A Robust Collaborative Learning Framework Using Data Digests and Synonyms to Represent Absent Clients

Chih-Fan Hsu
Inventec Corporation
Taipei 111, Taiwan
hsuchihfan@gmail.com

Ming-Ching Chang
University at Albany - SUNY
Albany NY 12065, USA
mchang2@albany.edu

Wei-Chao Chen
Inventec Corporation
Taipei 111, Taiwan
chen.wei-chao@inventec.com

Abstract

We propose Collaborative Learning with Synonyms (CLSyn), a robust and versatile collaborative machine learning framework that can tolerate unexpected client absence during training while maintaining high model accuracy. Client absence during collaborative training can seriously degrade model performances, particularly for unbalanced and non-IID client data. We address this issue by introducing the notion of data digests of the training samples from the clients. The expansion of digests called synonyms can represent the original samples on the server and thus maintain overall model accuracy, even after the clients become unavailable. We compare our CLSyn implementations against three centralized Federated Learning algorithms, namely FedAvg, FedProx, and FedNova as baselines. Results on CIFAR-10, CIFAR-100, and EMNIST show that CLSyn consistently outperforms these baselines by significant margins in various client absence scenarios.

1. Introduction

With the success of modern machine learning (ML), the performance of a model often depends on the availability and quality of data. In practice, the large amount of data may be held by multiple parties. Collecting data to a central site for model training can incurs large overheads in management, compliance, privacy concerns, or even regulation and judicial issues [1, 2]. In this regard, *collaborative ML* takes a step further by assuming that multiple learning parties can collaboratively train a ML model [2]. *Federated learning* [3] is collaborative ML without centralized training data, where modern concepts including heterogeneity, data privacy, model privacy, security, anti-adversarial attacks, incentive mechanisms are properly addressed.

Client robustness is an important issue in collaborative learning. Client absence can be caused by short-term network communication breakage [3, § 3.5], or long-term cease of participation due to business regulation or competitions, which are much less addressed in the literature. Model training with client absence can degrade severely when the client data is *non-independent-and-identically-distributed* (non-IID) [4]. The client absence issue becomes more severe as the skewness of the client data increases.

In this paper, we develop a solution for robust, uninterrupted collaborative training under client absence. We consider the following three scenarios: (1) unreliable clients, (2) continue training after clients leave for a long duration or disappear permanently, (3) carry on a stopped training after new clients join.

We propose *Collaborative Learning with Synonyms* (CLSyn), a collaborative machine learning framework that can address client absence by learning to synthesize training samples while client data are available. We name the synthesized samples the *synonym* of the training samples that are kept at a centralized training *moderator*. An encoder is used at each client to produce digests, while a *generator* is learned at the centralized moderator to generate synonyms from digests. Since under non-IID client data setting, the model trained at each client is very different, such that any client leaving can cause catastrophic performance drop. To this end, a corresponding *replacement* model operating on the synonyms can be use to continue the joint model training. In each training iteration, data digests are produced at each client and sent to the moderator for model update. At the moderator side, synonyms can be generated from the respective digests of each client to continue the collaborative training. This way, the raw private data never leave each client to ensure privacy during training.

The proposed CLSyn algorithm can jointly learn synonym generation and the collaborative model training. Our design is motivated by the following two intuitions and controlled by respective loss functions. (1) A *Data Similarity Loss* (DSL) ensures the projection of the private data (*i.e.* the digest) and the projection of synonym should be similar. (2) A *Synonym Classification Loss* ensures that the synonym and the digest produced at each client should be classified well by the learned model. Our experiments compare CLSyn with fully centralized learning (which violates data

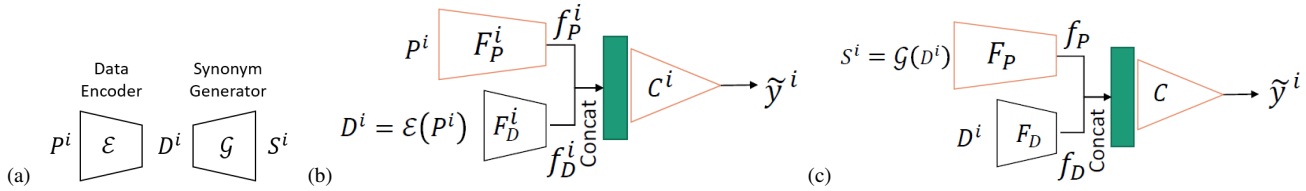


Figure 1: **Overview of the proposed network models.** (a) Encoder $\mathcal{E}$ encodes the private data P^i of each client i into the *digest* D^i, and the synonym generator $\mathcal{G}$ to decode D^i into the *synonym* S^i of the original data. (b) The client model $\mathcal{M}^i$ takes its private data P^i and digest D^i and perform feature extraction. The extracted features are concatenated to perform classification. (c) The server model $\mathcal{M}$ has an identical structure as $\mathcal{M}^i$ but working with different data access.

