Confederated Learning: Going Beyond Centralization

Zitai Wang
SKLOIS, IIE, CAS
SCS, UCAS
wangzitai@iie.ac.cn

Qianqian Xu*
IIP, ICT, CAS
xuqianqian@ict.ac.cn

Ke Ma
SCST, UCAS
make@ucas.ac.cn

Xiaochun Cao
SCST, Shenzhen Campus, SYSU
SKLOIS, IIE, CAS
caoxiaochun@mail.sysu.edu.cn

Qingming Huang*
SCST, UCAS
IIP, ICT, CAS
BDKM, CAS
Peng Cheng Laboratory
qmhuang@ucas.ac.cn

ABSTRACT

Traditional machine learning implicitly assumes that a single entity (*e.g.*, a person or an organization) could complete all the jobs of the whole learning process: data collection, algorithm design, parameter selection, and model evaluation. However, many practical scenarios require cooperation among entities, and existing paradigms fail to meet cost, privacy, or security requirements and so on. In this paper, we consider a generalized paradigm: different roles are granted multiple permissions to complete their corresponding jobs, called **Confederated Learning**. Systematic analysis shows that confederated learning generalizes traditional machine learning and the existing distributed paradigms like federation learning. Then, we study an application scenario of confederated learning which could inspire future research in the context of cooperation between different entities. Three methods are proposed as the first trial for the cooperated learning under restricted conditions. Empirical results on three datasets validate the effectiveness of the proposed methods.

CCS CONCEPTS

• **Computing methodologies** → **Cooperation and coordination**.

KEYWORDS

Confederated Learning, Cooperated Learning, Learning Paradigm

ACM Reference Format:
Zitai Wang, Qianqian Xu, Ke Ma, Xiaochun Cao, and Qingming Huang. 2022. Confederated Learning: Going Beyond Centralization. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548157

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548157

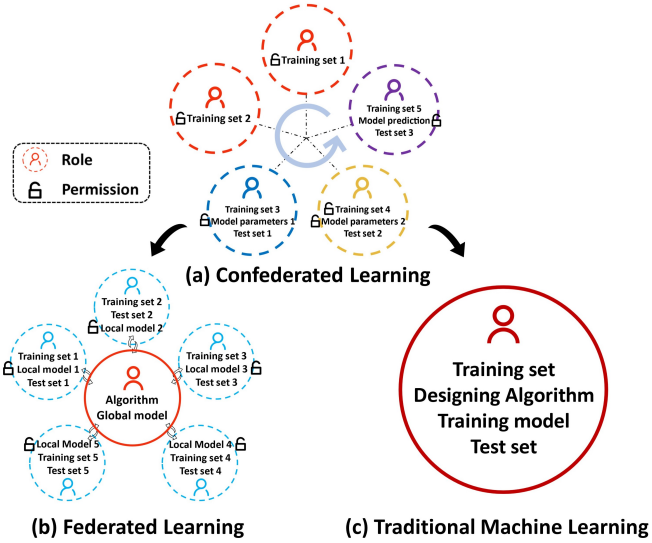


Figure 1: Illustration of the role and permission issues in machine learning. (a) In confederated learning, different roles are granted multiple permissions to complete their corresponding jobs. (b) There exist two roles in federated learning: center and edge. All edges have the same permissions. (c) Traditional machine assumes that the single role would have all permissions.

1 INTRODUCTION

Traditional machine learning implicitly assumes that one entity would control the whole learning process. As shown in Figure 1(a), a single entity first receives a training set, then customizes the algorithm and tunes the parameters according to the given dataset, and finally makes predictions for unseen data points based on the learned model. From this perspective, the prior arts belong to a centralized learning paradigm, and the permission setting is relatively simple.

When participating in large-scale socialized production, the machine learning process starts to involve multiple entities. Cooperation among entities becomes crucial since each entity has its own cost, privacy, or security concerns. Such requirements give

rise to cooperative paradigms like federated learning [23], which aims to train a centralized model based on the data distributed over numerous edge entities. As shown in Figure 1(b), a typical federated learning method consists of two procedures:

- Edge entities receive the global model and train the local models according to his/her data samples.
- Center entity aggregates all the local models to update a new global model.

With federated learning, participants could obtain a high-quality model without sharing private data. However, the federated learning paradigm has its own problems. On one hand, there are only two types of entities: center and edges. The edge entities usually trust the center unconditionally. Such a rigid role setting fails to cover many cooperation scenarios. On the other hand, the learning process depends on the credibility of the center entity. Establishing a trustworthy center entity requires high costs. As a consequence, federated learning is costly and unreliable in many scenarios.

In this paper, we provide a systematic analysis of the cooperation paradigm in machine learning from the perspective of roles and permissions. Firstly, we define the concept of role and permission in cooperation: each role grants specific permissions according to his/her own concerns. For example, a role is responsible for providing domain knowledge. However, he/she cannot release the dataset or model parameters for the other roles with security concerns. Based on the concept of role and permission, we propose a cooperation framework named **Confederated Learning**, which is consistent with the sociological concept of confederation and generalizes the existing learning paradigms. For instance, traditional machine learning is a special scenario where the single entity is granted all the possible permissions. As another example, there exist two types of roles in federated learning: one is responsible for providing knowledge in the form of model parameters, and the others leverage their own data separately.

We further establish a confederated learning task. There exist two roles in such a kind of cooperation: the first one aims to train a model h on a private dataset $\mathcal{S}_1$ but fails to achieve satisfactory performance. Meanwhile, the second role has trained a high-quality model g^* on its own dataset $\mathcal{S}_2$. In traditional machine learning, the first role could transfer the knowledge contained in $\mathcal{S}_2$ or g^* to boost the learning process. However, this is infeasible since the second role only has the permission to release the model in the form of API that only returns inference results. *How should these two roles cooperate under such restricted conditions*? In fact, such a scenario will become more common due to the fact that both the datasets and pre-trained models are becoming too large and expensive to release. Generative Pre-trained Transformer 3 (GPT-3) is such an example with 175 billion parameters. The large volume of model and training data means that the learning and inference process requires very expensive computation facilities [6].

