Learning Methodology

6.1 Problem Description

Using terminology from Section 5.1, for decentralized settings, we have k clients with client i having access to labeled data $D_L^{(i)}$ sampled from $P_L^{(i)}$ and unlabeled data $D_U^{(i)}$ sampled from $P_U^{(i)}$.

Algorithm 4 MixMatch Training

Require: $D_L = \{(x_i, y_i)\}_{i=1}^{n}$: Labeled training dataset, $D_U = \{x_i\}_{i=1}^{m}$: Unlabeled training dataset.
Require: T: Total epochs, α: MixUp parameter, η: Learning rate.
Ensure: Trained model M.

for epoch = 1 to T **do**
  Sample batch $(x_l, y_l) \in D_L$, $(x_u, x'_u) \in D_U$ ▷ Labeled and weakly augmented unlabeled data
  Forward pass through model M:
    $\hat{y}_u = M(x_u), \hat{y}'_u = M(x'_u)$
  Generate guessed labels for unlabeled data:
    $\tilde{y}_u = (\hat{y}_u + \hat{y}'_u)/2$
  Concatenate all inputs and targets:
    $all\_inputs = [x_l, x_u, x'_u]$
    $all\_targets = [y_l, \tilde{y}_u, \tilde{y}_u]$
  Sample mixing coefficient $l \sim \text{Beta}(\alpha, \alpha)$
  Set $l = \max(l, 1 - l)$
  Shuffle all_inputs and $all\_targets$ to get $input\_b$ and $target\_b$
  MixUp:
    $mixed\_input = l \cdot all\_inputs + (1 - l) \cdot input\_b$
    $mixed\_target = l \cdot all\_targets + (1 - l) \cdot target\_b$
  Forward pass with mixed inputs:
    $\hat{y}_l, \hat{y}_u = M(mixed\_input)$
  Compute supervised loss $L_x \leftarrow \ell(\hat{y}_l, mixed\_target[: batch\_size])$
  Compute unsupervised loss
    $L_u \leftarrow \|\hat{y}_u - mixed\_target[batch\_size :]\|^2$
  Total loss $L \leftarrow L_x + \lambda \cdot L_u$
  Update model M using gradients of L
end for

Our objective is to collaboratively learn a global classifier $h \in \mathcal{H}$ that minimizes the expected loss over all clients:

$$\min_{h \in \mathcal{H}} \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}_{(x,y) \sim \mathcal{P}_L^{(j)}} \left[\ell(h, (x, y)) \right]$$

In this decentralized framework, clients collaboratively train the classifier without directly sharing their data. Instead, they may communicate model updates or representations, ensuring data privacy and adherence to any relevant communication constraints. The global model aims to perform well across all client distributions, making it robust to distributional heterogeneity among client data.

6.2 FedAvg

Federated Averaging (FedAvg) [McMahan et al. 2023] is an algorithm designed for distributed model training. In FedAvg, a global model is initially broadcasted to each client. Each client then trains the model locally using its own dataset, resulting in a set of updated client-specific models. These local models' weights are then averaged and used to update the global model. This process is repeated iteratively, allowing the global model to improve progressively based on decentralized data while preserving data privacy. The algorithm 5 illustrate federated average algorithm.

7 Experimental Results

Table 1 presents the performance of four semi-supervised methods—Self-training, Mean Teacher, FixMatch, and MixMatch—compared to two baselines: (1) a baseline trained exclusively on labeled images and (2) a "golden baseline" trained on the full training dataset (including both labeled and

Algorithm 5 Federated Averaging (FedAvg) for Semi-Supervised Learning

Require: k: Number of clients.
Require: ClientTrain(): Semi-supervised training algorithm (e.g., Self-training, Mean Teacher, FixMatch, MixMatch).
Require: $D_L^{(i)}$: Labeled training dataset for client i, $D_U^{(i)}$: Unlabeled training dataset for client i.
Require: M: Number of global iterations.
Ensure: Global model f parameterized by θ.
1: Initialize global model parameters θ.
2: **for** epoch = 1 to M **do**
3: **for** each client $i = 1$ to k **in parallel do**
4: **Local Training:** Client i trains model with ClientTrain($D_L^{(i)}$, $D_U^{(i)}$, θ) to obtain updated parameters $\theta^{(i)}$.
5: **end for**
6: **Global Aggregation:** Update global model $\theta = \frac{1}{k} \sum_{i=1}^{k} \theta^{(i)}$.
7: **end for**

unlabeled images). The evaluation is conducted across three computer vision architectures: Simple CNN, ResNet-18, and DenseNet-121. From the table, we observe the following key insights:

