MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MASTER THESIS

MASTER IN COMPUTATIONAL SCIENCE AND ENGINEERING

Scalable Collaborative Learning via Representation Sharing

Author:
Frédéric BERDOZ
Visiting from EPFL

Supervisors:
Ramesh RASKAR (MIT)
Martin JAGGI (EPFL)

Handover date:
July 15th, 2022

Abstract

Privacy-preserving machine learning has become a key conundrum for multi-party artificial intelligence. Federated learning (FL) and Split Learning (SL) are two frameworks that enable collaborative learning while keeping the data private (on device). In FL, each data holder trains a model locally and releases it to a central server for aggregation. In SL, the clients must release individual cut-layer activations (smashed data) to the server and wait for its response (during both inference and back propagation). While relevant in several settings, both of these schemes have a high communication cost, rely on server-level computation algorithms and do not allow for tunable levels of collaboration. In this work, we present a novel approach for privacy-preserving machine learning, where the clients collaborate via online knowledge distillation using a contrastive loss (contrastive w.r.t. the labels). The goal is to ensure that the participants learn similar features on similar classes without sharing their input data. To do so, each client releases averaged last hidden layer activations of similar labels to a central server that only acts as a relay (i.e., is not involved in the training or aggregation of the models). Then, the clients download these last layer activations (feature representations) of the ensemble of users and distill their knowledge in their personal model using a contrastive objective. For cross-device applications (i.e., small local datasets and limited computational capacity), this approach increases the utility of the models compared to independent learning and other federated knowledge distillation (FD) schemes, is communication efficient and is scalable with the number of clients. We prove theoretically that our framework is well-posed, and we benchmark its performance against standard FD and FL on various datasets using different model architectures.

Contents

1 Introduction

Motivated by concerns such as data privacy, large scale training and others, Machine Learning (ML) research has seen a rise in different types of collaborative ML techniques. Collaborative ML is typically characterized by an orchestrator algorithm that enables training ML model(s) over data from multiple owners without requiring them to share their sensitive data with untrusted parties. Some of the well known algorithms include Federated Learning (FL) [35], Split Learning (SL) [16] and Swarm Learning [52]. While the majority of the works in collaborative ML rely upon a centralized coordinator, in this work, we design a decentralized learning framework where the server plays a secondary role and could easily be replaced by a peer-to-peer network. As we show in the rest of the paper, the main benefit of our decentralized approach over centrally coordinated ML is the increased flexibility among clients in controlling the information flow across different aspects such as communication, privacy, computational capacity, data heterogeneity, etc.

Most existing collaborative learning schemes are built upon FL, where the training algorithm and/or the model architecture is usually imposed to the clients to match the computational capacity of the weaker participant (or the server). We refer to this property as *non-tunable* collaboration, since the degree of collaboration is mostly imposed by the server. While this is less of a problem in cross-silo applications (small number of data owners, large local datasets) since the participants can easily find consensus over the best training parameters, it can constitute a strong barrier for participation in cross-device applications (large number of users with small local datasets and low/heterogeneous computational capacity). Indeed, finding consensus becomes harder in that scenario. To alleviate this, we propose a new framework for *tunable* collaboration, i.e., where each participant has near total control over its data release, its model architecture and how the knowledge of other users should be integrated in its own personalized model.

Our main idea is to share learned feature representations of each class among users and to use these representations cleverly during local training. Since the clients can choose which features are aggregated and shared, our framework enables the clients to assign different privacy levels to different samples. Our decentralized approach also ensures that the overall system remains asynchronous and functions as expected even if all but two clients are offline in the whole system. Finally, our framework makes it convenient to account for model heterogeneity and model personalization, since every user can select a subset of peers based on their goals of generalization and personalization. While some of these advantages have been introduced in recent FL based schemes, our framework allows natural integration of such several ideas. Our work is different from the body of work done in FL due to its decentralized design. Specifically, our system does not use server-level computation algorithms that are directly involved in the training of the model. Nevertheless, for completeness, we experimentally compare it with FL in Section 4.

Our contribution can be summarized as follows:

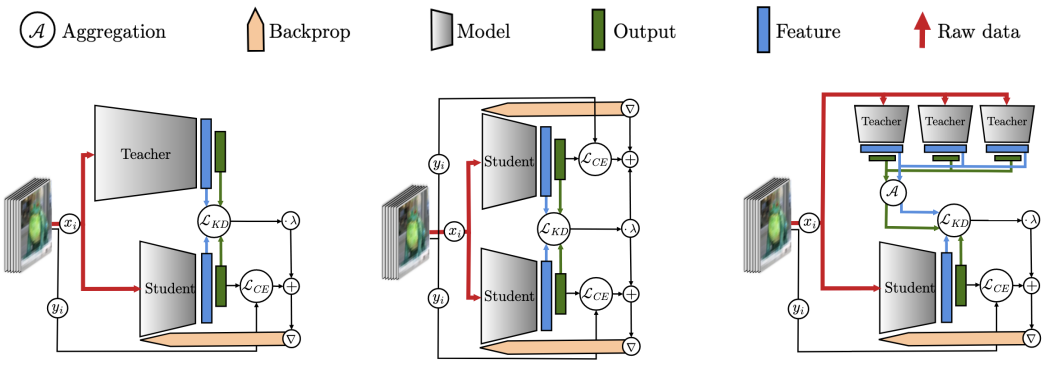
- We present a new *tunable* collaborative learning algorithm based on contrastive representation learning and online knowledge distillation.
- We prove theoretically that our local objective is well-posed.
- We show empirically the advantages of our framework against other similar schemes in terms of utility, communication, and scalability.

2 Related Work

Federated Learning FL is considered to be the first formal framework for collaborative learning. In their initial paper, McMahan et al. [35] introduce a new algorithm called `FedAvg`, in which each client performs several optimization steps on their local private dataset before sending the updated model back to the server for aggregation using weight averaging. While this approach alleviates the communication cost of the baseline collaborative optimization algorithm `FedSGD`, it also decreases the personalization capacity of the global model due to the naive model averaging, especially in heterogeneous environments. Several algorithms have been proposed to address these limitations. In particular, Li et al. [29] present `FedProx`, a slight improvement of `FedAvg` where partial updates are allowed. Arivazhagan et al. [4] introduce `FedPer`, in which the base layers are trained collaboratively using FL while each client stores and trains a personalization layer locally. Wang et al. [50] use `FedMa` to average the layers by looking at all possible permutations of neurons to find the best match. Sannara et al. [41] present `FedDist`, an aggregation algorithm that is able to modify the global model architecture by adding new neurons instead of averaging them (where deemed necessary). Wang et al. [51] introduce `FedNova`, an algorithm that aggregates the client models by normalizing them with the number of local updates in order to ensure objective consistency. Similarly, `Scaffold` (Karimireddy et al. [22]) and `VRL-SGD` (Liang et al. [31]) use gradient variance reduction among workers to achieve the same goal. Concerning the server update, Reddi et al. [40] introduce federated version of existing adaptive optimization algorithms like `Adagrad`, `Adam` and `Yogi`, and provide their corresponding convergence analysis. While improving the convergence rate, these algorithms suffer from the same constraints as `FedAvg`, i.e., homogeneous model architecture for every client, high communication overhead and *non-tunable* collaboration, all potential barriers for participation.

Fully Decentralized Learning The use of a central server in traditional FL constitutes a single point of failure and can also become a bottleneck when the number of clients grows, as shown by Lian et al. [30]. To alleviate these issues, Vanhaesebrouck et al. [48] formalize a new framework where each client participates in the learning task via a peer-to-peer network using gossip algorithms [42, 13]. In this configuration, there is no global solution and each client has its own personalized model, which enables both personalization and generalization. On the other hand, it creates other challenges about convergence, practical implementation and privacy [21]. Moreover, as in FL, the entire model must be released at every communication round, which can constitute a barrier for participation for the same reasons.

Knowledge Distillation The concept of knowledge distillation (KD) originated in Bucila et al. [8] as a way of compressing models, and was later generalized by Hinton et al. [18] (see Gou et al. [14] for an overview of the field). The standard use case for KD is that of a Teacher-Student (or



(a) Offline KD (b) Online KD (c) Ensemble offline KD

Figure 1: Different types of response-based (green) and feature-based (blue) knowledge distillation.

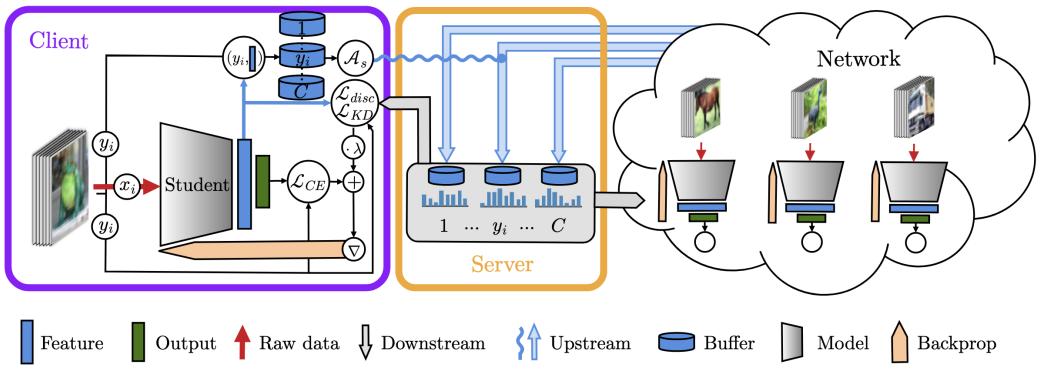


Figure 2: The proposed framework: clients exchange averaged ($\mathcal{A}_s$) feature representations for each class and use these representations in their own local training, but keep their raw data private (on device). The ensemble of participants (network) constitutes the teacher, and each single participant is a student. The server acts mainly as a relay as it does not take part in the training or aggregation of the models. At each communication round, the student downloads a subset of representations and uploads some of its own.

offline) configuration, in which the teacher model (usually a large and well-trained model) transfers its knowledge to the student model (usually a smaller model) by sharing its last layer activations on a given *transfer* dataset (see Fig. 1a). The knowledge is then *distilled* into the student model using a divergence loss between the teacher and student models outputs (response-based KD) or intermediate layers (feature-based KD) on the transfer dataset. Traditional KD schemes use a transfer set that is similar (or identical) to the teacher training dataset, but some recent work has focused on *data-free* (or *zero-shot*) KD. This can be achieved either by looking at some of the teacher model statistics to generate synthetic transfer data [34, 37, 6], or by training a GAN in parallel [36, 10, 2]. It has also been shown that positive results can be obtained using mismatched or random unlabeled data for the distillation [26, 38]. A key concept for our framework is the idea of online KD (or co-distillation [3], see Fig. 1b), where each model is treated as both a teacher and a student, meaning that the KD is performed synchronously with the training of each model (rather than after the training of the teacher model) [15, 55, 43]. Finally, ensemble KD (see Fig. 1c) refers to the setup where the knowledge is distilled from an ensemble of teacher (offline) or teacher-student (online) models.