Furthermore, we propose three methods as the first trial for the above cooperation: (1) prediction ensemble that considers the convex combination of the label in $\mathcal{S}_1$ and model prediction $g^*(\cdot)$ as the final training target; (2) sample re-weighting that emphasizes some data points based on the model prediction $g^*(\cdot)$; (3) consistent regularization that explicitly encourages the consistency between the predictions of the model g^* and h. Since all the proposed methods are model-free and independent of data distribution, they are suitable for the above scenario. Empirical results on three datasets show that the proposed methods are effective for cooperated learning under restricted conditions. Our contributions are summarized as follows:

- We provide a systematic analysis of the cooperation issue in machine learning. We show that the existing learning paradigms fail to assign different entities with the appropriate roles and permissions. **Confederated Learning**, which is equipped with the complete role and permission setting, is proposed to generalize the existing cooperation framework.
- A novel cooperation problem is established. In this scenario, the cooperator shares its knowledge in a restricted form. Three methods are proposed as the first trial for solving this confederated learning problem.
- Empirical results on three datasets show that the proposed methods help boost the learning process, which might inspire future research in the context of cooperated learning.

2 RELATED WORK

Traditional distributed machine learning allocates the whole learning workload across multiple machines for an increase of computation power and the total amount of I/O bandwidth [32], which is spawned from the rapid development of deep learning and the corresponding sophisticated applications such as self-driving [4], speech recognition [2]. Generally speaking, traditional distributed machine learning methods fall into two camps:

Data-parallel approaches distribute the whole datasets across multiple worker nodes (*i.e.,* entities in our context). All the worker nodes apply the same algorithm to their local datasets and synchronize the model through centralization or replication. Common synchronization strategies include:

- Bulk Synchronous Parallel (BSP) that synchronizes only between computation and synchronization phase of all the workers [35];
- Stale Synchronous Parallel (SSP) that allows faster workers to perform a certain number more iterations and updates all the workers when this number is exceeded [17];
- Approximate Synchronous Parallel (ASP) where each worker aggregates the insignificant updates to the same parameters locally until the aggregated updates are significant enough [18];
- Barrierless Asynchronous Parallel [13] / Total Asynchronous Parallel [18] (BAP/TAP) further allows workers to communicate in parallel without waiting for each other.

Generally, a stricter synchronization strategy provides a faster model convergence, while a looser one guarantees faster updates. With the help of existing distributed learning frameworks such as Google File System (GFS) [10], MapReduce [7], data-parallel approaches are adaptive to every machine learning method with an independent and identical distribution (*i.i.d.*) assumption.

In *model-parallel* approaches, each worker node updates different parts of the model on the entire dataset, and the final model is

the aggregation of all the model parts [32]. Since it is generally difficult to split up model parameters, model-parallel approaches are not inherently feasible for most machine learning algorithms. One common option is to train multiple instances of the local model and then aggregate the model predictions via ensemble strategies [37] like bagging, boosting, bucketing, random forests [5], stacking, and so on.

Topology is another consideration in designing distributed machine learning systems. According to the general taxonomy of distributed communication networks [3], there are several popular topologies for distributed machine learning:

- In tree-like topologies, each node only communicates with its parent and children. For example, AllReduce [1] calculates the global gradient by accumulating the local gradients recursively from bottom to top.
- Ring topology only requires synchronizing the messages from neighbors. Such a simple topology is suitable for scenarios where communication efficiency makes sense (*e.g.,* local environment with multiple GPUs [21]).
- Parameter Server (PS) paradigm [34] distributes all the model parameters across multiple parameter servers. All the other nodes only communicate with the global shared memory of parameter servers so that it becomes easy to inspect the model.
- Peer-to-Peer topology is a fully-distributed paradigm where no centralized node exists and each node communicates directly with each other. An inherent advantage is its higher scalability and better fault tolerance than centralized topologies [9].

Traditional distributed machine learning methods focus on accelerating the computing process or improving fault tolerance. In recent years, there is a rising interest in distributed machine learning in a privacy-sensitive context [36]. For example, Federated Learning [23] adopts a tree-like topology where the center worker updates the global model based on the aggregated local gradients. Similarly, Split Learning [31] adopts a tree-like topology where local workers share smashed data rather than raw data to preserve privacy. Although Shokri and Shmatikov [30] have shown privacy preservation is believed to be available by applying differential privacy, Hitaj et al. [16] show that record-level differential privacy in federated learning is generally ineffective. As another example, Gossip Learning [15] fully embraces this principle by adopting a peer-to-peer topology. Specifically, Gossip Learning performs independent random walks through the peer-to-peer networks and merges the current model with a few previous visitors.

3 RETHINKING COOPERATION IN MACHINE LEARNING

In this section, we make a systematic analysis of the cooperation issue in machine learning. On one hand, we discuss the limitation of existing paradigms in the context of cooperated learning. On the other hand, we formulate confederated learning and show how it generalizes existing paradigms.

3.1 Preliminary

Traditional Machine Learning. Traditional machine learning [26] could be formulated as a quaternary set $\{e, \mathcal{S}, \mathcal{H}, \mathcal{A}, \mathcal{M}\}$, where e is the single entity responsible for the whole learning process; $\mathcal{S} = \{\mathcal{S}^{\text{tr}}, \mathcal{S}^{\text{te}}\}$ denotes the training and test set; $\mathcal{H}$ represents a set of models; $\mathcal{A}$ is a specific learning algorithm set; and $\mathcal{M}$ denotes a metric set defined on the data distribution $\mathcal{S}^{\text{te}}$. Generally speaking, traditional machine learning adopts the following pipeline: (1) The entity e receives the training set $\mathcal{S}^{\text{tr}}$. (2) Then, the entity could design an algorithm $\mathcal{A}$ that returns a model from the hypothesis set $\mathcal{H}$. (3) Finally, the model could be evaluated the model based on the metric set $\mathcal{M}$. In other words, we can formulate traditional machine learning as follows:

$$\min_{h} \mathcal{M}(h; \mathcal{S}^{te}), \tag{1}$$

where $h := \mathcal{A}(\mathcal{S}^{tr}, \mathcal{H})$ is the output model of the algorithm.

Implicitly, traditional machine learning assumes that *we* are the only entity responsible for the learning process. As a result, this centralized paradigm is naturally not suitable for cooperation scenarios.