- Across all three computer vision architectures, semi-supervised methods consistently outperform the baseline that uses only labeled data. This highlights the importance of leveraging unlabeled data to enhance model performance in semi-supervised settings.
- Among the semi-supervised methods, FixMatch and MixMatch demonstrate superior performance, outperforming Self-training and Mean Teacher. This can be attributed to their effective use of strong data augmentation and consistency regularization, which help the model better exploit the unlabeled data.
- Despite the improvements, the semi-supervised methods still fall short of the golden baseline, which is trained on the full labeled dataset. This underscores the continued importance of high-quality labeled data for achieving optimal performance.

Table 2 presents the performance of four semi-supervised methods—Self-training, Mean Teacher, FixMatch, and MixMatch—compared to two baselines: (1) a baseline trained exclusively on labeled images and (2) a "golden baseline" trained on the full training dataset (including both labeled and unlabeled images) in **decentralized settings**, where the Federated Averaging algorithm is used to aggregate models' parameters from different clients. The number of clients is set to five. The evaluation is conducted across three computer vision architectures: Simple CNN, ResNet-18, and DenseNet-121. From the table, we observe the following key insights:

- The performance of almost all methods, across all models and datasets in decentralized settings, is lower compared to centralized settings. This implies that training data in a centralized model allows the model to learn better than merely aggregating model parameters.
- Although all methods experience a decrease in performance in decentralized settings compared to centralized settings, MixMatch and Self-training methods remain effective. This indicates that certain algorithms can preserve performance in decentralized settings, while others, such as FixMatch and Mean Teacher, show significant performance degradation in decentralized environments where collaboration between clients relies on sharing model parameters.

Table 1. Accuracy of Semi-Supervised Learning Methods on Different Datasets With Different Model Architectures in Centralized Setting

Methodology	CIFAR-10	STL-10	Cat And Dogs
Simple CNN			
Baseline	67.94%	68.14%	70.14%
Golden Baseline	82.01%	–	81.76%
Self-training	69.14%	69.38%	72.61%
Mean teacher	68.71%	70.59%	72.27%
FixMatch	70.70%	69.71%	72.81%
MixMatch	**72.35%**	**71.75%**	**74.34%**
ResNet-18			
Baseline	68.53%	75.55%	71.38%
Golden Baseline	85.84%	–	88.83%
Self-training	70.45%	77.61%	72.52%
Mean teacher	70.64%	74.38%	67.08%
FixMatch	75.12%	80.20%	76.91%
MixMatch	**79.33%**	**80.86%**	**81.30%**
DenseNet-121			
Baseline	67.00%	75.70%	73.46%
Golden Baseline	87.56%	–	88.38%
Self-training	70.09%	77.05%	74.35%
Mean teacher	70.23%	75.52%	70.79%
FixMatch	75.95%	77.37%	73.15%
MixMatch	**79.92%**	**80.32%**	**81.90%**

8 Discussion and Future Works

In this project, we implemented four semi-supervised learning (SSL) techniques—Self-Training, Mean Teacher, MixMatch, and FixMatch—and evaluated their performance across three datasets (STL10, CIFAR10, Cat and Dog) using three computer vision architectures (Simple CNN, ResNet-18, DenseNet-121) in both centralized and decentralized settings.

All four SSL methods consistently outperformed the baseline of training with limited labeled data, demonstrating the value of incorporating unlabeled data to enhance model performance. Among these, MixMatch and FixMatch achieved the best results. However, none of the SSL methods surpassed the performance of fully supervised training on the entire labeled dataset, highlighting the critical importance of high-quality labeled data in achieving optimal model performance.

In decentralized settings, where the Federated Averaging (FedAvg) algorithm was used to update model parameters across clients without sharing raw data, performance declined for all methods. This suggests that decentralized learning introduces efficiency challenges compared to centralized approaches. Nevertheless, Self-Training and MixMatch showed relatively robust performance, indicating that certain SSL methods adapt better to decentralized environments than others.