locality) as well as three common federated learning algorithms, namely, FedAvg [5], FedProx [6], and FedNova [7]. Experimental results show that the proposed CLSyn framework can significantly enhance collaborative training performance against client absence.

The contribution of this paper includes the following:

- We propose a collaborative learning framework based on data digest encoding and synonym generation, to handle client absence with non-IID data distributions.
- The ML model and synonym generator are jointly trained end-to-end, such that model training can progress seamlessly with either available or absent clients.
- CLSyn can support *privacy-preserving* model training, as only digests are sent out for training, and private data is never shared outside each client. The privacy level depends on the design of digest generation.
- Extensive evaluations on CIFAR-10, CIFAR-100, and EMNIST with various client leaving and rejoining scenarios demonstrate the efficacy of CLSyn.

2 Related Work

There exists a large amount of works regarding distributed or collaborative ML [1]. Most literature is on (1) addressing client data privacy or model privacy using *differential privacy* (DP) algorithms [8], (2) handling non-IID distribution for effective model learning, and (3) addressing limited computational resources or network bandwidth to perform distributed learning. These solutions cannot effectively deal with client leaves during training, especially for protected client data that are non-IID distributed. Due to privacy protection, private client data should be kept and accessible only at each client. As one or more clients leave, the non-IID distribution will cause a crucial amount of *representative* data to disappear, resulting in biased gradient update and long-term training degradation.

3 Methodology

We hypothesize that a training *data memorizing* mechanism is required to effectively handle client leaves. We handle client absence during collaborative training by encoding the private data of each client as *digests* that will be shared with the moderator. [1] When clients are absent, the *moderator* generates synonyms S to represent the private data P from the stored digests D to continue training (Fig. 1). The use of digests and synonyms is versatile and adaptable to most existing architectures to perform collaborative training in applications. The synonym generator $\mathcal{G}$ is jointly trained and optimized with the collaborative learning model on the moderator, such that the most suitable synonyms can be generated. Since the generator $\mathcal{G}$ will synthesize synonyms from the digests, $\mathcal{G}$ should be protected to avoid undesired access from any clients to avoid potential data leak or adversarial attacks.

3.1 Digest Encoding and Synonym Generation

Each client i in CLSyn encodes private training samples into digests D^i by a data encoder $\mathcal{E}$ locally. The digests are transmitted and stored at a training moderator, so collaborative training can continue using synonym S^j produced from D^j. The selection of the data encoder $\mathcal{E}$ depends on the property, complexity, and privacy level of the collaborative training task. Note that S should be within the same domain as P. Ideally, due to the transformation, they may not be identical.

In the CLSyn framework, the data encoder $\mathcal{E}$ and the client model $\mathcal{M}^i$ belong to the client, while the synonym generator $\mathcal{G}$ and the server model $\mathcal{M}$ belong to the moderator. At each client, we fix $\mathcal{E}$ (thus D is also fixed given the input) to prevent the digest D depending on $\mathcal{E}$. At the moderator, we expect $\mathcal{M}$ to learn from synonyms S generated by $\mathcal{G}$. To achieve this goal, the following model structure and training process are designed.

Since CLSyn must handle both cases of training with available clients and client absence, our design intuition is that, both the feature extractor and the task driven ML model should produce as-similar-as-possible results from (i) the digests and private data (D, P) and (ii) the digests and synonyms (D, S). Observe in Fig. 1(b,c) that the client model $\mathcal{M}_i$ and server model share identical structure except

[1] We assume that the digest and synonym generators are not reversible under adversarial attacks. Given that the synonym generator is only kept at the secure moderator and not accessible by the clients, the private data for each client is thus protected.

