Parallel Assisted Learning

Xinran Wang, Jiawei Zhang, Mingyi Hong, Yuhong Yang, Jie Ding

***Abstract*—In the era of big data, a population's multimodal data are often collected and preserved by different business and government entities. These entities often have their local machine learning data, models, and tasks that they cannot share with others. Meanwhile, an entity often needs to seek assistance from others to enhance its learning quality without sharing proprietary information. How can an entity be assisted while it is assisting others? We develop a general method called Parallel Assisted Learning that applies to the context where entities perform supervised learning and can collate their data according to a common data identifier. Under the PAL mechanism, a learning entity that receives assistance is obligated to assist others without the need to reveal any entity's local data, model, and learning objective. Consequently, each entity can significantly improve its particular task. The applicability of the proposed approach is demonstrated by data experiments.**

***Index Terms*—assisted learning, cooperative learning, decentralization, federated learning, machine learning.**

I. Introduction

There has been a rapid growth of large-scale multimodal data generated and privately held by decentralized entities. Examples are business and government entities that collect the same population's demographic, activity, and healthcare information. An entity often has its particular data, training model, and learning objective treated as proprietary assets. We envision that these entities may often want to coordinate with each other in an appropriate and privacy-preserving manner to acquire a diverse body of information.

Motivated by the above context, we develop operational methods for assisted learning. Any entity can seek personalized assistance from others to enhance its learning quality without sharing proprietary information. In our context, an entity can seek help by broadcasting task-oriented but nonsensitive statistics and incorporating others' feedback in one or more iterations to enhance its prediction performance. Moreover, two entities may wish to simultaneously improve their learning in an interconnected manner. Instead of running two separate assistance, they can bind their learning so that each entity has to assist the other if it is being assisted. In many cooperative learning scenarios, especially those involving competing entities, such a mechanism will ensure that each entity contributes positively to others. Our work is thus motivated by the following question. *How to bind each entity to assist others while it is being assisted?*

We will develop a general method called Parallel Assisted Learning (PAL) to realize such reciprocal learning in this work. PAL will enable collaborations among heterogeneous entities (with distinct and private objectives) to accomplish entity-specific missions. Our work was inspired by Pareto Improvement [1], a notion in economics that concerns the improvement of individuals' utilities without harming else. An essential characteristic of PAL is that the assistance allows each entity to have its unique input data, model, and objective (task labels), which are not required to be shared. This renders its fundamental difference from existing distributed learning frameworks such as Federated Learning [2]–[6], which need to operate on a globally shared model and objective.

X. Wang, J. Zhang, Y. Yang, and J. Ding are with the School of Statistics, University of Minnesota, Twin Cities, MN 55414, USA. M. Hong is with the Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities, MN 55414, USA.

Two examples are provided below to illustrate the application scenarios. We will revisit them in the experiments.

Example 1. Two clinics in a city hold features collected from the same set of patients. The data can be collated by a non-private patient ID in hindsight. One clinic holds features from lab tests, and the other holds pharmaceutical features. Due to limited infrastructures or regulation constraints, they cannot migrate data to each other. Also, each of them has a private learning task. They will invoke the PAL protocol to improve their single-clinic performance. Ideally, they will achieve near-oracle performance as if data were centralized and the tasks could be transparent.

Example 2. Two organizations collect surveys from the same cohort of mobile users. The user data can be collated by a non-private username. One collects economic features such as working hours and salary level, and the other focuses on demographic information such as gender and age. They may assist with each other's learning tasks in a privacy-preserving manner.

In general, PAL is expected to be useful in one or more of the following scenarios. First, an entity does not want to provide dedicated resources (such as machines or cloud-based interfaces) to assist others before it sees the performance gain from being assisted. Second, an entity favors such an incentive that collaborators will not cheat during both training and prediction stages, which may cause terminated collaboration and hinder everyone's performance gain otherwise. Third, an entity without an established reputation can still collaborate with known brands in an economical way. We will provide experimental results to support the envisioned use cases.

A. Contributions

The main contributions of this work are threefold.

- First, we introduce a new notion of cooperative learning called Parallel Assisted Learning. It is suitable for emerging machine learning services, where entities collecting heterogeneous features from the same cohort of objects can potentially assist each other's learning tasks. PAL is privacy-preserving in the sense that no entity will need to share its local data, model, or learning task.
- Second, we develop implementable methods and practical

guides to materialize PAL. The methods are based on the idea that entities interactively build local models by transmitting a blend of task-specific statistics and then provide assistance during the prediction stage.

• Third, we show that the proposed learning protocol can be applied to various learning tasks, including regression, classification, and classification-regression mixture. Under some conditions, we prove that each entity can achieve near-oracle prediction performance as if it had used all the data to train the model. The oracle provides a theoretical limit on what PAL can maximally bring to each entity.

Overall, PAL provides a win-win solution for entities that own unique modeling and data resources. To the best of the authors' knowledge, this is the first solution for multiple entities to perform learning tasks in parallel without sharing proprietary local data, models, and task labels.

B. *Related work*

Suppose multiple entities learn the same task separately based on their local data and models. In predicting the data input of a future object, an entity may directly query the prediction results from other entities and then combine the results (often linearly). Such a procedure, without joint modeling or coordination during the training stage, is often studied in the literature on statistical model averaging and expert learning [7], [8]. It aims to approach the prediction performance of the best entity in hindsight. Nevertheless, it does not fully utilize all the available data to train a powerful model since entities have no information exchange during the training.

Many distributed learning methods have been developed for an entity to achieve near-oracle performance, as if all the local resources were centralized. A popular direction of research is Federated Learning (FL) [2]–[6], where the main idea is to learn a joint model using the averaging of locally learned model parameters so that the training data do not need to be transmitted. The ultimate goal of FL is to exploit the resources of massive edge devices (also known as 'clients') for achieving a global objective orchestrated by a central server. Most existing work focused on the so-called horizontally partitioned data, meaning that clients hold local data of the same feature variables but from different objects. Another type of data, which we consider in this work, is known as vertically partitioned data, meaning that entities hold data of different feature variables collected from the same cohort of objects (illustrated in Figure 1). Under FL, this data format is studied in the context of the so-called vertical FL [9]–[11].

Unlike FL, the purpose of PAL is to facilitate multiple entities to assist each other's private learning tasks autonomously. These entities (e.g., research institutes, government agencies, and companies) are often rich in computation resources. However, they are restricted by the universe of variables and security constraints that prohibit sharing proprietary input data, training models, or objectives (task labels). Motivated by these practical considerations, PAL differs from FL in the following aspects. First, PAL is decentralized in that each entity can autonomously launch a local task to solicit assistance and participate in cooperation without a central coordinating server, while FL typically assumes the existence of a central server. Second, different from FL, the models and learning objectives of PAL entities can be different and need not be shared. Vertical FL algorithms, such as those proposed in [11], often build a large central model, where each client will contribute to such a model building by utilizing local data and local parameters. More specifically, vertical FL aims to build a global model by aggregating the distributed feature observed at each client and testing it on future data of complete features. In contrast, in the proposed work, each participating entity has its own learning objective, trains local models that only operate on data of local features in the test/prediction stage, and aims to make better local predictions through mutual assistance.

In line with the above discussions, our work is more closely related to a recent development in decentralized learning called Assisted Learning (AL) [12]. The earlier work of AL aims to improve a particular learning task of an entity regardless of other entities' tasks. We will revisit AL in Subsection II-C and show that it can be cast as a particular case of PAL when only one entity contributes task labels.

The rest of this paper is organized as follows. In Section II, we introduce the background and formulation of PAL. We also provide some sufficient conditions to guarantee the theoretical performance and discuss its extension to classification learning. In Section III, we demonstrate the proposed method with experimental studies using both synthetic and real-world datasets. We conclude the paper in Section IV.

II. General Description of PAL

A. *Notation*

We will focus on supervised learning in this paper. We use $x \in \mathcal{X} \subseteq \mathbb{R}^p$ and $y \in \mathcal{Y} \subseteq \mathbb{R}$ to denote the input data variable and prediction label, respectively. We will often use a function $f : \mathcal{X} \to \mathcal{Y}$ to represent a model. Let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+ \cup \{0\}$ denote the loss function that maps a true label y and its predicted label $\hat{y}$ to $\ell(y, \hat{y})$. Let $\mathbb{E}$ and $\mathbb{E}_n$ denote the expectation and empirical expectation (namely average over a sample), respectively. So, $\mathbb{E}_n g(y, x) = n^{-1} \sum_{i=1}^{n} g(y_i, x_i)$ for any measurable function g, where y_i and x_i denote the ith observation of y and x, respectively. With a slight abuse of notation, we may sometimes use x and y to denote the $n \times p$ data matrix that stacks n observations of the input variables and the $n \times 1$ vector of labels, respectively. Let $\|\cdot\|_2$ denote the Euclidean norm. A sequence of nonzero vectors $z_1, z_2, \dots$ is said to converge R-linearly to z if $\|z_r - z\|_2 \leq \gamma \rho^r$ for all r, for some scalars $\gamma > 0$ and $\rho \in (0, 1)$.