Knowledge Distillation for Collaborative Learning A growing body of literature has recently investigated ways of using online KD for collaborative learning in order to alleviate the need of sharing model updates (FL) or individual smashed data (SL). Jeong et al. [20] present Federated Knowledge Distillation (FD), where each client uploads its mean (per class) logits to a central server, who aggregates and broadcasts them back. These soft labels are then used as the teacher output for the KD loss during local training. A closely related idea is to compute the mean logits on a common public dataset [28, 19, 9], but we argue that selecting this dataset can induce bias and is not always feasible, since additional trust is needed for its selection, and sufficient relevant data might be lacking. Besides FD, KD can enable collaborative learning in various ways: In an attempt to decrease the communication cost, Wu et al. [53] introduce `FedKD`, in which each client trains a (large) teacher network on their private data and transfer locally its knowledge to a smaller student model, which is then used in a standard FL algorithm. On the other hand, Lin et al. [32] and Chen and Chao [11] use KD on an unlabeled synthetic or public dataset to make the FL aggregation algorithm more robust. Our approach differs significantly from these schemes, as it does not rely on traditional FL algorithms.

3 Methods

Preamble and Motivation Consider any classification task on d-dimensional raw inputs with C distinct classes and a set of N participating users $\{u\}_{u=1}^{N}$, each with a local private dataset $\mathcal{D}_u$ and a model f_u:

$$\mathcal{D}_u := \{(\mathbf{X}_i, Y_i) \overset{iid}{\sim} p_u\}_{i=1}^{n_i}, \qquad p_u(\mathbf{x}, y) := p_{\mathbf{X},Y|U=u}(\mathbf{x}, y), \qquad f_u = \tau_u \circ \phi_u,$$

where $p_{\mathbf{X},Y,U}$ represents the joint probability of choosing a user U and drawing a sample $(\mathbf{X}, Y)$ from its distribution, and τ_u, ϕ_u represent the linear classifier and neural network (up to last hidden layer) of user u, respectively (with potentially different architectures across users). More precisely, let d' be the output dimension of ϕ_u and $\mathbf{w}_u = \{\boldsymbol{\theta}_u, \mathbf{W}_u, \mathbf{b}_u\}$ be model weights of user u. We have:

$$\begin{aligned} \phi_u : \mathbb{R}^d &\to \mathbb{R}^{d'} & \tau_u : \mathbb{R}^{d'} &\to \mathbb{R}^C \\ \mathbf{x} &\mapsto \mathbf{x}' := \phi_u(\mathbf{x}; \boldsymbol{\theta}_u) & \mathbf{x}' &\mapsto \mathbf{z} := \tau_u(\mathbf{x}'; \mathbf{W}_u, \mathbf{b}_u) = \mathbf{W}_u\mathbf{x}' + \mathbf{b}_u \end{aligned}$$

where $\mathbf{x}, \mathbf{x}'$ and $\mathbf{z}$ are the raw input, the feature representation and the logits, respectively. We motivate our approach as follows. Assuming no sharing of raw data $\mathbf{x}$ (for privacy concerns), any collaborative learning framework falls in one of two buckets: weight sharing or activation sharing. As mentioned in Section 2, sharing weights (e.g., FL) comes with strong constraints (communication, model architecture, etc.) and might not always be suitable. Concerning activation sharing (e.g., SL), it can be done at any layer (hidden or output). However, since activations are usually strongly correlated to the raw input [49], sharing single sample activations can raise privacy concerns. Instead, sharing averaged activations can easily be met with differential privacy guarantees (see privacy in Section 3). Due to the high non-linearity of neural networks, sharing averaged activations only makes sense at the output layer or last hidden layer (since the classifier τ_u is linear). Sharing only averaged outputs (averaged over samples of the same class) like in FD has been shown to have limited success as the quantity of shared information is restricted to the dimension of the output C, and we argue in this paper that sharing the feature representations (i.e., outputs of the last hidden layer, also averaged over samples of the same class) is more flexible and leads to better results. In this light, our objective is to collaboratively learn the best feature representation for each class, using contrastive (w.r.t. classes) representation learning [44] and feature-based knowledge distillation [14]. In other words, we want to learn collaboratively (i.e., only once) the structure of the feature representation space so that each client does not need to find it on its own with its limited amount of data and/or computational capacity.

Contrastive Objective In this section, we introduce a contrastive objective function for private online knowledge distillation (i.e., when users synchronously learn personalized models without sharing their raw input data and by collaborating via online distillation). This new objective function and its derivation are partly inspired by the offline contrastive loss presented in Tian et al. [44] and van den Oord et al. [46], with a few important differences. In the offline non-private setting, both the teacher and student models ϕ_t and ϕ_s have access to the same dataset $\{x_i\}_{i=1}^{n}$. In that scenario, the representations $\phi_s(x_i)$ and $\phi_t(x_j)$ are *pulled apart* if $i \neq j$ and are *pulled together* if $i = j$. However, in the private setting, ϕ_s does not have access to the raw input data that was used to train the teacher model, but has its own private dataset. At a given communication round and from the perspective of user u, we define ϕ_u as the student model and ϕ_U as the teacher model, where $U \sim p_U$ is a user selected at random. Consider the following procedure: u samples $Y \sim p_{Y|U=u}$ from its own data distribution and then samples either jointly (i.e., from the same observation of Y) or independently (from two independent observations of Y) the two random vectors $\boldsymbol{\Phi}_s$ and $\boldsymbol{\Phi}_t$ defined as follows:

$$\mathbf{X} \sim p_{\mathbf{X}|Y,U=u} \qquad \boldsymbol{\Phi}_s := \phi_u(\mathbf{X}), \tag{1}$$

$$U \sim p_U, \quad (\mathbf{X}_1, ..., \mathbf{X}_{n_{avg}}) \overset{iid}{\sim} p_{\mathbf{X}|Y,U}, \qquad \boldsymbol{\Phi}_t := \frac{1}{n_{avg}} \sum_{i=1}^{n_{avg}} \phi_U(\mathbf{X}_i). \tag{2}$$

The parameter n_{avg} defines over how many samples we take the average, which in turn defines the concentration of the distribution of $\boldsymbol{\Phi}_t$. From the (student) perspective of client u and in the

spirit of collaboration, the goal is to maximize the mutual information $\mathcal{I}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)$. Still from the perspective of u, let $p_{s,t}$, p_s and p_t be the joint and marginal distributions of $\mathbf{\Phi}_s$ and $\mathbf{\Phi}_t$, respectively, and let I be a Bernoulli random variable indicating if a tuple $(\mathbf{\Phi}_s, \mathbf{\Phi}_t)$ has been drawn from the joint distribution $p_{s,t}$ or from the product of marginals $p_s p_t$. Finally, let $q(\mathbf{s}, \mathbf{t}, i)$ be the joint distribution of $(\mathbf{\Phi}_s, \mathbf{\Phi}_t, I)$ such that $q(\mathbf{s}, \mathbf{t}|i = 1) = p_{s,t}(\mathbf{s}, \mathbf{t})$ and $q(\mathbf{s}, \mathbf{t}|i = 0) = p_s(\mathbf{s})p_t(\mathbf{t})$ and suppose that the prior $q(i)$ satisfy $q(i = 1) = \frac{1}{K+1}$ and $q(i = 0) = \frac{K}{K+1}$, i.e., for each sample from the distribution $p_{s,t}$, we draw K samples from the distribution $p_s p_t$. We can show the following bound.

Theorem 1. *Using the above notation, let $h(i, \mathbf{s}, \mathbf{t})$ be any estimate of $q(i|\mathbf{s}, \mathbf{t})$ with Bernoulli parameter $\hat{h}(\mathbf{s}, \mathbf{t})$, and let $\mathcal{L}_{disc}(h, \phi_u)$ be defined as follows:*

$$\mathcal{L}_{disc}(h, \phi_u) := -\mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_{s,t}} \left[\log \hat{h}(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \right] - K \mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_s p_t} \left[\log(1 - \hat{h}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)) \right]. \quad (3)$$

The mutual information $\mathcal{I}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)$ can be bounded as

$$\mathcal{I}(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \geq \log(K) - \mathcal{L}_{disc}(h, \phi_u), \quad (4)$$

with equality iff $h = q$ (better estimates lead to tighter bounds). The proof is joined in the supplementary material.

Hence, by minimizing $\mathcal{L}_{disc}$ in Eq. (4), we optimize a lower bound on the mutual information $\mathcal{I}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)$. Taking advantage of the classifier τ_u, a natural choice for h is the discriminator h_u with Bernoulli parameter

$$\hat{h}_u(\mathbf{s}, \mathbf{t}; \mathbf{W}_u, \mathbf{b}_u) = \langle \text{softmax}(\tau_u(\mathbf{s})), \text{softmax}(\tau_u(\mathbf{t})) \rangle. \quad (5)$$

With this choice, $\hat{h}_u(\mathbf{s}, \mathbf{t})$ can be interpreted as the estimated probability that the features $\mathbf{s}$ and $\mathbf{t}$ come from the same class. Note that in their work, Tian et al. [44] train an external discriminator (i.e., that is not defined using the model classifier τ_u).