Federated Learning. Federated learning can be denoted as $\{\mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{A}, \mathcal{M}\}$, where the entity set $\mathcal{E} = \{e_i^{\text{eg}}\}_{i=1}^{N_{\text{eg}}} \cup \{e^{\text{ct}}\}$ contains the center entity and the edge entities; $\mathcal{S} = \{\mathcal{S}_i^{\text{tr}}, \mathcal{S}_i^{\text{te}}\}_{i=1}^{N_{\text{eg}}}$ denotes the set of local data distributed on each edge entity; the algorithm $\mathcal{A} = \{\mathcal{A}^l, \mathcal{A}^g\}$ contains a local update algorithm $\mathcal{A}^l$ and a global aggregation algorithm $\mathcal{A}^g$; and the metric set $\mathcal{M}$ is defined on the local test sets $\{\mathcal{S}_i^{\text{te}}\}_{i=1}^{N_{\text{eg}}}$. Specifically, a typical federated learning iteration consists of four steps: (1) The center entity e^{ct} determines the algorithm $\mathcal{A}$ and initializes the global model from the hypothesis $\mathcal{H}$. (2) The center entity transmits the current model to the edge entities. (3) Each edge entity e_i^{eg} updates the received model based on the local data $\mathcal{S}_i^{\text{tr}}$ via the local update algorithm $\mathcal{A}^l$. (4) The center entity collects the local models and generates the new global model via the global aggregation algorithm $\mathcal{A}^g$. After several iterations, each edge entity e_i^{eg} could evaluate the global model based on their test sets $\mathcal{S}_i^{\text{te}}$.

Although involving the context of cooperated learning, the limitation of federated learning is obvious. On one hand, the edge entity should trust the center entity unconditionally, which fails to cover many practical scenarios. On the other hand, the learning process relies on the center entity, while establishing a trustworthy entity is generally costly. Therefore, it is necessary to explore a more generalized paradigm to describe practical cooperation scenarios.

3.2 Confederated Learning

Definition. Let $\mathcal{R} = \{r_i\}_{i=1}^{N_r}$ and $\mathcal{P} = \{p_n\}_{n=1}^{N_p}$ be the set of role and permission in cooperation. Meanwhile, let $\mathcal{S} = \{\mathcal{S}_j\}_{j=1}^{N_s}, \mathcal{H} = \{\mathcal{H}_k\}_{k=1}^{N_h}, \mathcal{A} = \{\mathcal{A}_l\}_{l=1}^{N_a}$, and $\mathcal{M} = \{\mathcal{M}_m\}_{m=1}^{N_m}$ denote the set of dataset, hypothesis, algorithm, and metric, respectively; and $h_{j,k,l}$ denote the model returned by the algorithm $\mathcal{A}_l$ from the hypothesis set $\mathcal{H}_k$ based on the dataset $\mathcal{S}_j$. Then, let $p(r_i, \mathcal{S}_j) \subseteq \mathcal{P}$ denote role

i's permissions on the dataset $\mathcal{S}_j$, and

$$P(r_i) = \{p(r_i, \mathcal{S}_j), p(r_i, \mathcal{H}_k), p(r_i, \mathcal{A}_l), p(r_i, \mathcal{M}_m), p(r_i, h_{j,k,l})\} \tag{2}$$

contains all the permissions of the role r_i. For example, if a role r_1 is responsible for building a dataset $\mathcal{S}_1$, the corresponding permission set $p(r_1, \mathcal{S}_1)$ could be denoted as {'create', 'read', 'update', 'delete'}. Based on the concept of role and permission, we can formulate confederated learning as a set

$$\{\mathcal{R}, \mathcal{P}, \mathcal{P}_{\mathcal{R}}, \mathcal{S}, \mathcal{H}, \mathcal{A}, \mathcal{M}\}, \tag{3}$$

where $\mathcal{P}_{\mathcal{R}} = \{P(r_i)\}_{i=1}^{N_r} \subset \mathcal{P} \times \mathcal{R}$ is the set of all roles' permissions.

The flexible role setting helps confederated learning cover more cooperation scenarios, as long as the permissions are well-defined. For example, a cloud platform (*i.e.*, role r_1) provides its computing power with a non-customizable algorithm $\mathcal{A}_1$ and a predetermined hypothesis set $\mathcal{H}_1$. And we (*i.e.*, role r_2) hope to collect training samples $\mathcal{S}_2$ according to the current performance on the metrics $\mathcal{M}_2$. Then, the problem could be described in a concise manner:

$$\{\{r_1, r_2\}, \mathcal{P}, \mathcal{P}_{\mathcal{R}}, \mathcal{S}_2, \mathcal{A}_1, \mathcal{H}_1, \mathcal{M}_2\}, \tag{4}$$

where $\mathcal{P}_{\mathcal{R}} = \{P(r_1), P(r_2)\}$,

$$P(r_1) = \{p(r_1, \mathcal{H}_1), p(r_1, \mathcal{A}_1)\}; \tag{5}$$

and

$$P(r_2) = \{p(r_1, h_{2,1,1}), p(r_2, \mathcal{S}_2), p(r_2, \mathcal{M}_2)\}. \tag{6}$$

Furthermore, confederated learning also brings a new point of view to understanding existing paradigms. We next consider federated learning and transfer learning as the representative of the existing paradigms.

Cooperated Federated learning. Let r^{eg} denote the edge role and $\{r_i^{\text{eg}}\}_{i=1}^{N_{\text{eg}}}$ be the corresponding entities. Following the paradigm of confederated learning, we could formulate federated learning as

$$\left\{\{r^{\text{eg}}, r^{\text{ct}}\}, \mathcal{P}, \mathcal{P}_{\mathcal{R}}, \{\mathcal{S}_i\}_{i=1}^{N_{\text{eg}}}, \mathcal{H}, \{\mathcal{A}^g, \mathcal{A}^l\}, \mathcal{M}\right\}, \tag{7}$$

where $\mathcal{P}_{\mathcal{R}} = \{P(r_i^{\text{eg}}), P(r^{\text{ct}})\}$,

$$P(r_i^{\text{eg}}) = \{p(r^{\text{eg}}, h_{\emptyset,\mathcal{H},\mathcal{A}^g}), p(r^{\text{eg}}, \mathcal{S}), p(r^{\text{eg}}, \mathcal{A}^l), p(r^{\text{eg}}, \mathcal{M})\}, \tag{8}$$

$$P(r^{\text{ct}}) = \{p(r^{\text{ct}}, h_{\mathcal{S}^{\text{eg}},\mathcal{H},\mathcal{A}^l}), p(r^{\text{ct}}, \mathcal{H}), p(r^{\text{ct}}, \mathcal{A}^g), p(r^{\text{ct}}, \mathcal{A}^l)\}, \tag{9}$$

$$p(r^{\text{eg}}, \mathcal{S}) = \{p(r_i^{\text{eg}}, \mathcal{S}_i)\}_{i=1}^{N_{\text{eg}}}, \tag{10}$$

$$p(r^{\text{ct}}, h_{\mathcal{S}^{\text{eg}},\mathcal{H},\mathcal{A}^l}) = \{p(r^{\text{ct}}, h_{\mathcal{S}_i^{\text{eg}},\mathcal{H},\mathcal{A}^l})\}_{i=1}^{N_{\text{eg}}}. \tag{11}$$