The paper will study interactions between multiple entities, each storing a model and dataset. As such, we will introduce the notion of a module that represents the private resource held by an entity: *a module $\mathcal{M} = (\mathcal{A}, x)$ is a pair of algorithm $\mathcal{A}$ and input data $x \in \mathcal{X}^{n \times p}$ such that for any task-specific label vector $y \in \mathcal{Y}^n$, it will induce a prediction model denoted by $f_{\mathcal{M} \leftarrow y}$.* For example, $f_{\mathcal{M} \leftarrow y}$ is obtained by solving an empirical risk minimization problem $f_{\mathcal{M} \leftarrow y} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(y_i, f(x_i))$, where $\mathcal{F}$ is a function class. In particular, if $\mathcal{A}$ represents the least squares method, $f_{\mathcal{M} \leftarrow y}$ becomes $\tilde{x} \mapsto \tilde{x}^{\mathrm{T}} (x^{\mathrm{T}} x)^{-1} x^{\mathrm{T}} y$ for any feature vector $\tilde{x} \in \mathbb{R}^p$.

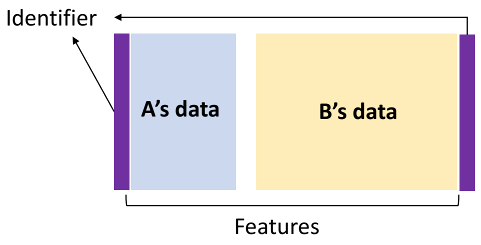

Fig. 1: Illustration of the data collation.

In the context of privacy-preserving learning, $\mathcal{M} = (\mathcal{A}, x)$ is treated as private. To learn a task represented by y, the entity can obtain $f_{\mathcal{M} \leftarrow y}$ as the operational model. However, this model may not be adequate due to a lack of information in this entity's x regarding y. Thus, this entity may need assistance from a peer module observing the same cohort of data objects. Then, we introduce the following notion of data collation.

We say *two datasets (matrices or column vectors) are collated if their rows can be aligned by a unique data identifier.* For example, the identifier can be a timestamp, username, or unique object index. The identifier is assumed to be non-private. In this work, we assume that the data of all the modules can be collated (see Figure 1).

B. Problem Formulation

Suppose that there are m entities (or modules), indexed by $I = \{1, \ldots, m\}$. Each entity i consists of data $x_i \in \mathbb{R}^{n \times p_i}$ and a learning algorithm based on the model $f_{i,\theta} : \mathbb{R}^{p_i} \rightarrow \mathbb{R}$ with $\theta \in \Theta_i$. Here, p_i is the input dimension of entity i, while p in Subsection II-A was used for a generic entity. The function form $f_{i,\cdot}$, parameter space Θ_i, and dimension p_i may differ among different i's. We first consider regression learning. Extensions are included in Subsection II-F.

Suppose that each entity i will have a private learning task, represented by the response vector $y_i \in \mathbb{R}^n$. Each entry in y_i and the associated row in x_i correspond to a data object (e.g., a patient, a mobile user). Ideally, any entity, denoted by $j \in I$, aims to build the regression model $\mathbb{E}(y_j \mid x_i : i \in I) \triangleq f_j(x_i : i \in I)$ using all the variables held by other entities. Here, y_j is a random variable representing the task of entity j, and f_j is the regression function. We suppose that f_j can be written in an additive form $f_j(x_i : i \in I) = \sum_{i \in I} f_{i,j}(x_i)$, where $f_{i,j}$ only operates on the entity i's data. We assume that $f_{i,j}$ is parameterized by $\theta_{i,j} \in \Theta_i$, which represents *the entity i's model parameter for the entity j's learning task.* For technical convenience, we assume such a parameterization is well-specified in the sense that the data-generating $f_{i,j}$ falls into the parametric form.

Assumption 1. We can express f_j in the form of

$$f_j(x_i : i \in I) = \sum_{i \in I} f_{i,\theta_{i,j}}(x_i), \quad \text{with } \theta_{i,j} \in \Theta_i. \tag{1}$$

If the entity j could centralize the data $\{x_i : i \in I\}$ and know the parametric forms, it would typically estimate the unknown parameters by solving the following empirical risk minimization problem.

$$\text{For } j : \{\hat{\theta}_{i,j}\}_{i \in I} = \underset{\{\theta_{i,j}\}_{i \in I}}{\arg\min} \, \mathbb{E}_n \ell\big(y_j, \sum_{i \in I} f_{i,\theta_{i,j}}(x_i)\big). \tag{2}$$

It is common to append a regularization term in (2), but we omit it for brevity. In the prediction stage, consider an object whose ith modality, denoted by x_i^f, is observed by entity i ($i \in I$). Then, the entity j will use $\sum_{i \in I} f_{i,\hat{\theta}_{i,j}}(x_i^f)$ to predict the corresponding response y_j^f of its interest. In other words, the entity j will only need each entity i ($i \neq j$) to share the value of $f_{i,\hat{\theta}_{i,j}}(x_i^f)$.

Moreover, suppose that all the m entities need to simultaneously accomplish their separate tasks in the form of (2). Our goal is to develop a learning strategy that features the following characteristics:
• Privacy: None of the entities needs to share its data x_i, form of $f_{i,\cdot}$ (and its estimates), or task label y_i with other entities.
• Parallel assistance: An entity has to contribute to others' learning tasks if it needs to improve its own learning.

C. Unilateral assistance

We first introduce the method developed in [12] for solving the above (2) for a single entity j, without considering other entities' tasks. To avoid sharing sensitive information in the training stage, we need to operate the problem (2) in each entity's locality. For notational brevity, we describe the method with two learners, denoted by A and B, and suppose that B will assist A. Like before, we use the subscript i, j to highlight the entity i's model for the entity j's task.

The entity A will first initialize with $y_{a,a}^{(1)} = y_a$. At round k ($k = 1, 2, \ldots$), A fits a model by using $(y_{a,a}^{(k)}, x_a)$ as labeled data and solving

$$\hat{\theta}_{a,a}^{(k)} = \underset{\theta_a \in \Theta_a}{\arg\min} \, \mathbb{E}_n \ell(y_{a,a}^{(k)}, f_{a,\theta_a}(x_a))$$

Then, A will send the fitted residual of the current round, $r_{a,a}^{(k)} = y_{a,a}^{(k)} - f_{a,\hat{\theta}_{a,a}^{(k)}}(x_a)$, to B. In its locality, B will fit a model by using $(r_{a,a}^{(k)}, x_b)$ as the labeled data and solving

$$\hat{\theta}_{b,a}^{(k)} = \underset{\theta_b \in \Theta_b}{\arg\min} \, \mathbb{E}_n \ell(r_{a,a}^{(k)}, f_{b,\theta_b}(x_b)).$$

Then, B will calculate the residual $r_{b,a}^{(k)} = r_{a,a}^{(k)} - f_{b,\hat{\theta}_{b,a}^{(k)}}(x_b)$ and send $y_{a,a}^{(k+1)} \triangleq r_{b,a}^{(k)}$ to A.

The intuition behind the above training process is that an entity will iteratively send the current residual and outsource predictable information from others. Consider the case where each entity uses linear regression models and least squares estimation. The process can be regarded as iterative projections of task labels onto the column space of each entity's data matrix to approximate the projection onto the space of their collated data matrix in hindsight [12]. Consequently, during the prediction stage, to mimic the prediction result of the oracle model estimated from centralized data, A will need to aggregate the prediction results of all local models of A and B obtained along the process. Consider a future object whose modality x_i^f is observed by entity i. Then, A will use $\hat{y}_a^f \overset{\Delta}{=} \sum_{\ell=1}^k f_{a,\hat{\theta}_{a,a}^{(\ell)}}(x_a^f) + \sum_{\ell=1}^k f_{b,\hat{\theta}_{b,a}^{(k)}}(x_b^f)$ to predict the response y_a^f of its interest.

We will see in Subsection II-E that unilateral assistance can be regarded as a particular case of PAL, which can integrate concurrent tasks into training and prediction stages.

D. PAL: from unilateral to reciprocal assistance

Suppose that A has a learning objective with the label y_a, and B also has a learning objective with the label y_b. A straightforward solution is to separately run two unilateral assistance, first letting B assist A, and then the other way around. But what if A is not cooperative: after B assists A in the *training stage*, A will not assist B? In practice, if that occurs, B can choose not to assist A in the *prediction stage*. But that is not desirable, as B already consumed computation resources, and B would like to charge A's prediction queries. We thus ask the question: *How can an entity be assisted while it is assisting others?*

Our main idea is to let each entity seek assistance from others by iteratively exchanging non-private statistics. Such statistics received by an entity will contain unknown task information deliberately injected by others so that each entity has an incentive to 'do well.' In the *training stage* (Stage I), at the first round of assistance, A sends its residual vector to B. Upon receipt of the query, B blends the received vector with some statistics $s_{b,a} \in \mathbb{R}^n$ calculated from its local task, treats it as learning labels, and fits a model. After that, B sends the residual back to A, who will initialize the next round. The above procedure continues until an appropriate stop criterion is triggered. The training stage initialized by A is then completed. After that, B will initialize another training stage by swapping the role with A. After the second training stage is completed, both entities can proceed to the prediction stage.

In the *prediction stage* (Stage II), upon arrival of a new feature vector, A queries the prediction results from B's local models and combines them with its own to form the final prediction. So does B. Each entity will need to tell how they blend the learned labels to 'decode' the faithful prediction values.