Final Objective Intuitively, from the perspective of u, minimizing $\mathcal{L}_{disc}$ ensures that its classifier τ_u can distinguish if two feature representations, one local and one from another user, come from the same class. To improve the algorithm convergence and to ensure that τ_u can classify the feature representation $\mathbf{\Phi}_t$ of another client (similar as in Invariant Risk Minimization [5]), we also introduce a classical feature-based KD term $\mathcal{L}_{KD}$. This term minimizes the L_2 distance between the local and global feature representations of a same class (we define the global representation of class c as the expected value $\mathbb{E}_{(X,U) \sim p_{X,U|Y=c}}[\phi_U(\mathbf{X})]$). An important distinction between $\mathcal{L}_{KD}$ and $\mathcal{L}_{disc}$ is that the first one uses an *inter*-client averaged representation, whereas the second one uses an *intra*-client averaged representation. Combining $\mathcal{L}_{KD}$ and $\mathcal{L}_{disc}$ with the standard cross-entropy loss $\mathcal{L}_{CE}(\tau_u, \phi_u)$ using the meta parameters λ_{KD} and λ_{disc}, the final optimization problem of u becomes:

$$\text{Find} \quad \boldsymbol{\theta}_u^\star, \mathbf{W}_u^\star, \mathbf{b}_u^\star = \underset{\boldsymbol{\theta}_u, \mathbf{W}_u, \mathbf{b}_u}{\text{argmin}} \; \mathcal{L}_{CE}(\tau_u, \phi_u) + \lambda_{KD} \mathcal{L}_{KD}(\phi_u) + \lambda_{disc} \mathcal{L}_{disc}(\hat{h}_u, \phi_u). \quad (6)$$

Implementation With the standard assumption that the datasets are composed of IID samples drawn from their corresponding distributions, the expected losses $\mathcal{L}_{CE}$, $\mathcal{L}_{KD}$ and $\mathcal{L}_{disc}$ can be approximated by their unbiased mini-batch estimators L_{CE}, L_{KD} and L_{disc}, respectively. For one given sample, the loss functions are given by $\ell_{CE}(\mathbf{z}, c) := -\log\left(\text{softmax}(\mathbf{z})\right)_c$, $\ell_{KD}(\mathbf{x}', \mathbf{x}'') := \|\mathbf{x}' - \mathbf{x}''\|^2$ and finally

$$\begin{aligned} \ell_{disc} : \{0, 1\} \times \mathbb{R}^{d'} \times \mathbb{R}^{d'} &\to \mathbb{R}_+ \\ (i, \mathbf{s}, \mathbf{t}) &\mapsto -i \log\left(\hat{h}(\mathbf{s}, \mathbf{t})\right) - (1 - i) \log\left(1 - \hat{h}(\mathbf{s}, \mathbf{t})\right). \end{aligned} \quad (7)$$

Finally, for the sampling procedure of the contrastive objective, we use the most intuitive scheme: for every training sample $(\mathbf{x}_i, y_i)$ with feature representation $\mathbf{s}_i = \phi_u(\mathbf{x}_i)$, we use one observation of $\mathbf{\Phi}_t$ sampled using y_i (i.e., $I = 1$), and one observation of $\mathbf{\Phi}_t$ sampled using each $c \neq y_i$ (i.e., $I = 0$). Thus, we have $K = C - 1$. With these considerations, a detailed description of our collaborative learning framework can be found in the supplementary material.

Communication In terms of communication, the uplink and downlink volumes are of order $\mathcal{O}((M_\uparrow + 1)Cd')$ and $\mathcal{O}(N(M_\downarrow + 1)Cd')$ per round, where $M_\uparrow$ and $M_\downarrow$ represent the number of $\mathbf{\Phi}_t$ realizations (per class) that are uploaded and downloaded by the clients, respectively (these parameters can be tuned to match the communication capacity of the network). For comparison, these volumes are of order $\mathcal{O}(D)$ and $\mathcal{O}(ND)$ for FL (with D the model size) and $\mathcal{O}(nd')$ and $\mathcal{O}(Nnd')$ for SL. Because in most scenarios $D \gg n \gg d' \gg C$ and since $M_\uparrow$ and $M_\downarrow$ are tunable, we observe that our framework is communication efficient compared to FL and SL.

Relaxation to peer-to-peer We emphasize here that our contrastive objective $\mathcal{L}_{disc}$ is fully compatible with a peer-to-peer configuration, since the server only acts as a relay for the observations of $\mathbf{\Phi}_t$. For the traditional feature-based KD objective $\mathcal{L}_{KD}$, we use one global representation per class (i.e., one for the whole network), which in theory can only be computed by a central entity. One way to alleviate this could be to use the average of all the observations of $\mathbf{\Phi}_t$ that were downloaded by user u as a proxy for the global feature representations. However, to focus solely on the effectiveness of the proposed objectives rather than the topology of the network, we only present experiments in which a central entity computes the global representations.

Privacy Our framework naturally attains a certain degree of privacy by sharing representations from the penultimate layer, which, as per information bottleneck principle [45], has minimal mutual information from the original data. Although such a level of privacy may not be enough from a membership attack point of view, we emphasize that the privacy is amplified by averaging out these representations, hence amplifying the privacy of individual samples. To further make our framework formally private, we propose using recent progress in differentially private mean estimation [33, 7] to obtain a differentially private mean before it gets shared with other clients. Furthermore, our framework naturally dovetails with recent works [24, 1] in variable privacy levels for different samples. The users can assign variable levels of privacy to different samples before they get aggregated in a privacy preserving manner.

4 Experiments

Datasets, models and training We run several experiments with the MNIST [12], Fashion-MNIST [54] and CIFAR10 [25] datasets. For MNIST, we use a simple convolutional neural network (CNN) similar to LeNet5 ($\approx 30K$ parameters) [27] and for Fashion-MNIST and CIFAR10, we use ResNet9 ($\approx 2,4M$ parameters) and ResNet18 ($\approx 11.3M$ parameters) architectures [17], respectively. For the dimension of feature representation space, we set $d' = 84$ for LeNet5, $d' = 128$ for ResNet9 and $d' = 256$ for ResNet18. In order to simulate a scenario where the data is sparse, we only select a fraction of the train dataset (1200 samples for MNIST, 6000 for Fashion-MNIST and 10000 for CIFAR10) that we split uniformly at random across $N \in \{2, 5, 10\}$ users. For the validation, we use the entire test dataset for each task (10000 samples). In order to have fair comparisons, we train all the models for the same number of communication rounds, and we stop the training as soon as framework has reasonably converged ($r = 100$ for MNIST/LeNet5, $r = 20$ for Fashion-MNIST/ResNet9 and CIFAR10/ResNet18). We compare our framework with centralized learning (or CL, i.e., with $N = 1$ and $\lambda_{KD} = \lambda_{disc} = 0$), independent learning (or IL, i.e., with $\lambda_{KD} = \lambda_{disc} = 0$), federated learning using `FedAvg` (FL) and federated knowledge distillation (FD). We use the default learning rate $\eta = 10^{-3}$ for all experiments and we perform 1 local epoch of training per communication round. For CL, IL, FD and our framework, we use the `Adam` stochastic optimization algorithm [23]. Finally, supported by Fig. 3, we set $\lambda_{KD} = 10$ and $\lambda_{disc} = 1$ in our final objective (Eq. (6)).

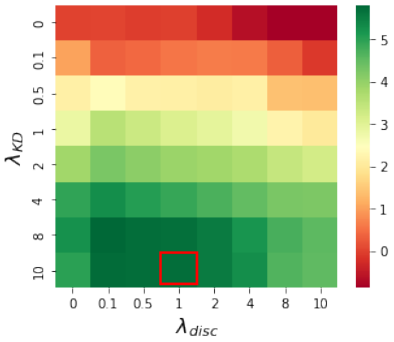


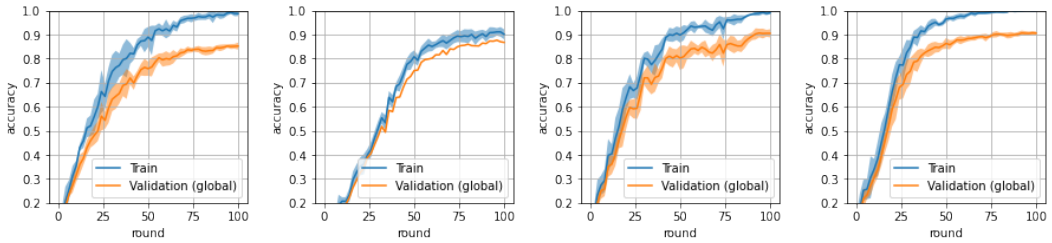
Figure 3: Ablation study for λ_{KD} and λ_{disc}. Average test accuracy improvement [%] w.r.t. IL (upper left corner) when different combinations of λ_{KD} and λ_{disc} are used (MNIST/LeNet5 experiment with 5 users and 100 communication rounds). The red square indicates the value of λ_{KD} and λ_{disc} used in all our experiments.

Network emulation For our contrastive objective $\mathcal{L}_{disc}$, we use $n_{avg} = 10$ (i.e., for every class, each user selects n_{avg} samples of that class, computes and averages their feature representations, uploads them to the server and downloads the representations of another user chosen at random). For the feature-based KD objective $\mathcal{L}_{KD}$, each client average the feature representation of all the samples of a same class and uploads these averaged representations to the server, which in turn averages them to obtain one global representation per class. For simplicity, we use $M_{\uparrow} = M_{\downarrow} = 1$ (clients upload and download one observation of $\mathbf{\Phi}_t$ for each class).

Software and hardware We use the CUDA enabled PyTorch library [39] with a GTX 1080 Ti (11GB) GPU and an Intel(R) Xeon(R) CPU (2.10GHz). The codebase related to this work can be found on the github repository `https://github.com/fberdoz/MIT-master-thesis`.