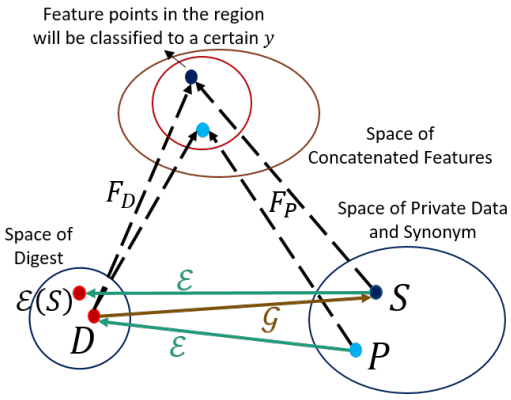


Figure 2: The relations and mappings among the client data P, digests D and synonyms S.

working with different data access.

Fig. 2 provides a schematic plot showing the relations and mappings among *representation spaces* (client data P, digests D, and synonyms S) and *functions* (synonym generator $\mathcal{G}$, data encoder $\mathcal{E}$, and feature extractors F). The collaborative ML model takes the concatenation of the extracted features from the private data P (or synonyms S) and digests D for making a decision.

Client and server models. In collaborative learning, model training occurs at each client, and model gradients are transmitted to the server and aggregated to update the server model. In CLSyn, when a client i is available, client model $\mathcal{M}^i$ is trained using private data P^i together with its digest D^i, as shown in Fig. 1(b). The model gradient $\nabla\mathcal{M}^i$ is sent to the server, where the gradients from all clients are aggregated to generate the server model $\mathcal{M}$. Note that $\mathcal{M}^i$, $\nabla\mathcal{M}^i$, and $\mathcal{M}$ are in the same shape and size. CLSyn follows the common structure of the collaborative learning that enforces identical structures for the server model $\mathcal{M}$ and the client model $\mathcal{M}^i$, with the only difference of data accessing. When the client is available, its private data are used to generate digests for training; whenever clients become absent, the server will take the digests and reconstruct the synonyms for the missing clients to continue training. This way, CLSyn training does not interrupt whether clients are present or not.

Fig. 1(a) illustrates the relationship between the data encoder and synonym generator. Ideally, the dimension of the digest D is usually less than the that of the private data P and the synonyms S. Combining with the non-invertible data encoder $\mathcal{E}$, private data are thus protected by the sharing of the digests. In our experiment, we design two shallow convolutional networks to represent $\mathcal{E}$ and $\mathcal{G}$, respectively.

3.2 The CLSyn Training Steps

The CLSyn training can be viewed as an extension of FedAvg [9] with the newly introduced designs of digests, synonyms, and loss functions. The loss functions control the update of the generator $\mathcal{G}$ and help retaining model performance when trained with possibly absent clients. Every iteration of CLSyn training involves the following steps:

- Moderator pushes model $\mathcal{M}$ to every client i.
- Each client i encodes private data P^i into digest D^i and then trains model $\mathcal{M}^i$ with P^i and D^i as input using stochastic gradient decent (SGD). The digest D^i and model gradients $\nabla\mathcal{M}^i$ are then transmitted to the moderator. D^i only needs to be transmitted once at the beginning of training if $\mathcal{E}$ is fixed.
- Moderator collects $\nabla\mathcal{M}^i$ and determines if any client is absent. If client j is absent, a replacement model $\hat{\mathcal{M}}^j$ is used to calculate the gradient $\nabla\mathcal{M}^j$ using the digests D^j and the generated synonyms $S^j = \mathcal{G}(D^j)$.
- Moderator updates $\mathcal{M}$ by aggregating $\nabla\mathcal{M}^i$ and $\nabla\mathcal{M}^j$.
- Moderator updates $\mathcal{M}$ and $\mathcal{G}$ with all collected digests D and the generated synonyms S.

3.3 Losses for Joint Training of $\mathcal{G}$ and $\mathcal{M}$

The ML model $\mathcal{M}$ aims to learn: (1) how best to generate appropriate S and (2) how best to perform classification *i.e.* determine y from S and D. To achieve this, we introduce two intuitions that establish the joint training of $\mathcal{M}$ and $\mathcal{G}$. Fig. 2 illustrates concepts related to these ideas.