A particular PAL protocol. We suggest the following specific protocol as a default solution. At the beginning, the entity B will train a local model using (y_b, x_b), represented by $\hat{\theta}_b$; Then, B will calculate the residual $r_b \overset{\Delta}{=} y_b - f_{b,\hat{\theta}_b}(x_b)$. Likewise, the entity A does so in its locality and obtains the fitted residual r_a. In the training stage initialized by A, it will send r_a to B, who will blend $s_{b,a} \overset{\Delta}{=} \tau_b r_b$ into r_a (after data collation). Then, B will treat $r_a + \tau_b r_b$ as its learning labels, and re-learns a model. Here, $\tau_b \neq 0$ is a constant specified by B, and it will be kept a secret from A until the training stage is over. B will then send the most recent residual back to A, who will then treat it as the label and learn a local model. Then, A will start the second round of learning. In this protocol, the entity B only injects information at the first round of training but not further rounds. The above procedure is illustrated in Table I. We will run the procedure another time, but swapping A and B's roles. The pseudocode is summarized in Algorithm 1. We will explain more details in the following remarks and discuss some extensions of the algorithm in Subsection II-F.

Intuitive explanations. Since there is no further injection made after the first round in the above protocol, B's actual targeted label from the first round is

$$r_a + \tau_b r_b = y_a + \tau_b y_b - (f_{a,\hat{\theta}_a}(x_a) + \tau_b f_{b,\hat{\theta}_b}(x_b)).$$

In other words, the procedure can be seen as a unilateral assistance (Subsection II-C) from B to A in fitting the blended label $r_a + \tau_b r_b$. Accounting for the A's model $\hat{\theta}_a$ and B's model $\hat{\theta}_b$ locally stored at the beginning, A and B are cooperatively learning $\tilde{y}_a \overset{\Delta}{=} y_a + \tau_b y_b$ and predicting it at the prediction stage (once predictions from the local models are aggregated). Likewise, when B initializes an assisted learning, A will blend the received residual with $\tau_a r_a$, so the targeted label is virtually $\tilde{y}_b = \tau_a y_a + y_b$. As long as A knows τ_b and B knows τ_a (with $\tau_a \tau_b \neq 1$), we have

$$y_a = \frac{\tilde{y}_a - \tau_b \tilde{y}_b}{1 - \tau_a \tau_b}, \quad y_b = \frac{\tilde{y}_b - \tau_a \tilde{y}_a}{1 - \tau_a \tau_b}. \tag{3}$$

As a result, both A and B will be able to 'decode' their wanted task labels during the prediction stage. The notions of $\tilde{y}_a$ and $\tilde{y}_b$ at Line 3, Stage II of Algorithm 1 have the same interpretation as here, except that they are finite-sample estimates.

In summary, PAL is operated in the following manner. In the training stage, A and B build and store their local models iteratively (Stage I of Algorithm 1). In the prediction stage, these models' prediction results for a future object are aggregated to estimate $\tilde{y}_a$ (initialized by A) and $\tilde{y}_b$ (initialized by B), which will then be used to decode y_a and y_b according to Equation (3) (Stage II of Algorithm 1). Throughout the above procedure, each entity acts as a separate module (see Subsection II-A).

Remark 1 (The purpose of blending)**.** The blending aims to ensure that A and B will complete the training stage together. If they do not cooperate, they cannot decode desirable results in the prediction stage. It is because an entity does not know the injected statistics from the other side (even if τ_a, τ_b are public). If they are dedicated to the fitting of $\tilde{y}_a$ and $\tilde{y}_b$ in assisted training, they can model these blended labels and further obtain their own predictions through Equation (3).

Remark 2 (The choices of τ-values)**.** What values do τ_a and τ_b take? The following factors may need to be considered. First, according to Equation (3), it is required that $\tau_a \tau_b \neq 1$. Second, it is conceivable that $\tau_b \neq 0$. Otherwise, the procedure initialized by A reduces to the unilateral assistance in Subsection II-C, which is not favorable to B. Likewise, it is appealing that $\tau_a \neq 0$. Third, since $\tau_b y_b = (\tau_b/\lambda)(\lambda y_b)$ for any nonzero constant λ, without loss of generality, we suppose that each entity will re-scale its response vector so that its ℓ_2-norm becomes n. Correspondingly, if τ_b is very large, the signal of y_a will be overwhelmed by y_b in the first training round initialized by A, which is virtually assisting B. Thus, we suggest each entity standardize the response and restrict their tau to be within a range (say $[0, 1]$ in absolute value) to promote benign cooperation. An example mechanism to take all the

TABLE I: Illustration of the assisted training initialized by the entity A.

	Initialization	Round 1		Round k ($k \geq 2$)	
A	fit y_a, obtain r_a		fit $r_{b,a}^{(1)}$, obtain $r_{a,a}^{(2)}$		fit $r_{b,a}^{(k)}$, obtain $r_{a,a}^{(k+1)}$
B	fit y_b, obtain r_b	fit $r_a + \tau_b r_b$, obtain $r_{b,a}^{(1)}$		fit $r_{a,a}^{(k)}$, obtain $r_{b,a}^{(k)}$	

Algorithm 1 Parallel Assisted Learning: A and B assisting each other

```
Input: Entity A with task label $y_a \in \mathbb{R}^n$, model $f_{a,\theta_a} : \mathbb{R}^{p_a} \to \mathbb{R}$ ($\theta_a \in \Theta_a$), and data $x_a \in \mathbb{R}^{n \times p_a}$. Entity B with task label $y_b \in \mathbb{R}^n$, model $f_{b,\theta_b} : \mathbb{R}^{p_b} \to \mathbb{R}$ ($\theta_b \in \Theta_b$), and data $x_a \in \mathbb{R}^{n \times p_b}$. Future objects to predict.
    Note: the notation $\square_{i,j}$ denotes entity $i$'s quantity for assisting entity $j$, for $i, j \in \{a, b\}$.
    Stage I: Training
1: A fits a model using $(y_a, x_a)$ as labeled data, by solving $\hat{\theta}_a = \arg\min_{\theta_a \in \Theta_a} \mathbb{E}_n \ell(y_a, f_{a,\theta_a}(x_a))$.
2: B fits a model using $(y_b, x_b)$ as labeled data, by solving $\hat{\theta}_b = \arg\min_{\theta_b \in \Theta_b} \mathbb{E}_n \ell(y_b, f_{b,\theta_b}(x_b))$.
3: A calculates the residual $r_a = y_a - f_{a,\hat{\theta}_a}(x_a)$, and chooses a value $\tau_a \in [-1, 0)$.
4: B calculates the residual $r_b = y_b - f_{b,\hat{\theta}_b}(x_b)$, and chooses a value $\tau_b \in (0, 1]$.
5: A initializes $r_{a,a}^{(k)} = r_a$, round $k = 1$.
6: for $k = 1, 2, \ldots$ (until a stop criterion is satisfied) do
7:     A sends $r_{a,a}^{(k)}$ to B. Let $\mathbb{1}_{k=1}$ denote one if $k = 1$ and zero otherwise.
8:     B fits a model using $y_{b,a}^{(k)} \triangleq r_{a,a}^{(k)} + \mathbb{1}_{k=1} \tau_b r_b$ as the label by solving $\hat{\theta}_{b,a}^{(k)} = \arg\min_{\theta_b \in \Theta_b} \mathbb{E}_n \ell(y_{b,a}^{(k)}, f_{b,\theta_b}(x_b))$.
9:     B calculates the residual $r_{b,a}^{(k)} = y_{b,a}^{(k)} - f_{b,\hat{\theta}_{b,a}^{(k)}}(x_b)$.
10:    B sends $r_{b,a}^{(k)}$ to A.
11:    A fits a model using $r_{b,a}^{(k)}$ as the label by solving $\hat{\theta}_{a,a}^{(k)} = \arg\min_{\theta_a \in \Theta_a} \mathbb{E}_n \ell(r_{b,a}^{(k)}, f_{a,\theta_a}(x_a))$.
12:    A calculates the residual $r_{a,a}^{(k+1)} \triangleq r_{b,a}^{(k)} - f_{a,\hat{\theta}_{a,a}^{(k)}}(x_a)$.
13: end for
14: Repeat the above Lines 5-12 with subscripts $a, b$ swapped.
15: B announces $\tau_b$ to A, and A announces $\tau_a$ to B at the same time.
16: After Stage I, let $k$ denote the number of rounds. A stores $\hat{\theta}_a$ (Line 1), $\hat{\theta}_{a,a}^{(\ell)}$ (Line 11), $\hat{\theta}_{a,b}^{(\ell)}$ (Lines 8&14), $\ell = 1, \ldots, k$, and similarly for B.
    Stage II: Prediction
    Initialization: Arrival of a future object whose modalities $x_a^f$ and $x_b^f$ are observed by A and B, respectively.
1: A calculates $\hat{y}_{a,a}^f \triangleq f_{a,\hat{\theta}_a}(x_a^f) + \sum_{\ell=1}^k f_{a,\hat{\theta}_{a,a}^{(\ell)}}(x_a^f)$ and $\hat{y}_{a,b}^f \triangleq f_{a,\hat{\theta}_a}(x_a^f) + \sum_{\ell=1}^k f_{a,\hat{\theta}_{a,b}^{(\ell)}}(x_a^f)$.
2: A queries the values of $\hat{y}_{b,a}^f \triangleq f_{b,\hat{\theta}_b}(x_b^f) + \sum_{\ell=1}^k f_{b,\hat{\theta}_{b,a}^{(\ell)}}(x_b^f)$ and $\hat{y}_{b,b}^f \triangleq f_{b,\hat{\theta}_b}(x_b^f) + \sum_{\ell=1}^k f_{b,\hat{\theta}_{b,b}^{(\ell)}}(x_b^f)$, which are calculated (privately) by B.
3: A calculates $\tilde{y}_a \triangleq \hat{y}_{a,a}^f + \hat{y}_{b,a}^f$ and $\tilde{y}_b \triangleq \hat{y}_{b,b}^f + \hat{y}_{a,b}^f$.
Output: A obtains the assisted prediction $\hat{y}_a^f \triangleq (1 - \tau_a \tau_b)^{-1} \{\tilde{y}_a - \tau_b \tilde{y}_b\}$, which follows from Equation (3). B operates similarly.
```

above factors into account is to encourage entities randomly generate τ_a and τ_b from $[-1, 0)$ and $(0, 1]$, respectively.