5 Discussion

Utility As seen in Table 1, our framework outperforms every other framework for when a small model is used (MNIST/LeNet5), especially when the number of clients grows, which is typically the kind of configuration that would be relevant for a cross-device application). In that setup, FL performs particularly poorly as it struggles to find a low-capacity model that matches the data distribution of each client. This is particularly visible on Fig. 4b. Although in that configuration, FD shows similar performance for small number of clients ($N = 2, 5$), it still exhibits a lower rate of convergence (compare Fig. 4c and Fig. 4d). Our framework even outperforms centralized learning (CL) when $N = 2$, suggesting that the added objectives can also be seen as regularizers. For the Fashion-MNIST dataset, our framework shows significant improvement over IL and FD (i.e., frameworks where there are no global model), but is not able to compete with FL anymore, which in that case even outperforms CL. However, the comparison between FL and our framework is unfair for large models due to the amounts of shared information (e.g., for ResNet9, one communication round of our framework communicates ≈ 1000 times fewer bits than standard FL. This ratio increases to ≈ 2000 for ResNet19). Similar conclusions can be drawn for the CIFAR10 experiments. However, in that case, even IL outperforms our framework for $N = 10$ after $r = 20$ rounds. Indeed, since our objective function is highly complex (the global minimum depends on the data of other peers), the algorithm struggles to converge to the optimal model when the number of parameters is very large. However, we argue that this is not critical since training ResNet18 models on low capacity devices is not particularly adapted to cross-device applications, as they usually require powerful GPUs and large datasets to converge.



(a) IL (b) FL (c) FD (d) Ours

Figure 4: Comparison of the train and test accuracy during 100 communication rounds of IL, FL, FD and our framework on the MNIST dataset with LeNet5 architectures and $N = 5$ users. The shaded areas represent $\pm$ the standard deviation of the metric across clients. For the validation, each model is tested using the entire test dataset.

Table 1: Average test accuracy over clients $[\%]$ of the different frameworks after r communication rounds when the same amount of data is divided uniformly at random between N users. We use 1200 training samples for MNIST, 6000 for Fashion-MNIST and 10000 for CIFAR10. The validation is done using the full test dataset (10000 samples for each task).

	MNIST ($r = 100$)			Fashion-MNIST ($r = 20$)			CIFAR10 ($r = 20$)		
CL	94.00			87.77			66.15		
	$N = 2$	$N = 5$	$N = 10$	$N = 2$	$N = 5$	$N = 10$	$N = 2$	$N = 5$	$N = 10$
FL	92.64	86.79	70.06	89.79	89.28	88.21	67.99	59.18	51.05
IL	91.46	85.26	72.86	86.04	83.61	80.52	59.85	46.46	**38.51**
FD	**94.45**	90.55	77.90	87.17	83.32	79.44	56.75	44.91	31.43
Ours	94.19	**90.63**	**82.07**	**87.91**	**84.44**	**80.77**	**63.49**	**47.28**	37.78

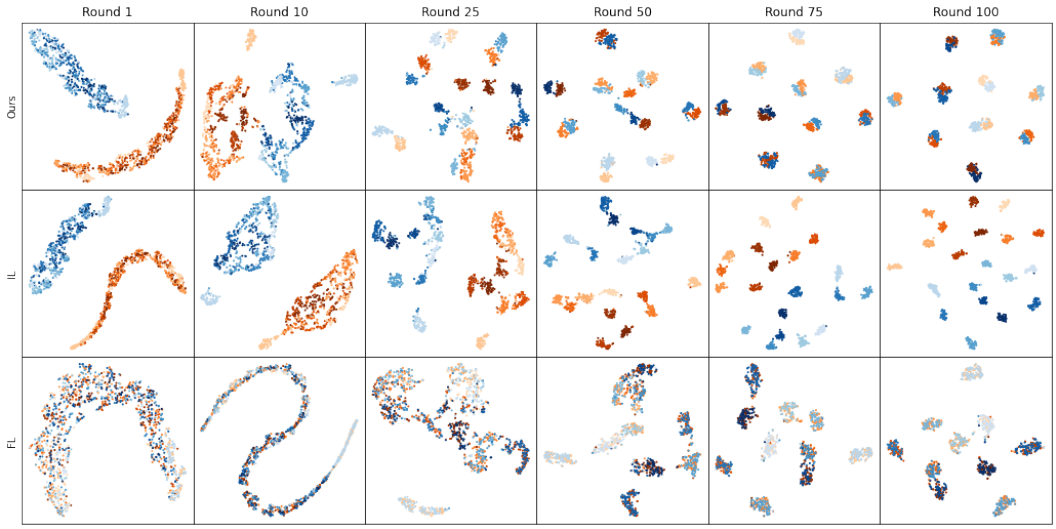


Figure 5: T-distributed Stochastic Neighbor Embedding [47] of the feature representation at different rounds in a two party configuration for the MNIST/LeNet5 experiment. Clients are distinguished by color and classes by opacity.

Effect of $\mathcal{L}_{KD}$ The feature-based KD term $\mathcal{L}_{KD}$ makes sure that similar feature representations are learned by each client. Fig. 5 provides insight into the evolution of the latent feature space during training. In FL, the structure of the latent feature space is imposed from the start to each client, whereas in our framework, clients have more freedom to learn the best structure that fits their data, since they are only *guided* towards a global representation structure. This clearly improves convergence, as we can see that clusters are already formed at round 25 with our algorithm, whereas with FL the latent feature space is still relatively chaotic at that point. On the other hand, we observe that for IL, the learned latent feature space is totally different for each client as they learn it by themselves, and the convergence rate is also notably slower (compare round 25 for instance).

Effect of $\mathcal{L}_{disc}$ The effect of the contrastive objective is more subtle, but can still improve the test accuracy significantly if tuned properly. In fact, whereas $\mathcal{L}_{KD}$ can be used without $\mathcal{L}_{disc}$, the reciprocal is not true as observed on Fig. 3. This is because we have defined the discriminator of user u (i.e., $\hat{h}_u$) using its feature representation classifier τ_u (see Eq. (5)). Hence, in order to generate useful gradients for back propagation (i.e., that do not look like noise), τ_u must be able to classify relatively well the feature representations that are generated by other models. We have also tested external discriminators (like in [44]), but the results were not as positive.

Limitations and Future work As shown by the results, our approach is particularly relevant when the model capacity is relatively small, but struggles when the model complexity grows, especially when compared with FL. However, the comparison is unfair in that scenario, as the amount of information that is shared between clients at each round is larger by several orders of magnitudes for FL. We also posit that the convergence gap could be reduced by adding sophistication into the algorithm (e.g., by having a local algorithm that selects the most relevant observed representations from other users, by only sharing a fraction of the feature representation dimensions, by introducing a more complex discriminator h, etc.). Another important limitation is the fact that our framework cannot be used for supervised tasks other than classification, as the feature representations need to be sorted by class. This is harder to alleviate since the whole system's architecture is build upon this requirement. Future work should also investigate how the number of classes and their distributions might influence the performance of the algorithm, and also if (and how) it could be used for unsupervised learning.

6 Conclusion

We introduce a new collaborative learning algorithm that enables tunable collaboration in cross-device applications and whose uplink and downlink communication does not scale with the model size (as in FL) or the dataset size (as in SL). We prove that our objective is well posed from the point of view of collaboration, as it maximizes a lower bound on the mutual information between the feature representations of different users across the network. Then, we show empirically that it is particularly relevant in setups where the number of clients is large and when each of them have limited computational resources. In that scenario, it converges faster than IL, FD and even standard FL. Due to its simplicity and modularity with other potential requirements (such as differential privacy, server free/peer-to-peer configurations, heterogeneous model architectures, etc.), we posit that our algorithm could be implemented in numerous settings where FL (or other similar frameworks) is not suitable, as long as careful task-specific deployment considerations are made.

Acknowledgments and Disclosure of Funding

I would like to show my gratitude for the supervision of Prof. Ramesh Raskar and Prof. Martin Jaggi during my time at MIT. My stay would not have been possible without them. Besides their guidance, this work was also strongly inspired by the many discussions I had with Abhishek Singh, Mohammad Mohammadi Amiri, Vivek Sharma, Kushagra Tiwary and all the other members of the Camera Culture group. I would also like to thank the Hasler Foundation for their financial support during my stay in the United States (project number 22008). Finally, as this works concludes my academic studies for now, I wish to extend my thanks to my family and friends for their constant support throughout this long journey. Et par-dessous tout, je remercie infiniment mes parents pour m'avoir donné l'éducation et les moyens de réaliser ce parcours académique.

Frédéric Berdoz
Boston, 14th of August 2022

References

[1] Jayadev Acharya, Kallista Bonawitz, Peter Kairouz, Daniel Ramage, and Ziteng Sun. Context aware local differential privacy. In *International Conference on Machine Learning*, pages 52–62. PMLR, 2020.

[2] Sravanti Addepalli, Gaurav Kumar Nayak, Anirban Chakraborty, and Venkatesh Babu Radhakrishnan. Degan: Data-enriching gan for retrieving representative samples from a trained classifier. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3130–3137, 2020.

[3] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. *CoRR*, abs/1804.03235, 2018. URL http://arxiv.org/abs/1804.03235.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019. URL http://arxiv.org/abs/1912.00818.

[5] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2019.

[6] Kartikeya Bhardwaj, Naveen Suda, and Radu Marculescu. Dream distillation: A data-independent model compression framework, 2019. URL https://arxiv.org/abs/1905.07072.

[7] Gavin Brown, Marco Gaboardi, Adam Smith, Jonathan Ullman, and Lydia Zakynthinou. Covariance-aware private mean estimation without private covariance estimation. *Advances in Neural Information Processing Systems*, 34, 2021.

[8] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL https://doi.org/10.1145/1150402.1150464.

[9] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer, 2019. URL https://arxiv.org/abs/1912.11279.

[10] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3514–3522, 2019.

[11] Hong-You Chen and Wei-Lun Chao. Feddistill: Making bayesian model ensemble applicable to federated learning. *CoRR*, abs/2009.01974, 2020. URL https://arxiv.org/abs/2009.01974.

[12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[13] Alexandros G. Dimakis, Soummya Kar, José M. F. Moura, Michael G. Rabbat, and Anna Scaglione. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, 2010. doi: 10.1109/JPROC.2010.2052531.

[14] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[15] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. Online knowledge distillation via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[19] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *CoRR*, abs/2008.06180, 2020. URL https://arxiv.org/abs/2008.06180.

[20] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *CoRR*, abs/1811.11479, 2018. URL http://arxiv.org/abs/1811.11479.

[21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[22] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Ios Kotsogiannis, Stelios Doudalis, Sam Haney, Ashwin Machanavajjhala, and Sharad Mehrotra. One-sided differential privacy. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 493–504. IEEE, 2020.