For future work, we aim to explore more sophisticated federated learning algorithms, such as FedProxy [Peng et al. 2024], to address the limitations of FedAvg. Additionally, we plan to extend

Table 2. Accuracy of Semi-Supervised Learning Methods on Different Datasets With Different Model Architectures In Decentralized Setting

Methodology	**CIFAR-10**	**STL-10**	**Cat And Dogs**
Simple CNN			
Baseline	61.41%	62.00%	69.85%
Golden Baseline	76.84%	–	80.82%
Self-training	68.24%	**69.42%**	**74.59%**
Mean teacher	62.23%	61.98%	68.51%
FixMatch	65.15%	63.04%	71.91%
MixMatch	**69.66%**	68.50%	72.12%
ResNet-18			
Baseline	62.80%	71.04%	68.51%
Golden Baseline	82.89%	–	85.91%
Self-training	70.51%	**76.62%**	74.84%
Mean teacher	61.23%	68.94%	59.71%
FixMatch	70.04%	75.85%	67.47%
MixMatch	**74.01%**	73.56%	**80.20%**
DenseNet-121			
Baseline	61.56%	72.09%	70.54%
Golden Baseline	84.32%	–	86.90%
Self-training	68.56%	**76.00%**	73.55%
Mean teacher	61.48%	69.73%	66.68%
FixMatch	70.70%	73.23%	67.42%
MixMatch	**75.81%**	74.85%	**80.37%**

our evaluation to a broader range of datasets and model architectures to further validate and generalize our conclusions.

References

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Lies Hadjadj, Emilie Devijver, and Yury Maximov. 2024. *Self-Training: A Survey*. (2024). https://arxiv.org/abs/2202.12040 arXiv: 2202.12040 [cs.LG].

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. *MixMatch: A Holistic Approach to Semi-Supervised Learning*. (2019). https://arxiv.org/abs/1905.02249 arXiv: 1905.02249 [cs.LG].

Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. 2023. *SoftMatch: Addressing the Quantity-Quality Trade-off in Semi-supervised Learning*. (2023). https://arxiv.org/abs/2301.10921 arXiv: 2301.10921 [cs.LG].

Luoxin Chen, Francisco Garcia, Varun Kumar, He Xie, and Jianhua Lu. 2021. *Industry Scale Semi-Supervised Learning for Natural Language Understanding*. (2021). https://arxiv.org/abs/2103.15871 arXiv: 2103.15871 [cs.CL].

Adam Coates, Andrew Ng, and Honglak Lee. 2011. *An Analysis of Single-Layer Networks in Unsupervised Feature Learning*. (2011). https://proceedings.mlr.press/v15/coates11a.html.

Yue Duan, Zhen Zhao, Lei Qi, Luping Zhou, Lei Wang, and Yinghuan Shi. 2023. *Towards Semi-supervised Learning with Non-random Missing Labels*. (2023). https://arxiv.org/abs/2308.08872 arXiv: 2308.08872 [cs.LG].

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. *Deep Residual Learning for Image Recognition*. (2015). https://arxiv.org/abs/1512.03385 arXiv: 1512.03385 [cs.CV].

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. *Densely Connected Convolutional Networks*. (2018). https://arxiv.org/abs/1608.06993 arXiv: 1608.06993 [cs.CV].

Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. 2019. *Label Propagation for Deep Semi-supervised Learning*. (2019). https://arxiv.org/abs/1904.04717 arXiv: 1904.04717 [cs.CV].

Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. (2009). https://api.semanticscholar.org/CorpusID:18268744.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2023. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. (2023). https://arxiv.org/abs/1602.05629 arXiv: 1602.05629 [cs.LG].

Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. 2019. *Realistic Evaluation of Deep Semi-Supervised Learning Algorithms*. (2019). https://arxiv.org/abs/1804.09170 arXiv: 1804.09170 [cs.LG].

Yassine Ouali, Céline Hudelot, and Myriam Tami. 2020. *An Overview of Deep Semi-Supervised Learning*. (2020). https://arxiv.org/abs/2006.05278 arXiv: 2006.05278 [cs.LG].

Zhaopeng Peng, Xiaoliang Fan, Yufan Chen, Zheng Wang, Shirui Pan, Chenglu Wen, Ruisheng Zhang, and Cheng Wang. 2024. *FedPFT: Federated Proxy Fine-Tuning of Foundation Models*. (2024). https://arxiv.org/abs/2404.11536 arXiv: 2404.11536 [cs.LG].

Jay C. Rothenberger and Dimitrios I. Diochnos. 2024. *Meta Co-Training: Two Views are Better than One*. (2024). https://arxiv.org/abs/2311.18083 arXiv: 2311.18083 [cs.CV].

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. 2020. *FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence*. (2020). https://arxiv.org/abs/2001.07685 arXiv: 2001.07685 [cs.LG].

Antti Tarvainen and Harri Valpola. 2018. *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results*. (2018). https://arxiv.org/abs/1703.01780 arXiv: 1703.01780 [cs.NE].