Remark 3 (What to blend)**.** In the proposed PAL protocol, B blends $\tau_b r_b$. In writing the paper, we also considered two alternative options. One option is to blend B's task label directly, namely $\tau_b y_b$. Using a similar argument as above, A and B will still target the blended label $\tilde{y}_a \triangleq y_a + \tau_b y_b$. However, since all the terms on the right-hand side of (3) will be transparent to A, it can decode B's labels during the prediction stage. This may not be appealing for B. Thus, we suggest using $\tau_b r_b$ instead of $\tau_b y_b$ as an 'extra-protection' of B's private task. The other option we considered is to continuously blend some statistics in each round. In the suggested protocol, B will inject $s_{b,a}$ only in the first round of training. In general, B can inject $s_{b,a}^{(k)}$ at round k. For example, $s_{b,a}^{(k)} = \tau_b^{(k)} r_b$, with the sequence $\{\tau_b^{(k)}\}_k$ satisfying $\sum_{k=1}^{\infty} \tau_b^{(k)} = \tau_b$. We omitted this extension for brevity. Lastly, we point out that if B does not have a task and simply lets $y_b = 0$, PAL reduces to the unilateral assistance mentioned in Subsection II-C.

Remark 4 (PAL as a privacy-preserving learning strategy)**.** In the default PAL training and prediction stages, entities have complete autonomy on model fitting. They do not need to share their local models and datasets, and they are bound to assist each other. As a result, PAL meets our general goal outlined in Subsection II-B. Nevertheless, the information exchange among entities inevitably leaks some data information. To quantify data privacy, one may further integrate a data privacy framework such as differential privacy [13] and interval privacy [14] into the learning approach. One may also enhance objective and model privacy by alleviating the risk of reconstructing a trained model through prediction interfaces [15], [16].

Remark 5 (Stop criteria)**.** We suggest two stop criteria for practical implementations. With the first stop criterion, the training is repeated K rounds until the fitting error no longer decreases much or K reaches a pre-specified limit. Since assisted learning is designed for organizational learners, it is appealing to have a small number of rounds due to negotiation costs in each round. From our experiments, the number to attain near-oracle performance is often within five.

The second stop criterion is to use the cross-validation technique [17]. In particular, A and B only use a portion of the data (e.g., the first 70% rows aligned by data IDs) to perform PAL. They preserve the remaining rows to evaluate out-sample performance as if in the prediction stage. The learning process continues until A's or B's validation error no longer decreases. In that case, the corresponding entity may terminate the training stage. To have a desirable generalizability, the validation data size cannot be too large. A comprehensive discussion on the splitting ratio can be found in [17]. We used 30% of the data for validation in the experiments.

The second stop criterion is appealing for scenarios where entities contribute unequally. For example, suppose that A is much more resourceful than B so that A can significantly assist B but not the other way around. Then, A could stop early so that B gets what it contributes. As a result, the PAL scheme applies not only to cooperative entities but also to competing entities. An implicit assumption of the second stop criterion is that entities need to exchange τ_a and τ_b before the training so that they can decode and validate at the end of each round. If an entity does not wish to share the underlying tau initially, an alternative option is to exchange only approximated tau at each round (see Remark 3).

Remark 6 (Non-cooperation)**.** Entities are incentivized to do better together during both the training and prediction stages. With cooperative training, each of them could potentially improve the single-entity performance during prediction, which will be shown in Figures 2-5. Otherwise, an entity will stop early, and all the entities' performance gains will be hindered, as shown in Table III. Also, suppose that a participating entity, say B, chooses to perturb its results sent to others in the prediction stage. This may compromise A's prediction performance compared with its baseline, e.g., A's single-entity learning and A's assistance received from an entity other than B. In that case, A may terminate the mutual assistance.

In summary, the PAL training can be terminated at the discretion of either A or B. It is good to ensure that A can stop appropriately if B is adversarial, B's model and data are not of high quality, or A's validation performance has reached a plateau, and vice versa. Similarly, in the prediction, they will not help each other if any participant is not satisfied with the prediction performance.

Remark 7 (Computation complexity)**.** In each training round of Algorithm 1, each entity learns a local model. The complexity, say c_n, depends on the specific model used and sample size. Additionally, it calculates the residual with complexity $O(n)$ with n the sample size. Thus, an entity's overall complexity is $O(nk + c_n k)$, where k is the number of rounds.

Remark 8 (Multiple entities)**.** In general, there will be m entities participating in m assisted training procedures, each initialized by a particular entity. Then, during the assisted prediction stage, each entity uses an appropriate linear combination of the aggregated predictions from m procedures to obtain the final prediction. For example, suppose that entity i injects $\tau_{i,j} y_i$ in the training initialized by j. Let $T = [\tau_{i,j}]_{1 \leq i,j \leq m}$, with $\tau_{i,i} = 1$. Suppose that T is invertible. Then, following a similar notation as in (3), the decoding is obtained by using $[y_1, \ldots, y_m]^{\mathrm{T}} = T^{-1}[\tilde{y}_1, \ldots, \tilde{y}_m]^{\mathrm{T}}$.

E. Theoretical justification of PAL

We provide theoretical analysis of PAL in this subsection. We show that PAL can converge to a solution that is equivalent to the training from centralized data. We can easily verify the assumptions for linear and generalized linear models. Though they are not mild for general supervised learning, we found from experimental studies that the same PAL methodology works well in a broader range of contexts. Apart from Assumption 1, we assume the following for technical analysis.

Assumption 2. The loss function ℓ satisfies $\ell(y, \tilde{y}) = s(y - \tilde{y})$, for a nonnegative function s with $s(0) = 0$.

Assumption 3. Each entity i's model $f_{i,\theta}$ is additive in the sense that $f_{i,\theta} + f_{i,\theta'} = f_{i,\theta+\theta'}$ for any $\theta, \theta' \in \Theta_i$.

Assumption 4. The function $\mathbb{E}_n \ell\big(\tilde{y}, \sum_{i \in I} f_{i,\theta_i}(x_i)\big)$ is continuously differentiable and strongly convex as a function of $[\theta_i]_{i \in I}$. Also, it is strongly convex in each θ_i, $i \in I$.

The above assumptions hold, for example, when we use the quadratic loss $s : \delta \mapsto \|\delta\|_2^2$ and regression models with linear coefficients θ, including nonlinear regression functions admitting an expansion on a linear basis, e.g., polynomial, wavelet, spline, or neural tangent kernels [18].

Recall that our goal in the ideal case is to solve (2) using centralized data. With the blending scheme used in the default PAL algorithm (Remarks 3&8), we define the following oracle benchmark of PAL if data were centralized. For the assisted learning procedure initialized by entity j, the targeted label is $\tilde{y}_j = y_j + \tau_i y_i$ $(i, j \in I \triangleq \{a, b\})$, and the oracle solution is given by the empirical risk minimization

$$\{\tilde{\theta}_{i,j}\}_{i \in I} = \underset{\{\theta_{i,j}\}_{i \in I}}{\arg\min}\, \mathbb{E}_n \ell\bigg(\tilde{y}_j, \sum_{i \in I} f_{i,\theta_{i,j}}(x_i)\bigg) \qquad (4)$$

for each $j \in I$. When there is no label injection, namely $\tau_i = 0$, the above problem reduces to (2). Ideally, for a future observation $(x_i^f : i \in I)$, the entity A would calculate

$$\tilde{y}_a^f = f_{a,\tilde{\theta}_{a,a}}(x_a^f) + f_{b,\tilde{\theta}_{b,a}}(x_b^f),$$

and similarly $\tilde{y}_b^f$, and then use Equation (3) to decode the predicted labels, say y_a^f and y_b^f. It is natural to *compare this centralized benchmark, namely y_a^f and y_b^f, with the estimates $\hat{y}_a^f$ and $\hat{y}_b^f$ from Algorithm 1.*

Theorem 1. Under Assumptions 1- 4, for a fixed future observation $(x_i^f, i \in I)$ and for each $j \in I$, $\ell(\hat{y}_j^f, y_j^f)$ converges R-linearly to zero as the number of rounds $k \to \infty$.

In the above analysis, we considered an oracle for a given set of data. Alternatively, we may define an oracle from a statistical perspective, treating data as randomly generated. We suppose that the data generating process is given by the postulated model (1) with unknown true parameters $\theta^*_{i,j}$. Namely, $\mathbb{E}(y_j \mid x_i, i \in I) = \sum_{i \in I} f_{i,\theta^*_{i,j}}(x_i)$. Under some conditions, we prove that the PAL produces a prediction result that is close to that provided by the underlying parameters $\theta^*_{i,j}$, defined by

$$y_j^f \triangleq \sum_{i \in I} f_{i,\theta^*_{i,j}}(x_i), \quad \forall j \in I. \qquad (5)$$

Assumption 5. For all $j \in I$,

$$\sup_{\theta_{i,j} \in \Theta_i, i \in I} \left| \mathbb{E}_n \ell\big(y_j, \sum_{i \in I} f_{i,\theta_{i,j}}(x_i)\big) - \mathbb{E}\,\ell\big(y_j, \sum_{i \in I} f_{i,\theta_{i,j}}(x_i)\big) \right|$$

goes to zero as $n \to \infty$.