[25] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[26] Mandar Kulkarni, Kalpesh Patil, and Shirish Subhash Karande. Knowledge distillation using unlabeled mismatched images. *CoRR*, abs/1703.07131, 2017. URL http://arxiv.org/abs/1703.07131.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 12 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

[28] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *CoRR*, abs/1910.03581, 2019. URL http://arxiv.org/abs/1910.03581.

[29] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[30] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[31] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local SGD with lower communication complexity. *CoRR*, abs/1912.12844, 2019. URL http://arxiv.org/abs/1912.12844.

[32] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[33] Xiyang Liu, Weihao Kong, Sham Kakade, and Sewoong Oh. Robust and differentially private mean estimation. *Advances in Neural Information Processing Systems*, 34, 2021.

[34] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *CoRR*, abs/1710.07535, 2017. URL http://arxiv.org/abs/1710.07535.

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

[36] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, pages 4743–4751. PMLR, 2019.

[38] Gaurav Kumar Nayak, Konda Reddy Mopuri, and Anirban Chakraborty. Effectiveness of arbitrary transfer sets for data-free knowledge distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1430–1438, January 2021.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[40] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020. URL https://arxiv.org/abs/2003.00295.

[41] EK Sannara, François Portet, Philippe Lalanda, and VEGA German. A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison. In *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2021.

[42] Devavrat Shah. Gossip algorithms. *Found. Trends Netw.*, 3(1):1–125, jan 2009. ISSN 1554-057X. doi: 10.1561/1300000014. URL https://doi.org/10.1561/1300000014.

[43] Shagun Sodhani, Olivier Delalleau, Mahmoud Assran, Koustuv Sinha, Nicolas Ballas, and Michael G. Rabbat. A closer look at codistillation for distributed training. *CoRR*, abs/2010.02838, 2020. URL https://arxiv.org/abs/2010.02838.

[44] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgpBJrtvS.

[45] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[46] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[47] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[48] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. *CoRR*, abs/1610.05202, 2016. URL http://arxiv.org/abs/1610.05202.

[49] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 933–942. IEEE, 2020.

[50] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *CoRR*, abs/2002.06440, 2020. URL https://arxiv.org/abs/2002.06440.

[51] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[52] Stefanie Warnat-Herresthal, Hartmut Schultze, Krishnaprasad Lingadahalli Shastry, Sathyanarayanan Manamohan, Saikat Mukherjee, Vishesh Garg, Ravi Sarveswara, Kristian Händler, Peter Pickkers, N Ahmad Aziz, et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594 (7862):265–270, 2021.

[53] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation". *Nat Commun*, 13(1):2032, apr 2022.

[54] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

[55] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328, 2018.

A Appendix

A.1 Notation

- N: Number of participating users/clients/peers/data owners.
- d: Raw input data dimension (e.g., $d = 3072$ for 32×32 RBG images).
- d': Latent feature (or feature representation) space dimensionality (i.e., width of last hidden layer).
- C: Number of classes.
- $(\mathbf{X}, Y) \in \mathcal{X} \times \mathcal{Y}$: Labeled data, with $(\mathbf{x}, y)$ a realization/observation. For classification tasks (and most of this work), $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, ..., C-1\}$.
- $p_{\mathbf{X},Y,U} : \mathcal{X} \times \mathcal{Y} \times \{1, ..., N\} \to \mathbb{R}_+$: Joint probability distribution across users.
- $p_u(\mathbf{x}, y) := p_{\mathbf{X},Y|U=u}(\mathbf{x}, y)$: Data distribution of user u.
- $\mathcal{D}_u := \{(\mathbf{X}_i, Y_i) \overset{iid}{\sim} p_u\}_{i=1}^{n_u}$: Dataset of user u, with $D_u := \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_i}$ its realization.
- $\mathcal{F}$: Set of all the possible models for the task at hand (achievable or not).
- $\mathcal{F}_u \subset \mathcal{F}$: Selected (by u) subset of parametrized models with the same architecture.
- $\mathbf{w}_u := \{\boldsymbol{\theta}_u, \boldsymbol{W}_u, \mathbf{b}_u\} \in \mathcal{W}_{\mathcal{F}_u}$ with $\boldsymbol{\theta}_u \in \Theta_u, \boldsymbol{W}_u \in \mathbb{R}^{C \times d'}, \mathbf{b}_u \in \mathbb{R}^C$: Parameters of the neural network of u, with $\mathcal{W}_{\mathcal{F}_u}$ and Θ_u the set of achievable model parameters of $\mathbf{w}_u$ and $\boldsymbol{\theta}_u$, respectively.
- $\phi_u : \mathbb{R}^d \to \mathbb{R}^{d'}, \ \mathbf{x} \mapsto \mathbf{x}' = \phi(\mathbf{x}; \boldsymbol{\theta}_u)$: Neural network of u (or similar parameterized function) that maps a raw input into a latent feature space.
- $\tau_u : \mathbb{R}^{d'} \to \mathbb{R}^C, \ \mathbf{x}' \mapsto \mathbf{z} = \tau(\mathbf{x}'; \mathbf{W}_u, \mathbf{b}_u) := \mathbf{W}_u \mathbf{x}' + \mathbf{b}_u$: Linear classifier of u.
- $f_u := \tau_u \circ \phi_u \in \mathcal{F}_u$: Full model of user u, parametrized by $\mathbf{w}_u \in \mathcal{W}_{\mathcal{F}_u}$.
- $\lambda_{KD}, \lambda_{disc}, n_{avg}, M_\uparrow, M_\downarrow$: Hyperparameters.
- η: Learning rate.
- $\ell_{CE}, \ell_{KD}, \ell_{disc}$: Cross-entropy, feature-based KD and discriminator loss functions, respectively.
- $\mathcal{L}_{CE}, \mathcal{L}_{KD}, \mathcal{L}_{disc}$: True risk (i.e., expected value (over the data) of ℓ_{CE}, ℓ_{KD} and ℓ_{disc}, respectively).
- L_{CE}, L_{KD}, L_{disc}: Empirical risk corresponding to $\mathcal{L}_{CE}, \mathcal{L}_{KD}$ and $\mathcal{L}_{disc}$, respectively.
- $\boldsymbol{\Phi}_s, \boldsymbol{\Phi}_t$: Random vectors (feature representations) of the student (user u) and the teacher (random user U).
- $p_{s,t}, p_s, p_t$: Joint and marginal distributions of $\boldsymbol{\Phi}_s$ and $\boldsymbol{\Phi}_t$, respectively.
- q: Joint distribution of $\boldsymbol{\Phi}_s, \boldsymbol{\Phi}_t$ and I, where I is a binary random variable indicating if $\boldsymbol{\Phi}_s, \boldsymbol{\Phi}_t$ has been drawn from $p_{s,t}$ ($I = 1$) or $p_s p_t$ ($I = 0$).
- h: Binary discriminator with Bernoulli parameter $\hat{h}$ (i.e., learnable estimate of $q_{I|\boldsymbol{\Phi}_s,\boldsymbol{\Phi}_t}$).
- $\mathbf{s}, \mathbf{t}$: Observation/realization of $\boldsymbol{\Phi}_s$ and $\boldsymbol{\Phi}_t$, respectively.
- $\bar{\mathbf{t}}^c, \mathbf{t}^c$: Global (i.e., using all the samples across users) and local (i.e., using n_{avg} samples) average feature representations of class c, respectively.

A.2 Formalizing Collaborative Learning

Independent Learning To see why collaborative learning might be useful in practical applications, we first recall the problem statement and the main assumptions for independent learning (i.e., traditional non-collaborative ML). Let $\mathcal{F}$ be the set of all models for the task at hand, and let u be an isolated user who wants to fit a model $f \in \mathcal{F}$ to its data $(\mathbf{X}, Y) \sim p_u$. Ideally, this model would converge to $p_u(y|\mathbf{x})$ (as it is the best possible model in $\mathcal{F}$), so that the regression (continuous Y) or classification (discrete Y) on a new observation $\mathbf{x}$ can be done by simply computing

$$\hat{y} = \arg\max_y f(y|\mathbf{x}) \approx \arg\max_y p_u(y|\mathbf{x}).$$

While this is usually a good approach for classification, the distribution p_u can be very complex for regression and $p_u(y|\mathbf{x})$ might be too far from any computable model architecture $\mathcal{F}_u \subset \mathcal{F}$. This is why, in general, we skip the distribution estimation for the regression case, and we assume a model on the data itself (e.g., for the linear regression, we assume that the target Y is generated as follows: $Y = \boldsymbol{a}^\mathsf{T}\mathbf{X} + b + \varepsilon$ where ε is the unmodelled noise and $\mathbf{a}, b$ are the parameters to fit. In that scenario, the model $f_u(\mathbf{x}; \hat{\mathbf{a}}, \hat{b}) = \hat{\mathbf{a}}^\mathsf{T}\mathbf{x} + \hat{b}$ fits the relation between $\mathbf{X}$ and Y, not their conditional distribution). However, to simplify the formalization, we only consider conditional distribution estimation in this work, i.e. where one tries to fit $p_u(y|\mathbf{x})$. The standard approach to fit such a model goes as follows:

1. u chooses a parametrized model architecture $\mathcal{F}_u \subset \mathcal{F}$ that has enough complexity to model entirely $p_u(y|\mathbf{x})$ (or at least reasonably well if $p_u(y|\mathbf{x})$ is not in any computable architecture $\mathcal{F}_u$). This choice is often influenced by the computational power available to u. Let $\mathcal{W}_{\mathcal{F}_u}$ be the set of all valid parameters $\mathbf{w}_u$ for the model architecture $\mathcal{F}_u$.
2. u selects a loss functional $\ell : \mathcal{F}_u \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$ taking as input a function $f_u \in \mathcal{F}_u$ parametrized by $\mathbf{w}_u$ and a data sample $(\mathbf{x}, y)$ (e.g., the cross entropy loss). ℓ would also include any regularization terms of f_u.
3. u defines the *true risk* of the model f_u as

$$\mathcal{L}(f_u, p_u) := \mathbb{E}_{(\mathbf{X},Y)\sim p_u}[\ell(f_u, \mathbf{X}, Y)], \tag{8}$$

   and the independent learning problem reads:

$$\text{Find } \mathbf{w}_u^* := \quad \arg\min_{\mathbf{w}_u \in \mathcal{W}_{\mathcal{F}_u}} \mathcal{L}(f_u, p_u). \tag{9}$$