- **Intuition 1:** We aim to train $\mathcal{M}$ so that it is capable to correctly classify the information obtained from D and S. This intuition is enforced by the two black dash lines starting from D and S in Figure 2.
- **Intuition 2:** We want the digest of the synonym $\mathcal{E}(S)$ to resemble the digest of the private data D, *i.e.* $\mathcal{E}(S) \sim D$. This intuition is indicated by the green lines between the digest space and the original private data space.

The following two loss functions are proposed to achieve **Intuitions 1 and 2**. The **Synonym Classification Loss** $\mathcal{L}_{SCL}$ ensures that the synonyms S and digests D should be classified well by the model $\mathcal{M}$:

$$\mathcal{L}_{SCL} = \mathcal{L}_{ce}\left(\mathcal{M}(\mathcal{G}(D), D), y\right), \tag{1}$$

where the trainable networks are highlighted in red. The **Data Similarity Loss** $\mathcal{L}_{DSL}$ ensures that the projection of the private data and the projection of the synonyms S should be similar:

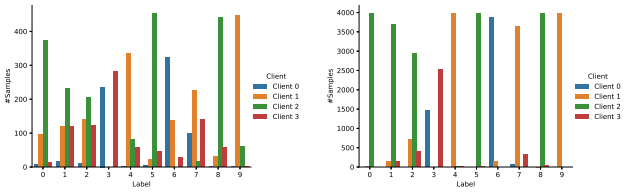
$$\mathcal{L}_{DSL} = \mathcal{L}_{mse}\left(\mathcal{E}(\mathcal{G}(D)), D\right). \tag{2}$$

The total loss for jointly training $\mathcal{G}$ and $\mathcal{M}$ is: $\mathcal{L}_{server} = \mathcal{L}_{DSL} + \lambda\mathcal{L}_{SCL}$, where λ is a balancing hyperparameter, which is set to $\lambda = 1$ in our experiments.

At each client i, the training data consist of the private data P^i and digest D^i produced via encoder $\mathcal{E}$. Each client trains the client model $\mathcal{M}^i$ using the **Classification Loss**:

$$\mathcal{L}_{client} = \mathcal{L}_{ce}\left(\mathcal{M}^i(P^i, D^i), y\right). \tag{3}$$

Once the client j is absent, a replacement model $\hat{\mathcal{M}}^j$ and the synonyms $\mathcal{G}(D^j)$ will take over the roles of the client and its data to continue training.



(a) $\mu = 0.5$ (b) $\mu = 0.1$

Figure 3: Examples of non-IID data distributions of the 10 classes of CIFAR-10 for the 4 clients; μ controls the skewness of non-IID data distributions.

4 Experiment Results

We perform experiments on three common classification datasets CIFAR-10 [10], CIFAR-100 [10], and EMNIST [11]. We evaluate CLSyn against three implementation of collaborative learning algorithms FedAvg [9], FedProx [6], and FedNova [7] as baselines.

Experiment setup. All experiments are conducted on a Linux server with Intel(R) Xeon(R) CPU E5-2690 v3 2.60GHz, 57GB RAM, and K80 GPU. We implement algorithms with Python and PyTorch 1.9.0+cu102. All collaborative learning clients are trained sequentially due to hardware limitations. We trained client models for 300 iterations with no early-stop. The total training iteration is empirically selected to ensure the model is convergent. For each training iteration, a client trains the client model with the private data for one epoch. Batch size is set according to the total number of training samples. Specifically, we set batch size to 32 and 256 for the CIFAR-10/CIFAR-100 and EMNIST experiments, respectively. Stochastic Gradient Descent with momentum 0.9 and learning rate 0.001 are used.

Dataset split. We distribute 80%/10% of the samples to the clients to train/validate the client models. The remaining 10% of samples are dispatched to the moderator for monitoring the performance of server model. We use classification accuracy for performance evaluation metric. We compare the accuracy of the server model trained with different algorithms on the test set of the original dataset. We use classification accuracy for performance evaluation metric.

Non-IID sample distributions. We distribute the training samples to each client via Dirichlet distribution to simulate a real world non-IID data distribution [12, 13, 14]. Fig. 3 shows examples of the sample distributions with different μ's. Generally, small μ values indicates a highly unbalanced data distribution among clients. We set μ to 0.1 to test the algorithm's ability to handle a difficult situation.