Assumption 6. $\mathbb{E}\,\ell\big(y_j, \sum_{i \in I} f_{i,\theta_{i,j}}(x_i)\big)$ is uniquely minimized at its data-generating parameter $\{\theta^*_{i,j}\}_{i \in I}$, $\forall j \in I$.

The above Assumption 5 is a type of uniform law of large numbers, commonly assumed for large-sample analysis [19]. Assumption 6 says that the loss is 'proper' [17] so that the expected loss is minimized at the data-generating model. With more sophisticated assumptions on the rate of convergence in Assumption 5 (see, e.g., [20]), it is possible to derive the rate of convergence of $\hat{y}^f_j - y^f_j$, where y^f_j was defined in (5).

Theorem 2. Under Assumptions 1-6, for all $j \in I$, $\ell(\hat{y}^f_j, y^f_j) \to 0$ in probability as $k, n \to \infty$.

F. Classification learning

We consider the situation where each entity is performing classification learning. For technical simplicity, we consider binary classification. We propose to use the following solution. Let $\mathcal{Y} = \{0, 1\}$ represent the label space, and treat $y \in \mathcal{Y}$ as numerical values. We apply the same training procedure in Algorithm 1. In the prediction stage, the entity A outputs $\hat{\mathbb{P}}(y^f_a = 1) = \sigma(c\hat{y}^f_a)$, where $\sigma : \mathbb{R} \to [0, 1]$ is the sigmoid function and c is a coefficient trained to calibrate the fitted probabilities. In this way, we can directly apply Algorithm 1 even for classification tasks. Under some reasonable assumptions, we justify the use of regression procedure for classification tasks.

Proposition 1. Suppose that the classification data $(y, x) \in \mathcal{Y} \times \mathbb{R}^p$ follow a generative model in the form of $y = g(\theta_*^{\mathrm{T}} x, \varepsilon)$, where x follows an elliptical distribution with mean zero and covariance matrix V, ε is a noise term independent with x, and $\theta_* \in \mathbb{R}^p$ is an unknown parameter. Then, the least squares solution (corresponding to the ℓ_2 loss) converges in probability to θ_*/c for a fixed constant c as the sample size goes to infinity.

Examples of elliptical distribution include multivariate Gaussian distribution and Student's t-distribution. Examples of the model g include the logistic regression model and the probit model. For example, suppose that each entity j uses a logistic regression model. According to the previous argument, we only need to run PAL with linear functions $g_{i,\theta_{i,j}}$. Then, an entity j will be able to (virtually) obtain a predicted linear function $\hat{f}^f_j : x \mapsto \hat{\theta}^{\mathrm{T}} x = \sum_{i \in I} g_{i,\hat{\theta}_{i,j}}(x_i)$ through assisted prediction (Section II-E). Here, $\hat{\theta}_j = [\hat{\theta}_{i,j}]_{i \in I} \to_p \theta^*_j / c_j$ as $n, k \to \infty$, where θ^*_j denotes the data-generating parameter of entity j's logistic regression model. The entity j can then estimate c_j in the following way. We define the profile log-likelihood of c as

$$l_j(c) = \sum_{i=1}^{n} \log\left(\frac{y_i}{1 + \exp(-c\hat{y}_i)} + \frac{1 - y_i}{1 + \exp(c\hat{y}_i)} \right),$$

where y_i is the observed label and $\hat{y}_i$ is the regression-fitted value, and then obtain the scalar c that maximizes $l_j(c)$.

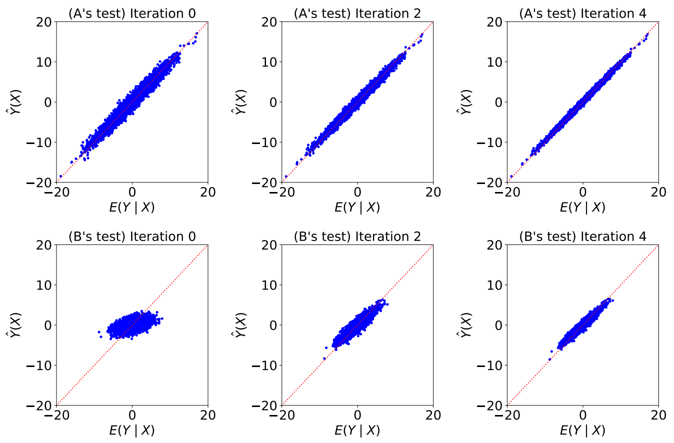

Fig. 2: A numerical experiment of Example 1, showing the predicted response against the expected response of entity A (top row) and entity B (bottom row) at iterations 0, 2, and 4 of PAL. Iteration 0 means single-entity learning.

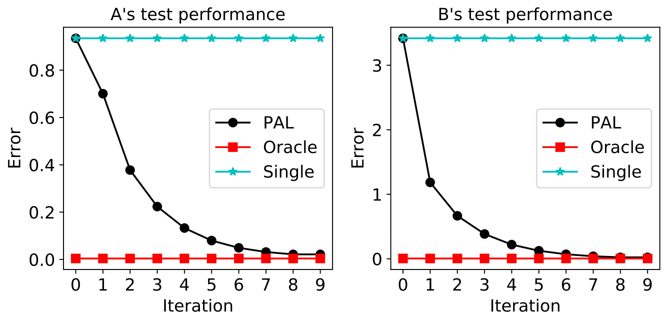

Fig. 3: The out-sample mean squared errors (tested from 1000 observations) corresponding to Figure 2 (Example 1).

III. Experimental Study

In this section, we provide experiments to demonstrate the performance of PAL under various circumstances. Each experiment was independently replicated ten times, and the standard errors are reported in parentheses. The computing infrastructure is a 3.4 GHz quad-core CPU.

A. Simulated data examples

Example 1 (regression). An entity A is equipped with data (y_a, x_a) and regression model f_a. Another entity B is equipped with data (y_b, x_b) and model f_b. Their data can be collated. It is desirable for A assisted by B to virtually learn a model as if trained on (y_a, x_a, x_b), and similarly for B. Figures 2 and 3 provide a numerical illustration that shows both entities can simultaneously enhance their prediction performance through PAL. In this experiment, the entities' data are 1000 observations of $[y_a, x_1, x_2, x_3]$ and $[y_b, x_4, x_5]$. We generated $x = [x_1, \dots, x_5]$ from zero-mean Gaussian with covariances $\mathrm{cov}(x_i, x_j) = 0.9^{|i-j|}$, task labels from $y_a = \beta_a^T x + \varepsilon_a$ with $\beta_a = [1, \dots, 1]$, $y_b = \beta_b^T x + \varepsilon_b$ with $\beta_b = [-2, -1, 0, 1, 2]$, and $\varepsilon_a, \varepsilon_b$ from standard Gaussian. The linear model was used for both entities' local training.

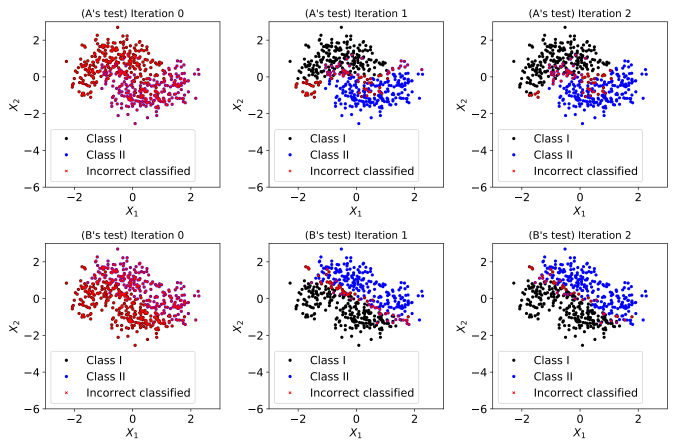

Fig. 4: A numerical experiment of Example 2, showing the two-class test data of entities A (top row) and B (bottom row), along with misclassified points (in red cross) at iterations 0, 1, and 2 of PAL.

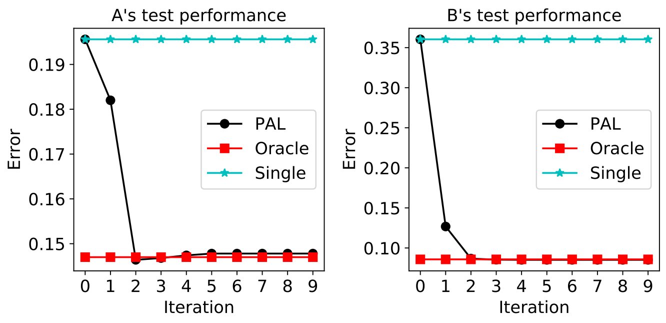

Fig. 5: The out-sample classification error (tested on 5000 observation) corresponding to Figure 4 (Example 2).

Example 2 (classification). The scenario is similar to Example 1, except that both entities are performing classification tasks. A numerical example is demonstrated in Figure 4 and 5. The data held by A and B are $[y_a, x_1]$ and $[y_b, x_2]$, respectively, where 1000 observations of $y_a, x \triangleq [x_1, x_2]$ were generated from the *sklearn* Moons dataset (with noise level 0.5), and y_b conditional on x follows from a logistic regression model with coefficients $[5, 5]$. The logistic regression model was used for both entities' local training.