Cooperated Transfer learning. In traditional paradigm, transfer learning assumes that the single entity has access to the data points drawn from a source distributions $\mathcal{D}^S$ and a target dataset $\mathcal{D}^T$. A common strategy is to assign weights to the source-domain instances [8, 19, 25, 33]:

$$\begin{aligned} &\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}^T}[\ell(\boldsymbol{x}, y; h)] \\ =&\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}^S}\left[\frac{P^T(\boldsymbol{x}, y)}{P^S(\boldsymbol{x}, y)}\ell(\boldsymbol{x}, y; h)\right], \end{aligned} \tag{12}$$

where $P^S(\boldsymbol{x}, y)$ and $P^T(\boldsymbol{x}, y)$ represent the joint distribution of samples drawn from the source distribution $\mathcal{D}^S$ and target distribution $\mathcal{D}^T$, respectively.

In the context of cooperated learning, there exist a source role r^S and a target role r^T. The source role r^S is responsible for providing the source data $\mathcal{S}^S$ drawn from the source distribution $\mathcal{D}^S$; and the target role aims to improve the performance on the target set $\mathcal{S}^T$ with the relation between the source and target domain. Then, we could formulate transfer learning as

$$\left\{\{r^S, r^T\}, \mathcal{P}, \mathcal{P}_{\mathcal{R}}, \{\mathcal{S}^S, \mathcal{S}^T\}, \mathcal{H}^T, \mathcal{A}^T, \mathcal{M}^T\right\}, \tag{13}$$

where $\mathcal{P}_{\mathcal{R}} = \{P(r^S), P(r^T)\}, P(r^S) = \{p(r^S, \mathcal{S}^S)\}$, and

$$P(r^T) = \{p(r^T, \mathcal{S}^S), p(r^T, \mathcal{S}^T), p(r^T, \mathcal{H}^T), p(r^T, \mathcal{A}^T), p(r^T, \mathcal{M}^T)\}. \tag{14}$$

4 METHODOLOGY

In this section, we study another cooperation scenario, as an application of confederated learning. Besides, three methods are proposed as the first trial for such a restricted cooperation scenario.

4.1 Problem Definition

For privacy or security concerns, we (*i.e.*, r^T) only have access to the predictions of a model g^* pre-trained on samples drawn from some unknown distributions $\mathcal{D}^U = \mathcal{X}^U \times \mathcal{Y}$, where $\mathcal{X}^U$ is a private feature space, and $\mathcal{Y}$ is a public label space. In other words, the model providers only released the label of each class, and we have no more knowledge about g^*. Our task is to boost the training process on the target distribution $\mathcal{D}^T = \mathcal{X}^T \times \mathcal{Y}^T$ by leveraging the knowledge provided by the predictions of g^*, where $\mathcal{X}^T$ is the target feature space, and $\mathcal{Y}^T \subseteq \mathcal{Y}$ is the target label space.

Let r^U denote the role providing models. Then, in the context of confederated learning, this problem could be formulated as:

$$\left\{\{r^U, r^T\}, \mathcal{P}, \mathcal{P}_{\mathcal{R}}, \{\mathcal{S}^U, \mathcal{S}^T\}, \{\mathcal{H}^U, \mathcal{H}^T\}, \{\mathcal{A}^U, \mathcal{A}^T\}, \mathcal{M}^T\right\}, \tag{15}$$

where $\mathcal{P}_{\mathcal{R}} = \{P(r^U), P(r^T)\}$;

$$P(r^U) = \{p(r^U, \mathcal{S}^U), p(r^U, \mathcal{H}^U), p(r^U, \mathcal{A}^U)\}, \tag{16}$$

and

$$P(r^T) = \{p(r^T, h_{\mathcal{S}^U,\mathcal{H}^U,\mathcal{A}^U}), p(r^T, \mathcal{S}^T), p(r^T, \mathcal{H}^T), p(r^T, \mathcal{A}^T), p(r^T, \mathcal{M}^T)\}. \tag{17}$$

Note that $p(r^T, h_{\mathcal{S}^U,\mathcal{H}^U,\mathcal{A}^U}) = \{\text{'predictions'}\}$ means that we have access to the predictions of g^*. Meanwhile, we assume that the target training set suffers from a distribution mismatch. That is, we have $\mathcal{S}^T = \{(\boldsymbol{x}_i, \overline{y}_i)\}_{i=1}^m$, where the observed label $\overline{y}_i$ is not necessarily the underlying true label y_i.

4.2 The Proposed Methods

Since g^* is generally well-trained, it is intuitive to require the consistency between g^* and the target model. Based on this intuition, we next propose three methods as the first trial for the cooperated learning under restricted conditions:

Sample Reweighting (SR). It is well-known that different data points have different impacts on the final performance, which usually comes from the distribution mismatch between the training set and evaluation set [22, 28]. Here we aim to emphasize the data points that are more responsible for the predictions, which induce the following problem:

$$\min_h \sum_{i=1}^{m} w_i \ell_i(h) + \Omega_h, \tag{18}$$

where $\ell_i(h)$ denotes the loss of h on the data point $\boldsymbol{x}_i$, w_i is the training weight assigned to $\boldsymbol{x}_i$, and Ω_h represents a regularization term. Since the distribution $\mathcal{D}^U$ is inaccessible, we define the training weight as a function of model prediction and the observed label:

$$w_i = w(g_i^*, \overline{y}_i), \tag{19}$$

where g_i^* represents the model prediction $g^*(\boldsymbol{x}_i)$, and the weighting w should adapt to different scenarios. Since the existence of distribution mismatch would hurt the model performance, the weighting function w is designed to eliminate its impact on the training process. Meanwhile, it is necessary to require $w \geqslant 0$, since minimizing a negative training loss could lead to an unstable learning process. Hence, we have

$$w(g_i^*, \overline{y}_i) = \frac{\exp(-\alpha \cdot d(g_i^*, \overline{y}_i))}{\sum_{i=1}^{m} \exp(-\alpha \cdot d(g_i^*, \overline{y}_i))}, \tag{20}$$

where $\alpha \geqslant 0$ is the temperature parameter of each class; $d : \mathbb{R} \times \mathcal{Y}^T \to \mathbb{R}$ measures the distance between the observed label $\overline{y}_i$ and the model prediction g_i^*.