4.1 Model Initialization

We initialize the data encoder $\mathcal{E}$ using the encoder of a pre-trained convolutional autoencoder [15]. The synonym generator $\mathcal{G}$ and the server model $\mathcal{M}$ are randomly initialized. All clients share a single version of $\mathcal{E}$ in CLSyn. This way, digests generated from different client data can map to the same digest space, even for non-IID client data distributions. $\mathcal{G}$ is securely stored at the moderator and not shareable with any client. This ensures private data protection, as data synonyms are only accessible by the moderator.

4.2 Training Without Client Leaving

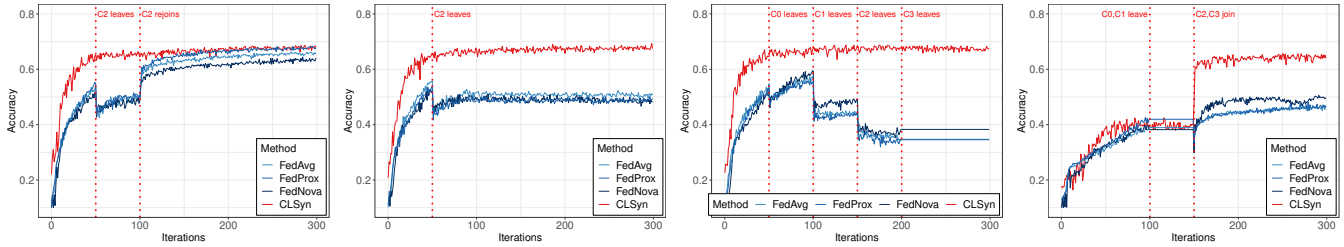
We first investigate the scenario without client leaving. We sample the distribution of client data by Dirichlet distribution with $\mu = 0.1$ for five repeat experiments, and compute the average testing accuracy of the latest 30 iterations. In this experiment, FedAvg, FedProx, FedNova, CLSyn, and centralized training achieve 0.69 (0.013), 0.69 (0.016), 0.69 (0.021), 0.68 (0.008), and 0.78 (0.005) accuracy, respectively. Values in the parentheses are the standard deviations of the five experiments, which shows the consistency of different methods. The highest accuracy of centralized training represents the cap of all decentralized training methods. Although the CLSyn test accuracy is slightly lower than the baseline methods (1%), performance improvement of CLSyn is significant in the tested client leaving scenarios that we will show next. The consistency of CLSyn is comparable to the centralized training and outperforms other algorithms.

4.3 Impact of Client Leaving

We target four client absence scenarios on CIFAR-10 for experiments: (1) the largest client (who stores the greatest number of samples) leaves temporarily; (2) the largest client leaves forever; (3) all clients leave sequentially; and (4) a pair of clients join the training in the beginning, and another pair of clients join later. Experiments are conducted with $\mu = 0.1$ to compare collaborative learning algorithms in highly-skewed non-IID scenarios, where client data distributions are shown in Fig. 3b. Specifically, client 0, 1, 2, 3 own unbalanced 13.5%, 32.6%, 46.4%, and 8.6% of training samples, respectively.

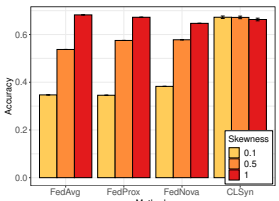
Fig. 4 shows the results of the four scenarios on CIFAR-10, where the *test accuracy* at each iteration illustrates the performance impact caused by client leaving. In Fig. 4a, client 2 leaves and rejoins training at iteration 50 and 100, respectively. Observe that CLSyn training curve is almost unaltered, while the test accuracy of the other curves drops significantly when client 2 leaves. This is because the server models trained by the baseline algorithms suffer from catastrophic knowledge forgetting due to the non-IID training samples caused by client absence. The final accuracy of the baseline algorithms is inferior to CLSyn, which indicates that temporary client leaving can cause permanent damage to the final trained model.

Fig. 4b and Fig. 4c show results for permanent client leaving and sequential client leaving until all clients are gone. The accuracy of CLSyn shows no large difference

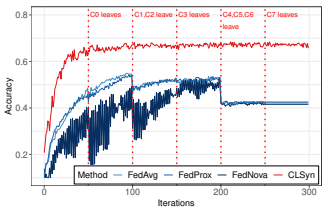


(a) Leaving temporarily (b) Leaving forever (c) Leaving sequentially (d) Group training

Figure 4: Comparison of CLSyn *vs.* baseline methods for the four targeted client absence scenarios on CIFAR-10.