B. Influence of different factors

In this experiment, we simulated two datasets. *Dataset 1* is from $y_a = x_1 + \cdots + x_5 + \varepsilon_a$, $y_b = -2x_1 - x_2 + x_4 + 2x_5 + \varepsilon_b$, where $x \in \mathbb{R}^5$ is zero-mean Gaussian with covariance $[0.9^{|i-j|}]_{i,j}$, and $\varepsilon_a, \varepsilon_b$ are standard Gaussian noise. *Dataset 2* is from $y_a = x_1 + \cdots + x_5 + \varepsilon_a$, $y_b = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon_b$ ('Friedman-1' dataset), where $x, \varepsilon_a, \varepsilon_b$ are similarly defined as above.
We supposed that the entity A held x_1, x_2, x_3 and B held x_4, x_5, and they used $\tau_a = 1, \tau_b = -1$. The PAL was run for a maximum number of ten rounds. The stop criterion based on local validation was used (see Remark 5), so the actual number of rounds could be smaller than ten.

TABLE II: (Subsection III-B) The performance of A's single-entity learning ($e_{a,\texttt{self}}$), PAL learning ($e_{a,\texttt{pal}}$), and oracle learning by centralizing data ($e_{a,\texttt{orac}}$) for different datasets and models, as evaluated by the root mean squared error (RMSE). Similar performances for B are reported.

	Data 1 (linear)		Data 2 (nonlinear)	
	LR-LR	RF-RF	LR-LR	RF-RF
$e_{a,\texttt{self}}$	1.94 (0.01)	2.06 (0.01)	1.42 (0.01)	1.48 (0.01)
$e_{a,\texttt{pal}}$	1.25 (0.01)	1.91 (0.01)	1.09 (0.01)	1.50 (0.01)
$e_{a,\texttt{orac}}$	1.08 (0.01)	1.61 (0.01)	1.08 (0.01)	1.27 (0.01)
$e_{b,\texttt{self}}$	2.86 (0.01)	2.94 (0.01)	4.67 (0.01)	4.79 (0.01)
$e_{b,\texttt{pal}}$	1.25 (0.01)	1.98 (0.01)	3.43 (0.01)	2.23 (0.02)
$e_{b,\texttt{orac}}$	1.07 (0.01)	1.71 (0.01)	3.43 (0.01)	2.15 (0.01)

TABLE III: (Subsection III-B) The PAL performance and the average number of assistance rounds (denoted by k) under various deviations.

	$e_{a,\texttt{pal}}$	$e_{b,\texttt{pal}}$	k
$q_a = 0, q_b = 0$	1.25 (0.01)	1.25 (0.01)	10.0 (0.0)
$q_a = 0.2, q_b = 0.2$	1.52 (0.01)	1.20 (0.01)	10.0 (0.0)
$q_a = 0.5, q_b = 0.2$	1.57 (0.04)	1.34 (0.16)	9.2 (0.8)
$q_a = 0.3, q_b = 0.3$	1.79 (0.04)	1.69 (0.25)	7.6 (1.2)
$q_a = 0.5, q_b = 0.5$	1.95 (0.0)	2.86 (0.01)	0.0 (0.0)

We considered four model choices of A and B in the second row of Table II, where 'LR' means linear model, and 'RF' means random forest with 50 trees and a max tree depth of five. From the results in Table II, PAL can significantly improve performance compared with single-entity learning. To study the influence of wrongly-transmitted τ_a, τ_b, we did the following ablation study. We took the linear-model case (namely 'LR-LR') as an example. Suppose that A and B send wrong tau-values to each other, with $\tau_a' = (1 - q_a)\tau_a$, $\tau_b' = (1 - q_b)\tau_b$. Here, q_a, q_b describes the level of deviations. The performance results under various (q_a, q_b) are summarized in Table III. The results show that more deviations will lead to worse cooperative performance and an earlier stop of assisted training.

C. Dependence on task difficulty

In this experiment, we studied the dependence of PAL performance on task difficulty. We considered a similar classification setting as in Example 2. The entities A and B each held one feature and used the PAL in Subsection II-F. In A's dataset 1, we generated A's task labels from the logistic regression model with function $5x_1 - 5x_2$. In A's dataset 2, A's labels were generated from the Moon dataset, which has a nonlinear decision boundary and does not fall into the generalized linear model in Proposition 1. We considered two different tasks for B. The 'easy' one corresponds to the label generated from the logistic regression function $5(x_1 + x_2)$, and the 'difficult' one is from the function $x_1 + x_2$ (since the signal-to-noise ratio is smaller). The results summarized in Table IV show that A's PAL performance is not sensitive to B's task difficulty, despite A's task. Also, the extent to which A could assist B highly depends on B's task difficulty.

TABLE IV: (Subsection III-C) The performance of A's single-entity learning ($\texttt{acc}_a$), PAL ($\texttt{acc}_{a,\texttt{pal}}$), and oracle ($\texttt{acc}_{a,\texttt{pal}}$), as evaluated by the classification accuracy (in percentage). Similar quantities are defined for the entity B.

	A's data 1		A's data 2	
B's task	Easy	Difficult	Easy	Difficult
$\texttt{acc}_a$	74.1 (0.2)		74.5 (0.2)	
$\texttt{acc}_{a,\texttt{pal}}$	92.2 (0.2)	90.0 (1.8)	80.9 (0.1)	79.4 (0.5)
$\texttt{acc}_{a,\texttt{orac}}$	92.3 (0.2)	92.3 (0.1)	80.8 (0.1)	80.7 (0.2)
$\texttt{acc}_b$	74.0 (0.2)	65.3 (0.1)	67.6 (0.2)	60.7 (0.3)
$\texttt{acc}_{b,\texttt{pal}}$	92.1 (0.1)	71.4 (0.7)	91.3 (0.1)	69.4 (0.4)
$\texttt{acc}_{b,\texttt{orac}}$	92.2 (0.1)	72.4 (0.2)	91.4 (0.1)	70.4 (0.1)

TABLE V: The performance of entity A's single-entity learning ($e_{a,\texttt{self}}$), A's PAL with $\tau_a = 1, \tau_b = -1$ ($e_{a,\texttt{pal}_1}$), and A's PAL under $\tau_a = 0.1, \tau_b = -0.1$ ($e_{a,\texttt{pal}_2}$), under various numbers features p_a held by A. Similar results are reported for entity B.

p_a	2	8	14
$e_{a,\texttt{self}}$	15.08 (0.05)	10.2 (0.05)	10.26 (0.05)
$e_{a,\texttt{pal}_1}$	12.83 (0.58)	9.49 (0.07)	9.45 (0.07)
$e_{a,\texttt{pal}_2}$	11.88 (0.44)	9.41 (0.06)	9.46 (0.07)
$e_{a,\texttt{orac}}$		9.03 (0.04)	
$e_{b,\texttt{self}}$	185.34 (1.6)	186.06 (1.65)	197.74 (1.82)
$e_{b,\texttt{pal}_1}$	184.05 (1.89)	175.7 (1.58)	177.42 (2.1)
$e_{b,\texttt{pal}_2}$	180.03 (1.59)	175.9 (1.46)	179.81 (1.76)
$e_{b,\texttt{orac}}$		171.68 (1.72)	

D. Real-world case study: medical benchmark

There exists a large market of PAL in, e.g., the internet-of-things [21], autonomous mobility [22], industrial control [23], unmanned aerial vehicles [24], and biological monitoring [25]. We will take the medical industry as an example to illustrate such learning scenarios. It is common for medical organizations to acquire others' assistance to improve clinical care [26], reduce capital costs [27], and accelerate scientific progress [28].

We considered the *MIMIC3* clinical database [29], which consists of several divisions that collected various features from the same cohort of patients. Suppose that the Intensive Care Unit (A) aims to predict the systolic blood pressure, and the Laboratory (B) will predict the length of stay. We supposed that A holds the first p_a variables, while B holds the remaining variables. Such an experimental design was intended to show that the PAL performance gain depends on the relative information offered by the assisting entity. Both entities used the random forest model with 50 trees and a max tree depth of five. We pre-processed MIMIC3 in a way similar to [30], [31], and obtained a dataset of 6916 patients and 16 variables that exclude the two task variables of A and B. In particular, the ordered variables are named capillary refill rate, diastolic blood pressure, fraction inspired oxygen, heart rate, height, mean blood pressure, oxygen saturation, respiratory rate, temperature, weight, pH, glucose, Glasgow coma scale total, Glasgow coma scale, eye Glasgow coma scale motor, and Glasgow coma scale verbal. We used 35% training, 15% validation (Remark 5), and 50% testing.

The results summarized in Table V show that PAL can significantly improve an entity's local learning. Also, such an improvement could depend on the number of features offered by the other entity. For example, A can help B more than the other way around if A gains more variables. We also experimented with two different pairs of τ_a, τ_b. The PAL performance is insensitive to the choice of tau values.