Intuitively, if the observed label is consistent with g_i^*, we will assign a higher weight to the associated data point $\boldsymbol{x}_i$. Otherwise, a lower weight would help eliminate the effect of a data point that is likely to be noisy.

Prediction Ensemble (PE). A straight-forward way to incorporate model predictions into the target training process is to use an ensemble of labels and model predictions as the training targets:

$$\hat{y}_i \leftarrow \text{Ensemble}(\overline{y}_i, g_i^*, h_i). \tag{21}$$

This induces the following optimization problem:

$$\min_h \sum_{i=1}^{m} \ell(h(\boldsymbol{x}_i), \hat{y}_i) + \Omega_h. \tag{22}$$

Here we consider the convex combination of the observed label and g^*:

$$\hat{y}_i \leftarrow \beta \overline{y}_i + (1-\beta) g_i^*, \tag{23}$$

where $\beta \in (0, 1)$ is the parameter determining the transfer degree. Intuitively, if we set β close to 0, the ensemble targets will rely more on g^*. Otherwise, the target model will be more cautious about g^* due to the domain difference between $\mathcal{D}^T$ and $\mathcal{D}^U$.

Consistent Regularization (CR). Another straight strategy is predictions regularization. Concretely, we design a regularization that explicitly requires the consistency between g^* and the target model:

$$\Omega_D(g_i^*, h_i) = \gamma \sum_{i=1}^{m} \|g_i^* - h_i\|^2, \tag{24}$$

Table 1: ECE↓ on the three datasets. The results show that the proposed methods help calibrate the model when improving the ACC.

Dataset	Method	Distribution Mismatch Measurement				
		0	0.2	0.4	0.6	0.8
M → U	Raw	0.0520	0.1157	0.2512	0.3867	0.4609
	CR	0.0465	0.1105	0.1954	**0.2908**	0.3685
	PE	**0.0169**	0.1598	0.2393	0.3962	0.4527
	SR	0.0410	**0.0762**	**0.1924**	0.2912	**0.3381**
AW → D	Raw	0.1539	0.2619	0.3863	0.2173	0.0924
	CR	0.1818	0.2354	0.1563	0.2038	0.0764
	PE	0.1424	**0.1406**	**0.1281**	**0.1354**	0.1795
	SR	**0.0970**	0.2108	0.1471	0.2024	**0.0527**
AD → W	Raw	**0.0456**	0.1381	0.2094	0.1946	**0.0907**
	CR	0.0601	**0.1182**	0.2094	**0.1602**	0.1983
	PE	0.1672	0.2461	**0.1458**	0.3098	0.2048
	SR	0.0635	0.1661	0.2356	0.2879	0.2663
DW → A	Raw	0.1672	0.2221	0.2409	0.1438	0.0796
	CR	0.1559	0.1905	0.2533	**0.0489**	0.1295
	PE	**0.0731**	0.1835	0.2525	0.0608	0.0892
	SR	0.1373	**0.1141**	**0.1071**	0.0664	**0.0574**
ACW → D	Raw	**0.1511**	0.3467	0.2967	0.3002	**0.2262**
	CR	0.2593	**0.1907**	0.2958	**0.2998**	0.2334
	PE	0.2447	0.3107	0.3019	0.4626	0.5584
	SR	0.1850	0.2053	**0.2582**	0.3656	0.3611
ADW → C	Raw	0.2037	0.1940	0.1323	**0.0770**	**0.0460**
	CR	0.1619	0.2760	0.1171	0.1419	0.1390
	PE	0.2121	0.1732	**0.0683**	0.1464	0.1233
	SR	**0.1250**	**0.1529**	0.1494	0.1092	0.1317

where γ is the hyperparameter that controls the transfer degree. If we set a higher value of γ, the learning process will rely more on g^*. On top of this, we have the following learning objective:

$$\min_h \sum_{i=1}^{m} \ell_i(h) + \Omega_D(g_i^*, h_i). \tag{25}$$

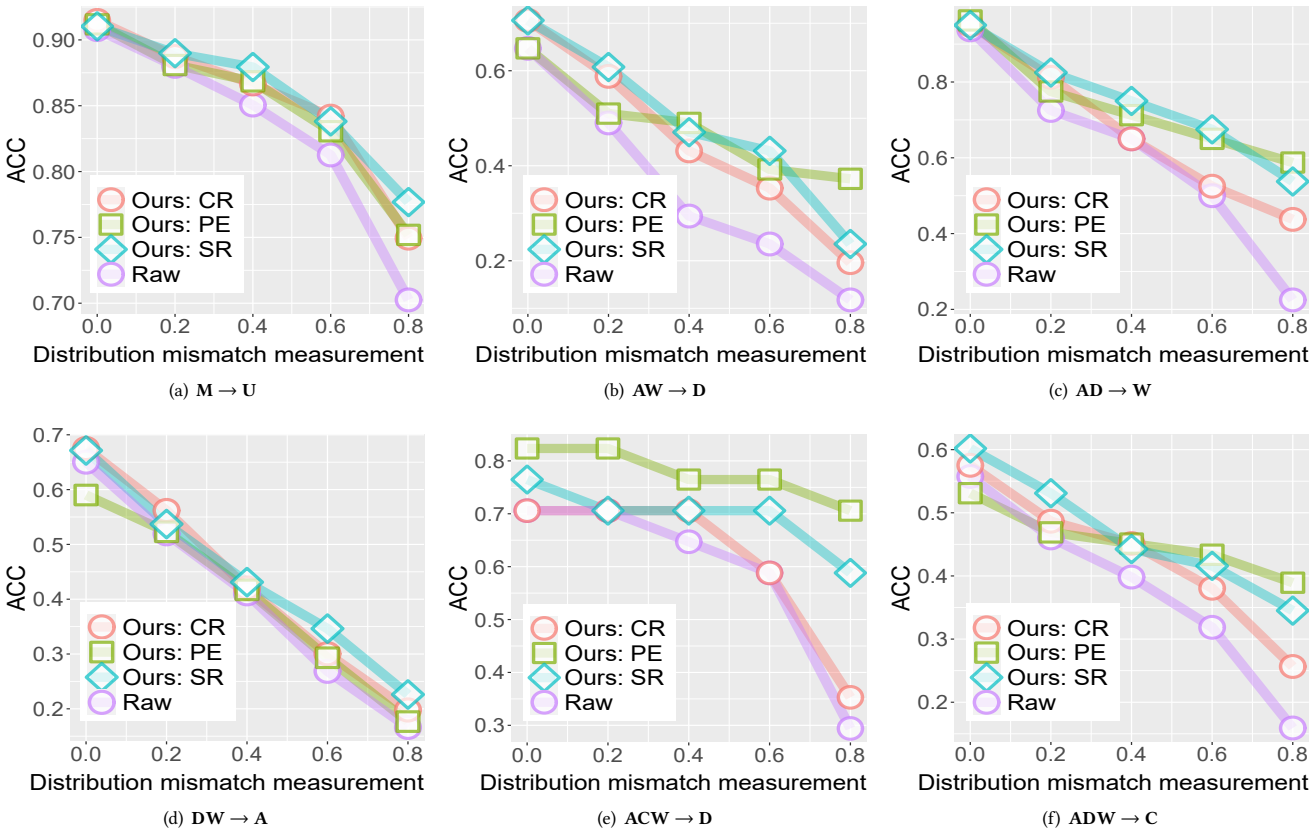
5 EXPERIMENTS

In this section, we conduct experiments on three public datasets to validate the feasibility of cooperated learning under restricted conditions.

5.1 Datasets

We perform evaluations on the following benchmark datasets, where g^* is trained on the source dataset:

- MNIST → USPS. The **MNIST (M)** dataset [24] and the **USPS (U)** dataset [20] are two standard datasets for handwritten digit recognition. The two datasets are drawn from different distributions and both consist of 10 categories. We view MNIST as the source domain and USPS as the target domain, since MNIST has a larger scale. When training the source model, we consider the whole source dataset. Meanwhile,



(a) **M → U** (b) **AW → D** (c) **AD → W**

(d) **DW → A** (e) **ACW → D** (f) **ADW → C**

Figure 2: ACC↑ on three datasets. The proposed methods help improve the performance under different degree of distribution mismatch.

we randomly sample 1,000 data points as the training set for the target task.

- The Office-31 dataset [29] contains 4,652 office images from 31 categories. These images are collected from three domains: **Amazon (A)** downloaded from Amazon.com, **Webcam (W)** recorded with a web camera and **DSLR (D)** captured by a digital SLR camera. We merge two of the domains as the source domain and view the other one as the target domain. Meanwhile, the source data are split into training/validation sets in a 90:10 ratio and the target data into train/validation/test sets in a 90:10:10 ratio.
- Office → Caltech. This dataset consists of the 10 common categories in Office-31 and Caltech-256 [11]. The Caltech dataset only contains one domain (*i.e.*, **Caltech (C)**). We select **DSLR (D)** or **Caltech (C)** as the target domain and the others as the source domain. Meanwhile, the data splitting strategy of the source domain and the target domain is the same as that of Office-31.

5.2 Implementation details

Our experiments are carried out on a Windows 10 desktop with an Intel Xeon E5-2620 v4 CPU and Nvidia TITAN X GPU in Pytorch 1.6 [27]. For the baseline model, we adopt LetNet-5 [24] and ResNet-18 [14] as the backbone for digit recognition and object classification, respectively. We leverage Stochastic Gradient Descent (SGD) as the optimizer and set batch size as 256. Meanwhile, the proposed methods share the same implementations, except that the specific hyperparameter is tuned via a grid search scheme. Specifically, β is tuned within $[0.6, 0.95]$ with a step size 0.05; α is within $[0.6, 1.4]$ with a step size 0.1; and γ is within $\{10^{-\epsilon} : \epsilon \in [0, 8], \epsilon \in \mathbb{N}\}$. Meanwhile, we adopt L2-norm regularization and fix λ as 0.0001.

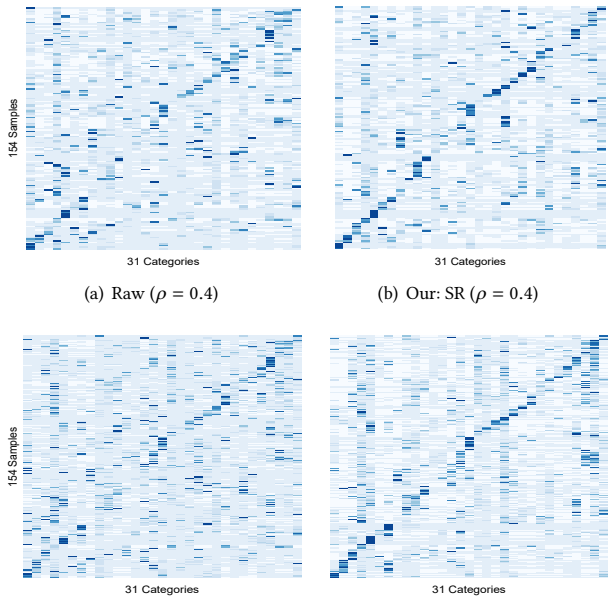
The distribution mismatch measurement we consider is symmetric label noise, which means the distribution of noise is uniform:

$$\mathbb{P}\left(\overline{y}_i \neq y_i\right) = \rho, \tag{26}$$

where ρ is set within $\{0, 0.2, 0.4, 0.6, 0.8\}$. In fact, our methods require no prior knowledge about data distribution and thus are independent of the kind of distribution mismatch. Uniform noise is beneficial to understanding the degree of distribution mismatch intuitively.

5.3 Evaluation Metrics

We adopt the following metrics to compare the performance of models:



(a) Raw ($\rho = 0.4$) (b) Our: SR ($\rho = 0.4$)

(c) Raw ($\rho = 0.6$) (d) Our: SR ($\rho = 0.6$)

Figure 3: Prediction results of noisy samples on AW → D. Our predictions concentrate more on the diagonal, which means the proposed method is more robust to the distribution mismatch.