4. Since p_u is unknown, $\mathbf{w}_u^*$ cannot be found directly. Hence, the true risk is replaced by the *empirical risk*

$$L(f_u, D_u) := \frac{1}{n_u} \sum_{i=1}^{n_u} \ell(f_u, \mathbf{x}_i, y_i), \tag{10}$$

   where D_u is a realization of the dataset $\mathcal{D}_u := \{(\mathbf{X}_i, Y_i) \overset{iid}{\sim} p_u\}_{i=1}^{n_u}$. With the iid assumption, it can easily be shown that $\mathbb{E}_{\mathcal{D}_u}[L(f_u, \mathcal{D}_u)] = \mathcal{L}(f_u, p_u)$, i.e., that L is an unbiased estimator of $\mathcal{L}$. The proxy problem of (9) can be expressed as

$$\text{Find } \bar{\mathbf{w}}_u := \quad \arg\min_{\mathbf{w}_u \in \mathcal{W}_{\mathcal{F}_u}} L(f_u, D_u). \tag{11}$$

   With the law of large numbers, we have that $\lim_{n_u \to \infty} \bar{\mathbf{w}}_u = \mathbf{w}_u^*$. In practice, however, finite values of n_u that exceed the sample complexity of the problem (largely dependent on p_u) are sufficient.
5. As the proxy problem rarely has a closed form solution, u tries to solve (11) using a learning algorithm, for example stochastic gradient descent (SGD). Let $\hat{\mathbf{w}}_u$ denote the output of this algorithm.

Finally, for inference on a new data sample $\mathbf{x}$, u estimates y by computing $\hat{y} = \arg\max_y f_u(y|\mathbf{x}; \hat{\mathbf{w}}_u)$.

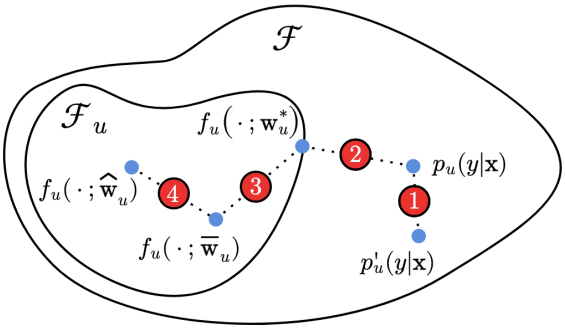


Figure 6: Different types of approximation errors.

Practical Limitations of Independent Learning With this formalization of IL, we can easily identify the four main sources of error that can arise in practical applications (see Fig. 6 for a conceptual representation of the different errors).

1. Even if the dataset $\mathcal{D}$ meets all the requirements (i.e., large enough and iid sampled on p_u), there is no guarantee that the data distribution at inference p'_u is the same as the data distribution of the training samples p_u. This is a problem that will never be fully solved as it would require to predict the future, but it can be partially addressed by retraining the model regularly on new data (which would hopefully make p_u more similar to p'_u), or by participating in a collaborative learning scheme where other users might have data that is currently more relevant (this would correspond to the assumption that p'_u is closer to $p_{\mathbf{X},Y,U}$ than to $p_u := p_{\mathbf{X},Y|U=u}$, with $p_{\mathbf{X},Y,U}$ the joint data distribution across users).
2. The optimal model $p_u(y|\mathbf{x})$ for the training distribution p_u is often too complex to be fully parametrized using a model architecture $\mathcal{F}_u$ compatible with the computational resources at hand. In that scenarios, a collaborative scheme like SL can relieve the edge user of some of the computation. However, even if the computational capacity is not the bottleneck, there is no guarantee that the chosen architecture contains the optimal model $p_u(y|\mathbf{x})$. In practice, these architectures are usually chosen using heuristics that are derived on similar but different tasks.
3. The third source of error concerns the empirical approach. There are two main reasons why $\overline{\mathbf{w}}_u$ might be different from $\mathbf{w}_u^*$:
   (a) The iid assumption during the sampling of the dataset $\mathcal{D}_u$ might not hold. In that case, the empirical risk is generally a biased estimate of the true risk and the problems (9) and (11) might not have the same solution, even when n_u grows to infinity. This is a hard problem that can only be addressed during the data acquisition.
   (b) The sample complexity might be larger than n_u, meaning that the empirical risk has too much variance compared to the true risk. In other word, the proxy problem (11) is too far from the true problem (9). Participating in a collaborative learning algorithm could potentially be a solution to this problem, as it would increase the number of samples on which the model is trained (or indirectly exposed).
4. Finally, the last source of error concerns the training algorithm itself. Indeed, even with all the previous assumptions, finding the solution to the proxy problem remains challenging. Most optimization algorithms rely on gradient descent, but since the empirical risk is usually highly non-convex w.r.t. the parameters $\mathbf{w}_u$, there is no guarantee that the algorithm does not get stuck in a local minima.

These sources of error constitute the main incentive for participating in a collaborative scheme. In the next subsection, we formalize the main collaborative frameworks and their objective.

Collaborative Learning With the same notation, consider the scenario when N users $\{u\}_{u=1}^N$ collaborate with the aim of achieving better performance than any IL algorithm. Here are the two main paradigms of such frameworks:

- *Global model approach*: When the goal is to learn a global model (e.g., FL), a central entity S (e.g., server) chooses the global architecture $\mathcal{F}_S \subset \mathcal{F}$ and the true risk of the global model $f_S \in \mathcal{F}_S$ is defined as

$$\mathcal{L}(f_S, p_{\mathbf{X},Y,U}) = \mathbb{E}_{(\mathbf{X},Y,U)\sim p_{\mathbf{X},Y,U}}[\ell(f_S, \mathbf{X}, Y)]. \tag{12}$$

  The marginal $p_U(u)$ represents the probability that a data sample is drawn from the distribution of user u. If the frequency of inference samples is proportional to the training datasets sizes, $p_U(u)$ should be replaced by $\frac{n_u}{\sum_{u'} n_{u'}}$ in the empirical risk. On the other hand, if the inference samples are equally likely to come from any user, $p_U(u)$ should be replaced by $\frac{1}{N}$. In the latter case, the empirical risk becomes

$$L(f_S, \{D_u\}_{u=1}^N) = \frac{1}{N}\sum_{u=1}^{N}\left(\frac{1}{n_u}\sum_{i=1}^{n_u}\ell(f_S, \mathbf{x}_i, y_i)\right). \tag{13}$$

- *Personalized model approach*: When a global model is not wanted/feasible, every user u choose a model architecture $\mathcal{F}_u \subset \mathcal{F}$ (dependently or independently, depending on the framework). For u, the true risk of its personalized model $f_u \in \mathcal{F}_u$ can generally be expressed as a tradeoff between local and global performance:

$$\mathcal{L}(f_u, p_{\mathbf{X},Y,U}) = \mathbb{E}_{(\mathbf{X},Y)\sim p_u}[\ell(f_u, \mathbf{X}, Y)] + \lambda\mathbb{E}_{(\mathbf{X},Y,U)\sim p_{\mathbf{X},Y,U}}[\ell(f_u, \mathbf{X}, Y)]. \tag{14}$$

  The parameter λ symbolizes the amount of collaboration, and $\lambda = 0$ is the same as IL. Since (14) can vary substantially depending on the application, we do not present the corresponding empirical risk.

It is also possible to mix these two paradigms, for example in split learning, where the model is separated into a global and personalized sub-models.

Practical Challenges of Collaborative Learning In the global model approach, the global model must have enough capacity to fit the distribution $p_{\mathbf{X},Y,U}$, whose complexity is usually higher than p_u. This can lead to large models that are too expensive to train locally, or to small models that significantly underfit the data. In addition to that, although it is more an engineering challenge than a theoretical limitation, the issue of communication can also hinder the performance of the training algorithm. However, the main challenge with collaborative learning comes from the global data distribution $p_{\mathbf{X},Y,U}$. Indeed, it is often the case that $p_u \neq p_{u'}$ for $u \neq u'$, or, in other words, that the data is not identically distributed across peers. This issue has been addressed in different works [21], but there is no "one size fits all" solution. To better understand the challenges, consider the point of view of a user u who wants to collaborate in a network with global distribution

$$p(\mathbf{x}, y) := p_{\mathbf{X},Y}(\mathbf{x}, y) = \sum_{u=1}^{N} p_{\mathbf{X},Y,U}(\mathbf{x}, y, u),$$

and assume that the data is not identically distributed ($p_u \neq p$). Using Bayes' theorem, we have

$$p_u(y|\mathbf{x})p_u(\mathbf{x}) \neq p(y|\mathbf{x})p(\mathbf{x}) \quad \text{and} \quad p_u(\mathbf{x}|y)p_u(y) \neq p(\mathbf{x}|y)p(y). \tag{15}$$

There are exactly 9 different scenarios that can arise from 15. These scenarios are listed in Table 2 and simple examples for each of them can be visualized in Fig. 7. Some of them are more challenging than others. For example, $p_u(\mathbf{x}) \neq p(\mathbf{x})$ and $p_u(y) \neq p(y)$ are not critical as they do not necessarily signify a different relation between $\mathbf{X}$ and Y. On the other hand, $p_u(\mathbf{x}|y) \neq p(\mathbf{x}|y)$ and especially $p_u(y|\mathbf{x}) \neq p(y|\mathbf{x})$ are symptomatic of the fact that the intrinsic relation between the input and output variables is different, which makes the problem harder to solve.