E. Real-world case study: census benchmark

In this subsection, we will use the *Census* dataset [32] to demonstrate a *mixed-type* PAL, where A and B perform classification and regression tasks, respectively. The data contain anonymous information, including income, age, education, occupation, capital gain or loss, working class, marital status, occupation, relationship, race, gender, native country, and working hours per week. Suppose that A's goal is to train a binary classifier to predict whether an individual's income is higher than \$50k or not, while B's goal is to train a regression to predict the working hours per week. Suppose that A holds the attributes of age and education, while B holds the remaining attributes. We run the PAL for A and B. Since A's task is classification, it also ran a calibration step as discussed in Subsection II-F. In running this experiment, we injected noise into the transmitted residual so that it satisfies different levels of differential privacy [33]. In particular, we truncated the transmitted residual at each round within a percentile and added the residual with a suitable amount of noise. The results, summarized in Table VI, show that the introduction of reasonable privacy guarantees does not significantly degrade the performance.

More implementation details are described below. The original data have been split into 32561 instances for training and 16281 for testing, randomly replicated ten times. Next, we briefly introduce how we implemented differential privacy. Recall that a randomized mechanism $\mathcal{M} : \mathcal{D} \mapsto \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies ε-differential privacy if for any two adjacent inputs $\mathcal{D}_1, \mathcal{D}_2$ and for any subset of outputs $S \subseteq \mathcal{R}$, it holds that $\mathbb{P}(\mathcal{M}(\mathcal{D}_1) \in S) \leq e^{\varepsilon}\mathbb{P}(\mathcal{M}(\mathcal{D}_2) \in S)$ [13]. Here, the adjacency means that $\mathcal{D}_1, \mathcal{D}_2$ only differ in one row. Briefly speaking, differential privacy measures privacy leakage by a privacy budget ε that bounds the likelihood ratio of the random output under two databases differing in a single individual. The smaller ε is, the more private for any individual. Intuitively, a small ε means that privacy is better protected. A standard randomized mechanism is the following. Suppose that a query function $f : \mathcal{D} \to \mathbb{R}^d$ satisfies $\|f(\mathcal{D}_1) - f(\mathcal{D}_2)\|_1 \leq c$ for any adjacent $\mathcal{D}_1, \mathcal{D}_2$ and some constant $c > 0$. Here, $\|\cdot\|_1$ is the standard ℓ_1 norm of vectors. Let $\texttt{Lap}(\lambda)$ denote the Laplace distribution with density function $p_\lambda(x) = (2\lambda)^{-1}\exp(-|x|/\lambda)$. Then, $\mathcal{M}(\mathcal{D}) = f(\mathcal{D}) + \zeta$, with ζ following i.i.d. $\texttt{Lap}(c/\varepsilon)$ (Laplacian noise), is a mechanism that satisfies ε-differential privacy [13]. In our experiment, at each round of transmission, we truncated the residual by their %10 and %90 quantiles. Let c denote that corresponding interval width. We added i.i.d. $\texttt{Lap}(c/\varepsilon)$ noise to each element.

IV. Conclusion

The interactions between learning entities in privacy-aware scenarios pose new challenges that cannot be well addressed

TABLE VI: (Subsection III-E) The performance of A's classification learning (evaluated by accuracy in percentage) and B's regression learning (evaluated by RMSE), under various DP levels (indexed by the privacy budget ε).

	A's $\texttt{acc}_a$	B's e_b	k
self	79.5 (0.1)	0.92 (0.01)	NA
pal,DP$_{\varepsilon=1}$	80.7 (0.2)	0.89 (0.01)	3.0 (0.1)
pal,DP$_{\varepsilon=5}$	83.3 (0.3)	0.88 (0.01)	6.4 (0.3)
pal,DP$_{\varepsilon=10}$	84.9 (0.1)	0.87 (0.01)	10.0 (0.1)
orac	85.0 (0.1)	0.87 (0.01)	NA

by classical machine learning. In this work, we proposed the notion of Parallel Assisted Learning, where the general goal is to expand single-entity learning capabilities with assistance from multiple entities but without sharing private modeling algorithms or data. Entities are allowed to hold heterogeneous tasks, such as regression on different response tasks, classification on various label tasks, and a mixture of regression and classification tasks. Theoretical analysis from both optimization and statistical perspectives is provided to justify the developed methods. An interesting future problem is to study PAL for unsupervised learning tasks.

APPENDIX A
PROOF OF THEOREM 1

Proof. The main idea of the proof is to connect the PAL procedure with the block coordinate descent algorithm. We still consider the two-entity problem for notational simplicity.

Recall that the notation $\hat{\theta}_{i,j}^{(k)}$ in Algorithm 1 denotes entity i's estimated parameter at round k of the learning initialized by entity j, for $k = 1, 2, \ldots$. For notational convenience, we define

$$\hat{\theta}_{a,a}^{(0)} = \hat{\theta}_a, \quad \hat{\theta}_{b,a}^{(0)} = \tau_b \hat{\theta}_b, \quad \hat{\theta}_{a,b}^{(0)} = \tau_a \hat{\theta}_a, \quad \hat{\theta}_{b,b}^{(0)} = \hat{\theta}_b,$$

with $\hat{\theta}_a$ and $\hat{\theta}_b$ defined in Algorithm 1 (lines 1-2).

To prove the theorem, simple calculations using the definition of y_a^f, y_b^f and the lines 15-16 of Algorithm 1 show that we only need to prove that $\sum_{\ell=0}^k \hat{\theta}_{i,j}^{(\ell)}$ converges R-linearly to $\tilde{\theta}_{i,j}$ as $k \to \infty$, $\forall i, j \in I$.

Recall the following oracle benchmark if A had all the data from B.

$$\{\tilde{\theta}_{i,a}\}_{i\in I} = \underset{\{\theta_{i,a}\}_{i\in I}}{\arg\min}\, \mathbb{E}_n \ell\biggl(\tilde{y}_a, \sum_{i\in I} f_{i,\theta_{i,a}}(x_i)\biggr), \qquad (6)$$

with $\tilde{y}_a \triangleq y_a + \tau_b y_b$, $I = \{a, b\}$, and similarly for $\{\tilde{\theta}_{i,b}\}_{i\in I}$.

Suppose that A runs the following block-wise descent algorithm in hindsight. At the kth round, $k = 1, 2, \ldots$, the entity A iteratively solves

$$\hat{\beta}_{b,a}^{(k)} = \underset{\beta_{b,a}\in\Theta_b}{\arg\min}\, \mathbb{E}_n \ell\Bigl(\tilde{y}_a, f_{a,\hat{\beta}_{a,a}^{(k-1)}}(x_a) + f_{b,\beta_{b,a}}(x_b)\Bigr), \qquad (7)$$

$$\hat{\beta}_{a,a}^{(k)} = \underset{\beta_{a,a}\in\Theta_a}{\arg\min}\, \mathbb{E}_n \ell\Bigl(\tilde{y}_a, f_{a,\beta_{a,a}}(x_a) + f_{b,\hat{\beta}_{b,a}^{(k)}}(x_b)\Bigr). \qquad (8)$$

Under Assumptions 2 and 3, we can establish a one-to-one mapping between the above procedure and Algorithm 1, by letting (for $k = 1, 2, \ldots$)

$$\beta_{b,a} = \biggl(\sum_{\ell=0}^{k-1} \hat{\theta}_{b,a}^{(\ell)}\biggr) + \theta_{b,a}, \quad \hat{\beta}_{b,a}^{(k)} = \sum_{\ell=0}^{k} \hat{\theta}_{b,a}^{(\ell)},$$

$$\beta_{a,a} = \biggl(\sum_{\ell=0}^{k-1} \hat{\theta}_{a,a}^{(\ell)}\biggr) + \theta_{a,a}, \quad \hat{\beta}_{a,a}^{(k)} = \sum_{\ell=0}^{k} \hat{\theta}_{a,a}^{(\ell)}.$$

Then, the PAL and the block-wise descent algorithms are operationally equivalent. Note that at each round of training, an entity only needs to fit the most recently received residual without accessing others' data. According to Assumption 4 and [34, Proposition 3.4], we conclude that $\hat{\beta}_{i,a} = \sum_{\ell=0}^k \hat{\theta}_{i,a}^{(\ell)}$ converges R-linearly to $\tilde{\theta}_{i,a}$ as $k \to \infty$, $\forall i \in I = \{a, b\}$. Similar results hold with the subscript a replaced with b.

□

APPENDIX B
PROOF OF THEOREM 2

Proof. First, we prove that if $(y_i, x_i : i \in I)$ are generated from the model

$$\mathbb{E}(y_j \mid x_i, i \in I) = \sum_{i\in I} f_{i,\theta_{i,j}^*}(x_i) \qquad (9)$$

with underlying parameters $\theta_j^* = [\theta_{i,j}^*]_{i\in I}$, the estimated parameter

$$\hat{\theta}_j = [\hat{\theta}_{i,j}]_{i\in I} \triangleq \underset{[\theta_{i,j}]_{i\in I}}{\arg\min}\, \mathbb{E}\, \ell\biggl(y_j, \sum_{i\in I} f_{i,\theta_{i,j}}(x_i)\biggr)$$

converges in probability to θ_j^*. For notational convenience, let

$$h(\theta_j) = \ell\biggl(y_j, \sum_{i\in I} f_{i,\theta_{i,j}}(x_i)\biggr),$$

where $\theta_j = [\theta_{i,j}]_{i\in I}$. For an arbitrary $\varepsilon > 0$, we will prove that

$$\mathbb{P}(\|\hat{\theta}_j - \theta_j^*\|_2 \geq \varepsilon) \to 0 \qquad (10)$$

as $n \to \infty$.