- ACC. The accuracy of target model prediction is the most common metric in classification tasks.
- Expected Calibration Error (ECE) [12]. A model should provide a calibrated confidence measure associated with the predicted class label, which is essential for some tasks like self-driving. ECE measures the degree of mis-calibration by partitioning predictions into M bins and taking a weighted average of the accuracy/confidence error in each bin:

$$\mathrm{ECE} = \sum_{j=1}^{M} \frac{|B_j|}{m} \left|\mathrm{Acc}\left(B_j\right) - \mathrm{Conf}\left(B_j\right)\right|, \tag{27}$$

where the average confidence within bin B_j is defined as:

$$\mathrm{Conf}\left(B_j\right) = \frac{1}{|B_j|} \sum_{i \in B_j} h_i. \tag{28}$$

It is worth mentioning that model calibration is inconsistent with accuracy, which means a highly accurate model might perform badly on calibration metrics. We set $M = 15$ and report the value when the model reaches the highest accuracy.

5.4 Results

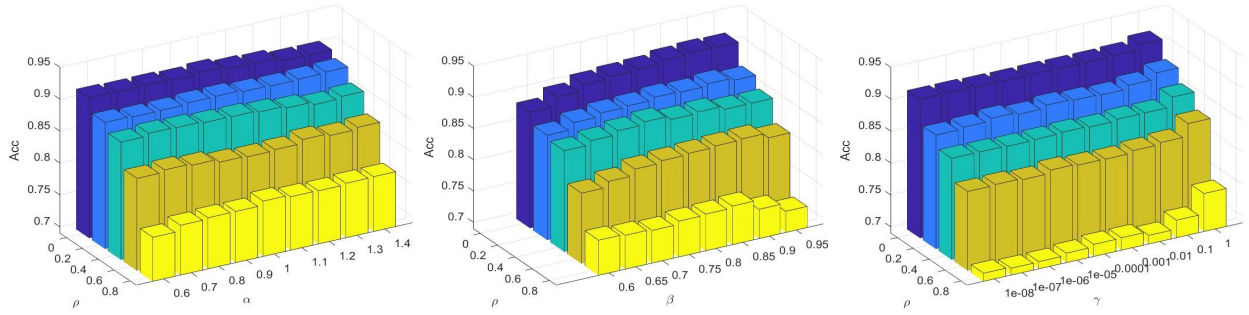
All the results are reported in Figure 2 and Table 1, where *Raw* represents the standard machine learning method, *CR*, *PR*, and *SR* are the proposed confederated learning methods. Meanwhile, the best performance is marked with orange in Table 1. Then, we could make the following observations:

- When there exists distribution mismatch between the target training and test set, all the proposed methods could improve the target performance and confidence calibration on most tasks, which validates the effectiveness of the proposed methods and the feasibility of confederated learning. For example, SR, PE, CR outperform the original model by 17.65%, 19.61%, and 13.72% on the **AW → D** dataset when $\rho = 0.4$, respectively.
- It is worth mentioning that Sample Reweighting and Consistent Regularization still improve the performance of the target model when there exists no distribution mismatch. This phenomenon shows that the proposed methods do boost the target model on the top of model predictions g_i^*.
- Prediction Ensemble becomes invalid in the **ADW → C** dataset. This failure case illustrates that it might be a suboptimal strategy to explicitly merge the model predictions in the context of confederated learning.
- The improvement in the **AD → W** dataset is not so significant as other situations, which shows the necessity of correlation between D^U and the D^T.
- Among the proposed methods, Sample Reweighting achieves the best performance in most situations, which might benefit from the specific training weight function w.
- As the value of distribution mismatch measurement becomes higher, the improvement also becomes more significant. For example, SR outperforms the original model by 0.40%, 2.89% and 7.43% on the USPS dataset when the distribution mismatch equals 0, 0.4 and 0.8, respectively.
- Comparing the performance on **AW → D** and **ACW → D** (*i.e.,* Figure 2 (b) and Figure 2 (e)), we could find that more powerful g^* can induce a more robust performance *w.r.t.* distribution mismatch.

5.5 Visualization

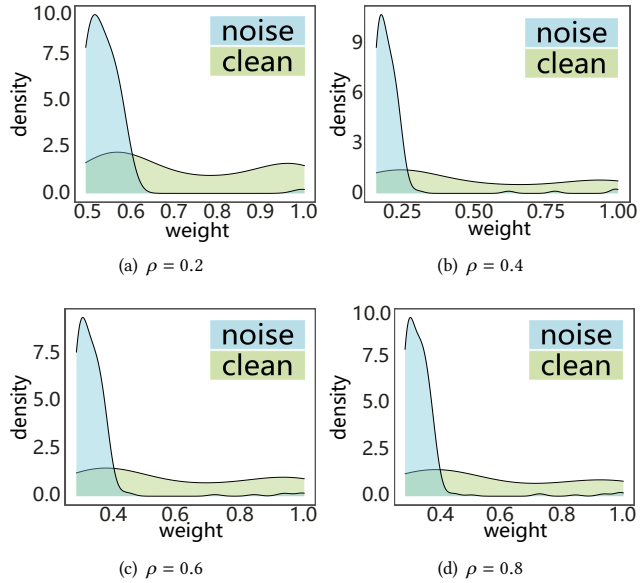
In order to make a finer-grained comparison, we plot the prediction results of noisy samples on **AW → D** when ρ equals 0.4 and 0.6 in Figure 3. To be concise, we rearrange the samples according to their labels. Since the x-axis of Figure 3 is also the rearranged labels, the ground-truth prediction will have a block-diagonal structure. In other words, if the heatmap of a prediction concentrates more on the diagonal, it means the prediction correctly classifies more samples. As we can see, the heatmap of SR clearly concentrate more on the diagonal, which again validates the effectiveness of the proposed method.

Besides, we also plot the distribution of the weight w_i on **AW → D** in Figure 5. We find that the distribution of noisy samples is skewed to the left, which represents they are assigned a lower weight. Meanwhile, the clean sample distribution is relatively evenly. In a word, the proposed methods eliminate the effect of data distribution mismatch and thus help improve the performance.