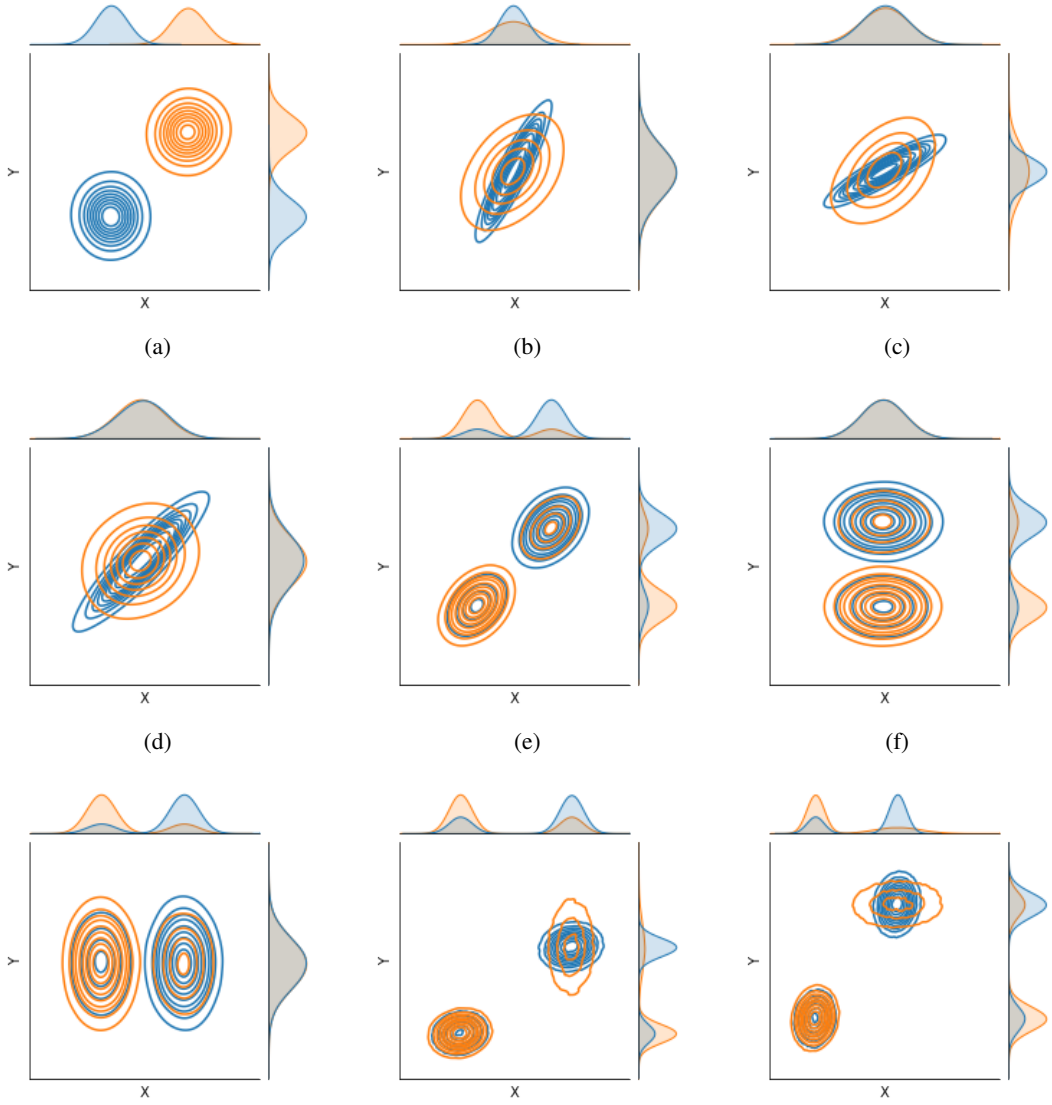


Figure 7: Visualization of the different types of discrepancies between two distributions p_u and p. In these examples, the input data X has one dimension and the output data Y is continuous. The joint distributions $p_u(x, y)$ and $p(x, y)$ are plotted inside the axes, the marginals $p_u(x)$ and $p(y)$ are plotted above the axes, and the marginals $p_u(y)$ and $p(y)$ are plotted on the right. See Table 2 for more details about which scenario corresponds to which subfigure.

Table 2: Different types of data distribution discrepancies. The symbol ⚠ indicates a configuration that is either not possible or breaks the assumptions $p_u \neq p$. In Fig. 7, examples of two distribution p_u and p that satisfy the 4 (in)equalities are presented for each of the 9 possible scenarios.

$p_u(\mathbf{x}) = p(\mathbf{x})$	$p_u(y\|\mathbf{x}) = p(y\|\mathbf{x})$	$p_u(y) = p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	⚠
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	⚠
		$p_u(y) \neq p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	⚠
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	⚠
	$p_u(y\|\mathbf{x}) \neq p(y\|\mathbf{x})$	$p_u(y) = p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	⚠
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	Figure 7d
		$p_u(y) \neq p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	Figure 7f
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	Figure 7c
$p_u(\mathbf{x}) \neq p(\mathbf{x})$	$p_u(y\|\mathbf{x}) = p(y\|\mathbf{x})$	$p_u(y) = p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	⚠
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	Figure 7g
		$p_u(y) \neq p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	Figure 7e
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	Figure 7i
	$p_u(y\|\mathbf{x}) \neq p(y\|\mathbf{x})$	$p_u(y) = p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	⚠
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	Figure 7b
		$p_u(y) \neq p(y)$	$p_u(\mathbf{x}\|y) = p(\mathbf{x}\|y)$	Figure 7h
			$p_u(\mathbf{x}\|y) \neq p(\mathbf{x}\|y)$	Figure 7a

A.3 Theoretical Background

Before presenting the proof of Theorem 1, we give some base knowledge in this section.

Discrete Random Variables Let $\mathbf{X}, \mathbf{Y}$ be two discrete random vectors (i.e., collection of random variables) taking values in the alphabets $\mathcal{X}$ and $\mathcal{Y}$, and with joint and marginal probability *mass* functions $p_{\mathbf{X},\mathbf{Y}}$, $p_{\mathbf{X}}$ and $p_{\mathbf{Y}}$, respectively. The *entropy* $\mathcal{H}$ of $\mathbf{X}$ (also referred to as the *Shannon entropy* or the *information entropy*) is defined as follows:

$$\mathcal{H}(\mathbf{X}) := -\sum_{\mathbf{x}\in\mathcal{X}} p_{\mathbf{X}}(\mathbf{x}) \log_a(p_{\mathbf{X}}(\mathbf{x})), \tag{16}$$

where a is the base of the logarithm and controls the *unit* of $\mathcal{H}$ (*bits* for $a = 2$, *nats* for $a = e$ (Euler's constant) and *bans* for $a = 10$). Intuitively, the entropy represents the uncertainty of the distribution $p_{\mathbf{X}}$, or, in other words, the expected amount of information that is gained by a single observation of $\mathbf{X}$. From its definition, it is non-negative and, with some additional work, it can be bounded as follows: $0 \leq \mathcal{H}(\mathbf{X}) \leq \log_a(|\mathcal{X}|)$. The *joint entropy* of $\mathbf{X}$ and $\mathbf{Y}$ is naturally defined as

$$\mathcal{H}(\mathbf{X},\mathbf{Y}) := -\sum_{\substack{\mathbf{x}\in\mathcal{X}\\ \mathbf{y}\in\mathcal{Y}}} p_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) \log_a(p_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})). \tag{17}$$

Additionally, the *conditional entropy* $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ (i.e., the average entropy of $\mathbf{X}$ when $\mathbf{Y}$ is observed) is defined as

$$\mathcal{H}(\mathbf{X}|\mathbf{Y}) := -\sum_{\substack{\mathbf{x}\in\mathcal{X}\\ \mathbf{y}\in\mathcal{Y}}} p_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) \log_a(p_{\mathbf{X}|\mathbf{Y}=\mathbf{y}}(\mathbf{x})). \tag{18}$$

A closely related quantity is the *mutual information* between $\mathbf{X}$ and $\mathbf{Y}$, defined as

$$\mathcal{I}(\mathbf{X},\mathbf{Y}) := \mathcal{H}(\mathbf{X}) - \mathcal{H}(\mathbf{X}|\mathbf{Y}) \equiv \mathcal{H}(\mathbf{Y}) - \mathcal{H}(\mathbf{Y}|\mathbf{X}). \tag{19}$$

The mutual information quantifies what is learned about one random variable when the other is observed, and is therefore also referred to as *information gain*. It is a measure of the dependence between two random vectors/variables.

Continuous Random Variables Although one could think that the generalization of these quantities to continuous random variables is simply done by replacing sums with integrals, one must remain cautious about their interpretation. To illustrate this, consider the continuous random vectors $\mathbf{X}$ and $\mathbf{Y}$ taking values in the sets $\mathcal{X}$ and $\mathcal{Y}$, and with joint and marginal probability *density* functions $p_{\mathbf{X},\mathbf{Y}}$, $p_{\mathbf{X}}$ and $p_{\mathbf{Y}}$, respectively (assuming they exist). We could be easily tempted to define a natural extension of the discrete entropy as follows[1]:

$$\mathcal{H}(\mathbf{X}) := -\int_{\mathcal{X}} p_{\mathbf{X}}(\mathbf{x}) \log_a(p_{\mathbf{X}}(\mathbf{x}))\, \mathrm{d}\mathbf{x}. \tag{20}$$

However, the intuitive properties of the discrete entropy do not transfer to this new quantity. To see this, consider the random variable X with uniform distribution on the support $[0, \frac{1}{2}]$. Using the definition in Eq. (20) with $a = 2$, we get a negative entropy of -1 bit, which is not intuitive. The quantity is therefore referred to as *differential entropy*, in contrast to the *Limiting Density of Discrete Points* (LDDP), which is considered to be the true continuous version of the entropy. Nevertheless, for reasons that are not presented here, the continuous version of the mutual information retains its significance if one replace the discrete entropy with the differential entropy in Eq. (19). That is, it represents the expected reduction of bits (if $a = 2$) necessary to encode one random variable when the other is observed. Hence, maximizing a lower bound on the mutual information between two random variables (as in Theorem 1) ensures that no useful information is lost during previous deterministic data manipulation (e.g., during the forward pass of a neural network).