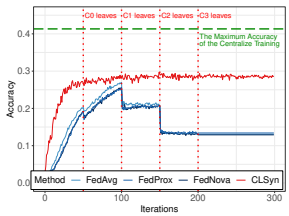


(a)

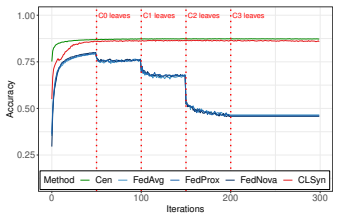


(b)

Figure 5: The impacts of (a) label skewness and (b) number of users on CIFAR-10.



(a)



(b)

Figure 6: Sequential leaving scenarios on (a) CIFAR-100 and (b) EMNIST. Green horizontal dash line shows the cap of maximum accuracy from centralized training.

between the two scenarios, which demonstrates the effectiveness of digests and synonyms against client leaving.

Fig. 4d shows group training at two stages. Specifically, clients 0 and 1 start the collaborative training and leave at iteration 100. Clients 2 and 3 then continue training at iteration 150. Note that there is no client joining training between iterations 100 and 150, and baseline algorithms have a straight horizontal line of accuracy (no model update). During the first 100 iterations, all algorithms perform poorly (0.4 testing accuracy) due to partial and biased data distribution. Once the new samples join in the training, CLSyn benefits largely from the stored digests and synonyms, which greatly boosts the server model as if clients 0 and 1 are still available.

4.4 Label Skewness and Number of Users

We investigate the effectiveness of CLSyn in handling different levels of label skewness, by setting the Dirichlet distribution hyperparameter μ as 0.1, 0.5, and 1.0. Fig. 5(a) shows the results. We observe a stable classification accuracy of the server model trained with CLSyn under different skewness. Models trained by the baselines suffer from accuracy degradation as the data skewness increases. The error bars represent standard deviations of the last 30 iterations.

We next test CLSyn on a eight-client scenario with sequential leaving. Specifically, one client leaves at iteration 50, then two leave at 100, one leaves at 150, three leave at 200, and one leaves at 250. Fig. 5(b) shows the result. Observe that the accuracy degradation has a positive correlation to the number of leaving clients. FedNova suffers from an unstable test accuracy during training. This is caused by the weights for aggregating the gradients at the moderator in the FedNova algorithm. The weight calculation not only involves the number of training samples but also the number of training steps, and the sum of weights is usually greater than one. Updating the model with a large gradient eventually causes an unstable learning curve during training. The situation becomes more severe when (1) the model is trained with a larger learning rate or (2) a larger number of clients join the collaborative learning. Overall, CLSyn outperforms all baseline algorithms by a large margin.

Table 1: Average test accuracy of different $\mathcal{G}$ initializations.

Dataset	Initial $\mathcal{G}$	Avg. Test Accuracy
CIFAR-10	Random	0.672 (0.005)
	Autoencoder	0.667 (0.007)
EMNIST	Random	0.861 (0.001)
	Autoencoder	0.868 (0.001)

4.5 Experiments on CIFAR-100 and EMNIST

To verify generalizability, we test CLSyn on two additional datasets CIFAR-100 and EMNIST. We keep the same network architectures as in the CIFAR-10 experiments, and only resize the EMNIST images from 28×28 to 32×32 and set the first conv kernel size to $3\times3\times1$. Fig. 6 shows the results. We observe a similar trend that CLSyn significantly outperforms other methods. The CIFAR-100 test accuracy is relatively low due to the insufficient network complexity. We observe similar overall trend as in CIFAR-10 experiments, that CLSyn can handle client leaving seamlessly.