(a) SR (b) PE (c) CR

Figure 4: Sensitivity analysis on the USPS dataset: (1) SR is not sensitive to the choice of hyperparameter α. (2) The choice of β depends on the degree of distribution mismatch. (3) CR does help improve the performance since the performance degenerates as the hyperparameter γ decreases.



(a) $\rho = 0.2$ (b) $\rho = 0.4$

(c) $\rho = 0.6$ (d) $\rho = 0.8$

Figure 5: The distribution of w_i on AW → D. The noisy samples are assigned with lower weights, while the weights of clean samples are higher. The skewed distribution validates the effectiveness of g^* in recognizing distribution mismatch.

5.6 Sensitivity analysis

Next, we study the influence of the hyperparameters that control the transfer degree (*i.e.*, α, β and γ). We plot the results in Figure 4. We could make the following observations:

- The performance of Sample Reweighting is not sensitive to the choice of hyperparameter α, which might result from the assigned weights accurately filtering out the noise in training data.
- The choice of β depends on the degree of distribution mismatch. When ρ equals to 0, Prediction Ensemble achieves the best performance when $\beta = 0.95$. In other words, the original labels contribute more to the final performance. As the distribution mismatch increases, the best performance relies more on g^*.
- Consistent Regularization achieves the best performance when γ is set to 1. The performance degenerates as we decrease the hyperparameter γ, which validates that the proposed regularization helps the model free from the impact of label noise.

6 CONCLUSION

In this paper, we consider the cooperation scenario where multiple entities are granted different roles with the corresponding permission, called confederated learning. Systematic analysis shows that confederated learning generalizes the paradigm of traditional machine learning and federated learning. Then, we study an application scenario of confederated learning, where the permission of each entity is restricted due to cost, privacy, and security concerns. We further propose three methods as the first trial for such a cooperation scenario: Sample Reweighting, Prediction Ensemble, and Consistent Regularization. All the three methods show encouraging empirical results on three public datasets. We hope this work could pave the way for future research in the context of cooperated learning.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102000, in part by National Natural Science Foundation of China: U21B2038, U1936208, 61931008, 62132006, 6212200758, 61976202, and 62006217, in part by the Fundamental Research Funds for the Central Universities, in part by Youth Innovation Promotion Association CAS, in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDB28000000, and in part by China Postdoctoral Science Foundation: 2021T140653 and 2020M680651.

REFERENCES

[1] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. 2014. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.* 15 (2014), 1111–1133.

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Awni Y. Hannun, Billy Jun, Tony Han, Patrick LeGresley, Xiangang Li, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Sheng Qian, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Chong Wang, Yi Wang, Zhiqian Wang, Bo Xiao, Yan Xie, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2016. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *International Conference on Machine Learning*. 173–182.

[3] Paul Baran. 1964. On distributed communications networks. *IEEE transactions on Communications Systems* 12 (1964), 1–9.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* abs/1604.07316 (2016).

[5] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45 (2001), 5–32.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeuIPS*.

[7] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Symposium on Operating System Design and Implementation*. 137–150.

[8] Lixin Duan, Ivor W. Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*. 289–296.

[9] Ian T. Foster and Adriana Iamnitchi. 2003. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *Proceedings ofthe InternationalWorkshop on Peer-to-Peer Systems*. 118–128.

[10] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. In *ACM Symposium on Operating Systems Principles*. 29–43.

[11] Gregory Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 object category dataset. (2007), 1 – 20.

[12] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML*. 1321–1330.

[13] Minyang Han and Khuzaima Daudjee. 2015. Giraph Unchained: Barrierless Asynchronous Parallel Execution in Pregel-like Graph Processing Systems. *Proc. VLDB Endow.* 8 (2015), 950–961.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[15] István Hegedüs, Gábor Danner, and Márk Jelasity. 2019. Gossip Learning as a Decentralized Alternative to Federated Learning. In *Distributed Applications and Interoperable Systems*. 74–90.

[16] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security*. 603–618.

[17] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A. Gibson, Gregory R. Ganger, and Eric P. Xing. 2013. More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server. In *Advances in Neural Information Processing Systems*. 1223–1231.

[18] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *USENIX Symposium on Networked Systems Design and Implementation*. 629–647.

[19] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. 2006. Correcting Sample Selection Bias by Unlabeled Data. In *NeuIPS*. 601–608.

[20] Jonathan J. Hull. 1994. A Database for Handwritten Text Recognition Research. *TPAMI* 16 (1994), 550–554.

[21] Sylvain Jeaugey. 2017. Nccl 2.0. In *GPU Technology Conference (GTC)*.

[22] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *ICML*. 1885–1894.

[23] Jakub Konecný, Brendan McMahan, and Daniel Ramage. 2015. Federated Optimization: Distributed Optimization Beyond the Datacenter. *CoRR* abs/1511.03575 (2015).

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86 (1998), 2278–2324.

[25] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S. Yu. 2013. Transfer Feature Learning with Joint Distribution Adaptation. In *ICCV*. 2200–2207.

[26] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of Machine Learning*. MIT Press.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeuIPS*. 8024–8035.

[28] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to Reweight Examples for Robust Deep Learning. In *ICML*. 4331–4340.

[29] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. 2010. Adapting Visual Category Models to New Domains. In *ECCV*. 213–226.

[30] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security*. 1310–1321.

[31] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *CoRR* abs/1812.00564 (2018).

[32] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. 2020. A Survey on Distributed Machine Learning. *ACM Comput. Surv.* 53 (2020), 30:1–30:33.

[33] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime G. Carbonell. 2019. Characterizing and Avoiding Negative Transfer. In *CVPR*. 11293–11302.

[34] Jinliang Wei, Wei Dai, Aurick Qiao, Qirong Ho, Henggang Cui, Gregory R. Ganger, Phillip B. Gibbons, Garth A. Gibson, and Eric P. Xing. 2015. Managed communication and consistency for fast data-parallel iterative analytics. In *ACM Symposium on Cloud Computing*. 381–394.

[35] Eric P Xing, Qirong Ho, Pengtao Xie, and Dai Wei. 2016. Strategies and principles of distributed machine learning on big data. *Engineering* 2 (2016), 179–195.

[36] Zhenwei Zhang and Ervin Sejdic. 2019. Radiological images and machine learning: Trends, perspectives, and prospects. *Comput. Biol. Medicine* 108 (2019), 354–370.

[37] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. CRC press.