A.4 Proof of Theorem 1

Recall that $q(\mathbf{s}, \mathbf{t}, i)$ is the joint distribution of $(\mathbf{\Phi}_s, \mathbf{\Phi}_t, I)$ such that $q(\mathbf{s}, \mathbf{t}|i = 1) = p_{s,t}(\mathbf{s}, \mathbf{t})$ and $q(\mathbf{s}, \mathbf{t}|i = 0) = p_s(\mathbf{s})p_t(\mathbf{t})$ and suppose that the prior $q(i)$ satisfy $q(i = 1) = \frac{1}{K+1}$ and $q(i = 0) = \frac{K}{K+1}$, i.e., for each sample from the distribution $p_{s,t}$, we draw K samples from the distribution $p_s p_t$. We have:

$$\begin{aligned}
\mathcal{I}(\mathbf{\Phi}_s, \mathbf{\Phi}_t) &= \mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_{s,t}} \left[-\log \frac{p_s(\mathbf{\Phi}_s)p_t(\mathbf{\Phi}_t)}{p_{s,t}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)} \right] && (21)\\
&= \mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_{s,t}} \left[-\log \left(K \frac{p_s(\mathbf{\Phi}_s)p_t(\mathbf{\Phi}_t)}{p_{s,t}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)} \right) \right] + \log(K) && (22)\\
&\geq \mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_{s,t}} \left[-\log \left(1 + K \frac{p_s(\mathbf{\Phi}_s)p_t(\mathbf{\Phi}_t)}{p_{s,t}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)} \right) \right] + \log(K) && (23)\\
&= \mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_{s,t}} \left[\log q(i = 1|\mathbf{\Phi}_s, \mathbf{\Phi}_t) \right] + \log(K) && (24)
\end{aligned}$$

where the last equality is obtained using the Bayes' rule on the posterior $q(i = 1|\mathbf{\Phi}_s, \mathbf{\Phi}_t)$:

$$\begin{aligned}
q(i = 1|\mathbf{\Phi}_s, \mathbf{\Phi}_t) &= \frac{q(\mathbf{\Phi}_s, \mathbf{\Phi}_t|i = 1)q(i = 1)}{q(\mathbf{\Phi}_s, \mathbf{\Phi}_t|i = 0)q(i = 0) + q(\mathbf{\Phi}_s, \mathbf{\Phi}_t|i = 1)q(i = 1)} && (25)\\
&= \frac{p_{s,t}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)}{K p_s(\mathbf{\Phi}_s)p_t(\mathbf{\Phi}_t) + p_{s,t}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)} && (26)\\
&= \left(1 + K \frac{p_s(\mathbf{\Phi}_s)p_t(\mathbf{\Phi}_t)}{p_{s,t}(\mathbf{\Phi}_s, \mathbf{\Phi}_t)} \right)^{-1}. && (27)
\end{aligned}$$

Hence, by optimizing $\mathbb{E}_{(\mathbf{\Phi}_s, \mathbf{\Phi}_t) \sim p_{s,t}} \left[\log q(i = 1|\mathbf{\Phi}_s, \mathbf{\Phi}_t) \right]$ with respect to the model parameters $\boldsymbol{\theta}_u$ of the student, we optimize a lower bound on the mutual information between $\mathbf{\Phi}_s, \mathbf{\Phi}_t$. By noting that

[1] As was Claude Shannon. Indeed, without any derivation, he assumed that this definition was the natural extension of the discrete entropy to continuous random vectors/variables.

$\log q(i=0|\mathbf{\Phi}_s, \mathbf{\Phi}_s) \leq 0$, we can further bound the expectation term in (24) as follows:

$$\begin{aligned}
\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim p_{s,t}}\left[\log q(i=1|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] &\geq \mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q|I=1}\left[\log q(i=1|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] \\
&\quad + K\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q|I=0}\left[\log q(i=0|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] && (28)\\
&= (K+1)\sum_i q(i)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q|I=i}\left[\log q(i|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] && (29)\\
&= (K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t,I)\sim q}\left[\log q(I|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] && (30)\\
&= (K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q}\left[\mathbb{E}_{I\sim q|\mathbf{\Phi}_s,\mathbf{\Phi}_t}\left[\log q(I|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right]\right] && (31)
\end{aligned}$$

However, similar to Tian et al. [44], the Bernoulli distribution $q(i|\mathbf{\Phi}_s, \mathbf{\Phi}_t)$ is unknown and must therefore be approximated by training a discriminator $h : \{0,1\} \times \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to [0,1]$. Using Gibbs' inequality, we obtain

$$-\mathbb{E}_{I\sim q|\mathbf{\Phi}_s,\mathbf{\Phi}_t}\left[\log q(I|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] \leq -\mathbb{E}_{I\sim q|\mathbf{\Phi}_s,\mathbf{\Phi}_t}\left[\log h(I,\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right], \tag{32}$$

where the right-hand term is the expected negative log-likelihood loss of the discriminator for a particular set $(\mathbf{\Phi}_s, \mathbf{\Phi}_t)$. Hence, Eq. (31) is proportional to minus the expected loss of the discriminator. Let $\hat{h}(\mathbf{s},\mathbf{t}) \in [0,1]$ denote the Bernoulli parameter of h given the data $(\mathbf{s},\mathbf{t})$ (i.e., $h(i,\mathbf{s},\mathbf{t}) = \hat{h}(\mathbf{s},\mathbf{t})^i(1-\hat{h}(\mathbf{s},\mathbf{t}))^{1-i}$), we define our learning objective for the discriminator as follows:

$$\begin{aligned}
-\mathcal{L}_{disc}(h,\phi_u) &:= (K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q}\left[\mathbb{E}_{I\sim q|\mathbf{\Phi}_s,\mathbf{\Phi}_t}\left[\log h(I,\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right]\right] && (33)\\
&= (K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t,I)\sim q}\left[\log h(I|\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] && (34)\\
&= (K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t,I)\sim q}\left[\log\left(\hat{h}(\mathbf{\Phi}_s,\mathbf{\Phi}_t)^I(1-\hat{h}(\mathbf{\Phi}_s,\mathbf{\Phi}_t))^{(1-I)}\right)\right] && (35)\\
&= +(K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q|I=1}\left[\log\hat{h}(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right]q(i=1) \\
&\quad (K+1)\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim q|I=0}\left[\log\left(1-\hat{h}(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right)\right]q(i=0) && (36)\\
&= \mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim p_{s,t}}\left[\log\hat{h}(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\right] + K\mathbb{E}_{(\mathbf{\Phi}_s,\mathbf{\Phi}_t)\sim p_s p_t}\left[\log(1-\hat{h}(\mathbf{\Phi}_s,\mathbf{\Phi}_t))\right], && (37)
\end{aligned}$$

which concludes the derivation.

A.5 Algorithms

Algorithm 1: GLOBALUPDATE

Input: Server S, N users with local dataset realizations $\{D_u\}_{u=1}^N$.

S initializes randomly $\{\bar{\mathbf{t}}^c\}_{c=1}^C$ and random observations $\{\{\mathbf{t}_m^c\}_{c=1}^C\}_{m=1}^{N\cdot M_\uparrow}$
Each client u initializes $\mathbf{w}_u^0 := \{\boldsymbol{\theta}_u^0, \mathbf{W}_u^0, \mathbf{b}_u^0\}$
$r \leftarrow 0$
while *training:*
  $r \leftarrow r+1$
  for *each client* u*:*
    u downloads $\{\bar{\mathbf{t}}^c\}_{c=1}^C$ and $M_\downarrow$ random observations $\{\{\mathbf{t}_m^c\}_{c=1}^C\}_{m=1}^{M_\downarrow}$
    $\mathbf{w}_u^r \leftarrow$ LOCALUPDATE$(D_u, \mathbf{w}_u^{r-1}, \{\bar{\mathbf{t}}^c\}_{c=1}^C, \{\{\mathbf{t}_m^c\}_{c=1}^C\}_{m=1}^{M_\downarrow})$
    u computes and uploads its local averaged representations $\{\bar{\mathbf{t}}_u^c\}_{c=1}^C$
    u computes (using n_{avg}) and uploads $M_\uparrow$ observations $\{\{\mathbf{t}_{u,m}^c\}_{c=1}^C\}_{m=1}^{M_\downarrow}$
  S aggregates $\{\{\bar{\mathbf{t}}_u^c\}_{c=1}^C\}_{u=1}^N$ to obtain $\{\bar{\mathbf{t}}^c\}_{c=1}^C$
  S stores (and shuffles) $\{\{\mathbf{t}_{u,m}^c\}_{c=1}^C\}_{m=1}^{M_\downarrow}$ in their corresponding class buffers
return $\{\mathbf{w}_u^r\}_{u=1}^N$

Algorithm 2: LOCALUPDATE

Input: Local dataset realization D_u, model parameters $\mathbf{w}_u = \{\boldsymbol{\theta}_u, \mathbf{W}_u, \mathbf{b}_u\}$, global features $\{\bar{\mathbf{t}}^c\}_{c=1}^{C}$ and observations $\{\{\mathbf{t}_m^c\}_{c=1}^{C}\}_{m=1}^{M_\downarrow}$, number of local training rounds E, averaging parameter n_avg.

Buffer initialization (per class): $\{\hat{\boldsymbol{\Phi}}_u^c \leftarrow \mathbf{0}\}_{c=1}^{C}$
for $e \in [1, ..., E]$*:*
  for *mini-batch* $\mathcal{B} \subset D_u$*:*
    $L_{KD}, L_{CE}, L_{disc} \leftarrow 0$
    for $(\mathbf{x}_i, y_i) \in \mathcal{B}$*:*
      $\mathbf{s}_i \leftarrow \phi_u(\mathbf{x}_i)$
      $L_{CE} \leftarrow L_{CE} + \frac{1}{|\mathcal{B}|}\ell_{CE}(\tau_u(\mathbf{s}_i), y_i)$
      $L_{KD} \leftarrow L_{KD} + \frac{1}{|\mathcal{B}|}\ell_{KD}(\mathbf{s}_i, \bar{\mathbf{t}}^{y_i})$
      Sample $m \sim$ UNIFORM$(1, ..., M_\downarrow)$
      $L_{disc} \leftarrow L_{disc} + \frac{1}{|\mathcal{B}|}\left(\ell_{disc}(1, \mathbf{s}_i, \mathbf{t}_m^{y_i}) + \sum_{c \neq y_i} \ell_{disc}(0, \mathbf{s}_i, \mathbf{t}_m^c)\right)$
    $L \leftarrow L_{CE} + \lambda_{KD} L_{KD} + \lambda_{disc} L_{disc}$
    $\mathbf{w}_u \leftarrow \mathbf{w}_u - \eta \nabla_{\mathbf{w}_u} L$
return $\mathbf{w}_u$