In the EMNIST experiments, CLSyn significantly outperforms all baselines with surprising results that are almost as good as centralized training. The inspection of the synonyms generated from the digests provides some explanation. Fig. 7(b,d,f) shows examples of the private data and the synonyms generated from random-initialized

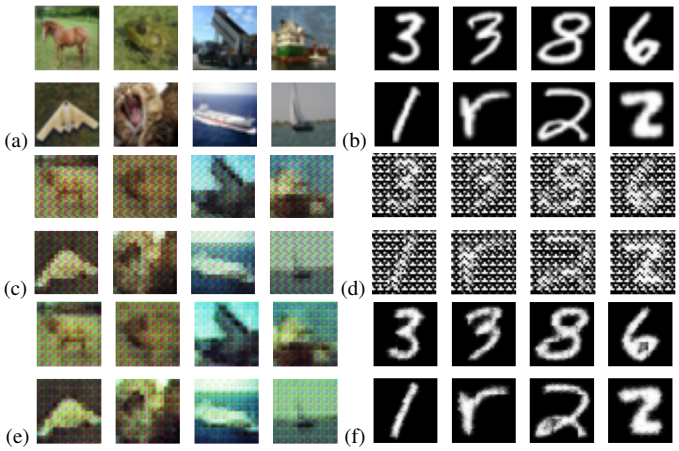


Figure 7: Examples of (a,b) client data samples, (c,d) corresponding synonyms learned from random initialization of $\mathcal{G}$, and (e,f) autoencoder produced synonyms, for results on (a,c,e) CIFAR-10 and (b,d,f) EMNIST, respectively.

$\mathcal{G}$ and autoencoder-initialized $\mathcal{G}$, respectively. Observe that the private data and the random-initialized synonyms are highly similar, which is caused by the shallow data encoder. The digests still preserve sufficient information to recover the original samples. As the training goes on, synonyms produced by $\mathcal{G}$ become similar to the original samples, and gradually the server model updates denominate the training. Thus, CLSyn training with a shallow data encoder can eventually approach centralized training.

4.6 Alternative $\mathcal{G}$ Initialization

To investigate how different initialization of the generator $\mathcal{G}$ can affect the synonym, we compare (i) random initialization and (ii) using the decoder network of the pre-trained autoencoder as the initialization in sequential leaving experiments. Fig. 7(c,d) show synonyms generated by randomly-initialized $\mathcal{G}$. Observe that $\mathcal{G}$ eventually converges to a state that generates synonyms resembling the private data in both CIFAR-10 and EMNIST datasets.

Fig. 7(e,f) show synonyms generated by autoencoder-initialized $\mathcal{G}$, which are visually similar to the original private data. While the synonyms are visually plausible, the preparation of such generator require a step that violates privacy concerns, as the autoencoder pre-training requires accessing to all private data of the clients. Table 1 shows the average test accuracy on the two different $\mathcal{G}$ initializations, where the accuracy has no significant difference. This result indicates the robustness of CLSyn training in learning to generate representation of the private data from digests.

5 Conclusion

We presented CLSyn, a robust learning framework using data digests and synonyms to address client absence in collaborative training. The key idea is to *memorize* training data as digests via a data encoder, and decode the digests into the synonyms as an alternative sample representation. CLSyn works well against unbalanced client data, which can significantly affect the training outcome. We test CLSyn in four client-leaving scenarios over three open datasets to investigate its effectiveness. Our results show that CLSyn outperforms FedAvg, FedProx, and FedNova in all scenarios by a significant margin.

Future Work includes further mitigation of the privacy concerns introduced from the shared digests and synonyms. This includes the exploration of principled methods that can de-identify the digests, such as pruning or merging to strengthen anonymity.

References

[1] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A survey on distributed machine learning. *ACM Comput. Surv.*, 53(2), Mar 2020.

[2] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. FDML: A collaborative machine learning framework for distributed features. In *SIGKDD*, page 2232–2240, 2019.

[3] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawit, Zachary Charles, Graham Cormode, and *et al. Advances and Open Problems in Federated Learning*. now publishers inc., 2021.

[4] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on Non-IID data silos: An experimental study. In *arXiv 2102.02079*, 2021.

[5] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of FedAvg on non-IID data. In *ICLR*, 2020.

[6] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2020.

[7] Jianyu Wang, Qianghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *NeurIPS*, 2020.

[8] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.*, 54, 2021.

[9] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *TR, U. Toronto*, 2009.

[11] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters, 2017.

[12] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *ICML*, pages 7252–7261, 09–15 Jun 2019.

[13] Qinbin Li, Bingsheng He, and Dawn Xiaodong Song. Practical one-shot federated learning for cross-silo setting. In *IJCAI*, 2021.

[14] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *ICLR*, 2020.

[15] Volodymyr Turchenko and Artur Luczak. Creation of a deep convolutional auto-encoder in Caffe. In *IDAACS*, pages 651–659, 2017.