By the definition of $\hat{\theta}_j$ we have

$$\mathbb{E}_n h(\hat{\theta}_j) \leq \mathbb{E}_n h(\theta_j^*) = \mathbb{E} h(\theta_j^*) + o_p(1)$$

where the last equality is implied by Assumption 5. Let $\Theta = \Theta_a \times \Theta_b$. Thus, we have

$$\begin{aligned}\mathbb{E}h(\hat{\theta}_j) - \mathbb{E}h(\theta_j^*) &\leq \mathbb{E}h(\hat{\theta}_j) - \mathbb{E}_n h(\hat{\theta}_j) + o_p(1) \\ &\leq \sup_{\theta_j\in\Theta} \Bigl| \mathbb{E}_n h(\theta_j) - \mathbb{E}h(\theta_j) \Bigr| + o_p(1), \quad (11)\end{aligned}$$

which converges to zero in probability as $n \to \infty$.

Assumptions 4&5 imply that

$$\inf_{\theta_j\in\Theta : \|\theta_j - \theta_j^*\|_2 \geq \varepsilon} \mathbb{E}h(\theta_j) > \mathbb{E}h(\theta_j^*). \qquad (12)$$

Inequality (12) ensures that there exists $\delta > 0$ such that $\mathbb{E}h(\theta_j) \geq \mathbb{E}h(\theta_j^*) + \eta$ for all θ_j satisfying $\|\theta_j - \theta_j^*\|_2 \geq \varepsilon$. Therefore,

$$\mathbb{P}(\|\hat{\theta}_j - \theta_j^*\|_2 \geq \varepsilon) \leq \mathbb{P}(\mathbb{E}h(\hat{\theta}_j) \geq \mathbb{E}h(\theta_j^*) + \eta)$$

which, according to (11), further goes to zero as $n \to \infty$. This concludes the proof of (10).

Second, we will prove $\ell(\hat{y}_j^f, y_j^f) \to 0$ in probability as $k, n \to \infty$. According to Assumption 3, for the assisted training initialized by entity j, the data-generating model behind $(\tilde{y}_j, x_i : i \in I)$ is in the form of (9) with parameter

$$\tilde{\theta}_{i,j}^* = \theta_{i,j}^* + \sum_{\ell \in I, \ell \neq j} \tau_\ell \theta_{i,\ell}^*.$$

The proof of the first part implies that the $\|\tilde{\theta}_{i,j} - \tilde{\theta}_{i,j}^*\|_2 \to 0$ in probability as $n \to \infty$, where $\tilde{\theta}_{i,j}$ was defined in (4) and also used in the proof of Theorem 1. Applying Theorem 1 and Assumption 2, we conclude that $\ell(\hat{y}_j^f, y_j^f) \to 0$ in probability as $k, n \to \infty$. □

Appendix C
Proof of Proposition 1

Proof. It was shown in [35] that θ_* is in the same space as $V^{-1}\mathbb{E}(x \mid y)$. Since θ_* is a vector that lines in a one-dimensional subspace of $\mathbb{R}^p$, if we consider $y = 1$,

$$\theta_* = \lambda V^{-1}\mathbb{E}(x \mid y = 1) \tag{13}$$

for some unknown constant λ. Additionally, since

$$\begin{aligned}\mathbb{E}(xy) &= \mathbb{P}(y=1)\mathbb{E}(xy \mid y=1) + \mathbb{P}(y=0)\mathbb{E}(xy \mid y=0) \\ &= \mathbb{P}(y=1)\mathbb{E}(x \mid y=1),\end{aligned}$$

Equation (13) implies that

$$\theta_* = \frac{\lambda}{\mathbb{P}(y=1)} V^{-1}\mathbb{E}(xy). \tag{14}$$

Note that the ordinal least squares applied to (y, x) gives

$$\hat{\theta}^{(\mathtt{ls})} \triangleq (\mathbb{E}_n xx^{\mathrm{T}})^{-1}\mathbb{E}_n(xy).$$

Thus, $\hat{\theta}^{(\mathtt{ls})}$ converge in probability to θ_*/c for a fixed constant c under the law of large numbers, as the sample size goes to infinity. It is worth noting that the above arguments imply that $c = \lambda/\mathbb{P}(y=1)$.

□

Acknowledgement

Jiawei Zhang and Jie Ding were supported in part by the National Science Foundation under grant number DMS-2134148. The authors thank three anonymous reviewers and Associate Editor Laurent Jacques for their constructive comments.

References

[1] Y. Censor, "Pareto optimality in multiobjective problems," *Appl. Math. Optim.*, vol. 4, no. 1, pp. 41–59, 1977.
[2] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.
[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.
[4] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *Prof. ICLR*, 2021.
[5] ——, "SemiFL: Communication efficient semi-supervised federated learning with unlabeled clients," *Proc. NeurIPS*, 2022.
[6] J. Ding, E. Tramel, A. K. Sahu, S. Wu, S. Avestimehr, and T. Zhang, "Federated learning challenges and opportunities: An outlook," in *Proc. ICASSP*. IEEE, 2022, pp. 8752–8756.
[7] Y. Yang, "Regression with multiple candidate models: selecting or mixing?" *Stat. Sin.*, pp. 783–809, 2003.
[8] J. Ding, J. Zhou, and V. Tarokh, "Asymptotically optimal prediction for time-varying data generating processes," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 3034–3067, 2019.
[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *in Proc. TIST*, vol. 10, no. 2, p. 12, 2019.
[10] Y. Liu, Y. Kang, L. Li, X. Zhang, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient vertical federated learning framework," *Scanning Electron Microsc Meet at*, 2019.
[11] Y. Liu, X. Zhang, Y. Kang, L. Li, T. Chen, M. Hong, and Q. Yang, "Fedbcd: A communication-efficient collaborative learning framework for distributed features," *IEEE Trans. Signal Process.*, 2022.
[12] X. Xian, X. Wang, J. Ding, and R. Ghanadan, "Assisted learning: A framework for multi-organization learning," *Proc. NeurIPS*, 2020.
[13] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Proc. EUROCRYPT*. Springer, 2006, pp. 486–503.
[14] J. Ding and B. Ding, "Interval privacy: A privacy-preserving framework for data collection," *IEEE Trans. Signal Process.*, 2022.
[15] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. USENIX Security*, 2016, pp. 601–618.
[16] X. Wang, Y. Xiang, J. Gao, and J. Ding, "Information laundering for model privacy," *Proc. ICLR*, 2021.
[17] J. Ding, V. Tarokh, and Y. Yang, "Model selection techniques–an overview," *IEEE Signal Process. Mag.*, vol. 35, no. 6, pp. 16–34, 2018.
[18] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Proc. NeurIPS*, vol. 31, 2018.
[19] T. S. Ferguson, *A course in large sample theory*. Routledge, 2017.
[20] J. Ding, E. Diao, J. Zhou, and V. Tarokh, "On statistical efficiency in learning," *IEEE Trans. Inf. Theory*, vol. 67, no. 4, pp. 2488–2506, 2021.
[21] E. Manavalan and K. Jayakrishna, "A review of internet of things (iot) embedded sustainable supply chain for industry 4.0 requirements," *Comput. Ind. Eng.*, vol. 127, pp. 925–953, 2019.
[22] F. Dandl, M. Hyland, K. Bogenberger, and H. S. Mahmassani, "Evaluating the impact of spatio-temporal demand forecast aggregation on the operational performance of shared autonomous mobility fleets," *Transportation*, vol. 46, no. 6, pp. 1975–1996, 2019.
[23] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6418–6428, 2014.
[24] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications," *IEEE Access*, vol. 7, pp. 105 100–105 115, 2019.
[25] X. Tang, J. Zhang, Y. He, X. Zhang, Z. Lin, S. Partarrieu, E. B. Hanna, Z. Ren, Y. Yang, X. Wang, N. Li, J. Ding, and J. Liu, "Multi-task learning for single-cell multi-modality biology," *bioRxiv*, 2022.
[26] J. T. Farrar, A. B. Troxel, K. Haynes, I. Gilron, R. D. Kerns, N. P. Katz, B. A. Rappaport, M. C. Rowbotham, A. M. Tierney, D. C. Turk *et al.*, "Effect of variability in the 7-day baseline pain diary on the assay sensitivity of neuropathic pain randomized clinical trials: an acttion study," *Pain®*, vol. 155, no. 8, pp. 1622–1631, 2014.
[27] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, 2013.
[28] B. Lo, "Sharing clinical trial data: maximizing benefits, minimizing risk," *Jama*, vol. 313, no. 8, pp. 793–794, 2015.
[29] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Sci. Data*, vol. 3, p. 160035, 2016.
[30] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *Scientific data*, vol. 6, no. 1, pp. 1–18, 2019.
[31] S. Purushotham, C. Meng, Z. Che, and Y. Liu, "Benchmarking deep learning models on large healthcare datasets," *J. Biomed. Inform*, vol. 83, pp. 112–134, 2018.
[32] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *Proc. KDD*, vol. 96, 1996, pp. 202–207.
[33] C. Dwork and K. Nissim, "Privacy-preserving datamining on vertically partitioned databases," in *Proc. CRYPTO*. Springer, 2004, pp. 528–544.
[34] Z.-Q. Luo and P. Tseng, "Error bounds and convergence analysis of feasible descent methods: a general approach," *Ann. Oper. Res.*, vol. 46, no. 1, pp. 157–178, 1993.
[35] K.-C. Li, "Sliced inverse regression for dimension reduction," *J. Am. Stat. Assoc.*, vol. 86, no. 414, pp. 316–327, 1991.