GAL: Gradient Assisted Learning for Decentralized Multi-Organization Collaborations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Collaborations among multiple organizations, such as financial institutions, medical centers, and retail markets in decentralized settings are crucial to providing improved service and performance. However, the underlying organizations may have little interest in sharing their local data, models, and objective functions. These requirements have created new challenges for multi-organization collaboration. In this work, we propose Gradient Assisted Learning (GAL), a new method for multiple organizations to assist each other in supervised learning tasks without sharing local data, models, and objective functions. In this framework, all participants collaboratively optimize the aggregation of local loss functions, and each participant autonomously builds its own model by iteratively fitting the gradients of the overarching objective function. We also provide asymptotic convergence analysis and practical case studies of GAL. Experimental studies demonstrate that GAL can achieve performance close to centralized learning when all data, models, and objective functions are fully disclosed.

1 Introduction

One of the main challenges in harnessing the power of big data is the fusion of knowledge from numerous decentralized organizations that may have proprietary data, models, and objective functions. Due to various ethical and regulatory constraints, it may not be feasible for decentralized organizations to centralize their data and fully collaborate to learn a shared model. Thus, a large-scale autonomous decentralized learning method that can avoid data, models, and objective functions transparency may be of critical interest.

Cooperative learning may have various scientific and business applications [1]. As illustrated in Figure 1, a medical institute may be helped by multiple clinical laboratories and pharmaceutical entities to improve clinical treatment and facilitate scientific research [2, 3]. Financial organizations may collaborate with universities and insurance companies to predict loan default rates [4]. The organizations can match the correspondence with common identifiers such as user identification associated with the registration of different online platforms, timestamps associated with different clinics and health providers, and geo-locations associated with map-related traffic and agricultural data. With the help of our framework, they can form a community of shared interest to provide better Machine-Learning-as-a-Service (MLaaS) [5, 6] without transmitting their local data, models, and objective functions.

The main idea of Gradient Assisted Learning (GAL) is outlined below. In the training stage, the organization to be assisted, denoted by Alice, will calculate a set of 'residuals' and broadcast these to other organizations. These residuals approximate the fastest direction of reducing the training loss in hindsight. Subsequently, other organizations will fit the residuals using their local data, models, and objective functions and send the fitted values back to Alice. Alice will then assign weights to each organization to best approximate the fastest direction of learning. Next, Alice will

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

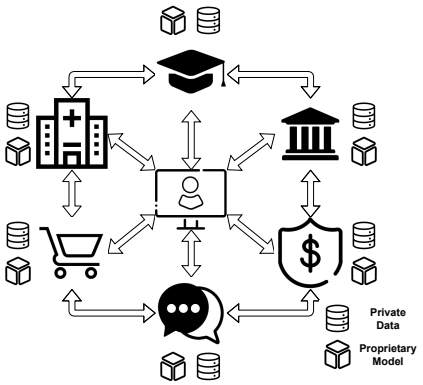


Figure 1: Decentralized organizations form a community of shared interest to provide better Machine-Learning-as-a-Service.

line-search for the optimal gradient assisted learning rate along the calculated direction of learning. The above procedure is repeated until Alice accomplishes sufficient learning. In the inference stage, other organizations will send their locally predicted values to Alice, who will then assemble them to generate the final prediction. We show that the number of assistance rounds needed to attain the centralized performance is often small (e.g., fewer than ten). This is appealing since GAL is primarily developed for large organizations with rich computation resources. A small number of interactions will reduce the communications and networking costs. Our main contributions are summarized below.

- We propose a Gradient Assisted Learning (GAL) algorithm that is suitable for large-scale autonomous decentralized learning. It can effectively exploit task-relevant information preserved by vertically decentralized organizations. Our method enables simultaneous collaboration among organizations without sharing data, models, and objective functions.
- We provide asymptotic convergence analysis and practical case studies of GAL. For the case of vertically distributed data, GAL generalizes the classical Gradient Boosting algorithm.
- Our proposed method can significantly outperform learning baselines and achieve near-oracle performance on various benchmark datasets. Compared with existing works, GAL does not need frequent synchronization of organizations. It also significantly reduces the computation and communication overhead.

2 Related work

Multimodal Data Fusion Vertically distributed data can be viewed as multimodal data with modalities provided in a distributed manner to different learners/organizations. Standard multimodal data fusion methods include the early, intermediate, and late data fusions [7, 8]. These methods concatenate different modes of data at the input, intermediate representation, and final prediction levels. However, these data fusion methods in decentralized settings often require organizations to share the task labels to train their local models synchronously. In contrast, our method presented below only requires that organizations asynchronously fit some task-related statistics named pseudo-residuals to approximate the direction of reducing the global training loss in hindsight.

Gradient Boosting Our approach was inspired by Gradient Boosting [9, 10], where weak learners are sequentially trained from the same dataset and aggregated into a strong learner. In our learning context, each organization uses side-information from heterogeneous data sources to improve a particular learner's performance. Our method can be regarded as a generalization of Gradient Boosting to address decentralized learning with vertically distributed data.

Federated Learning Federated learning [11, 12, 13, 14] is a popular distributed learning framework. Its main idea is to learn a joint model by averaging locally learned model parameters. It avoids the need for the transmission of local training data. Conceptually, the goal of Federated Learning is to exploit the resources of edge devices with communication efficiency. Vertical Federated Learning methods split sub-networks for local clients to jointly optimize a global model [15, 16, 17, 18, 19, 20]. These methods can be viewed as federated learning with an intermediate data fusion method, and

the central server will have access to the true labels. In order to converge, these methods typically require batchwise synchronization of backward gradients and a significant number of communication rounds [20]. Such batchwise synchronization is critical as each client contributes to part of the globally backpropable model, and a client's local update may not decrease the overall loss. In contrast, our proposed method trains multiple autonomous local models with pseudo-residuals, each contributing to a small portion of the overarching loss. Each round of updates will decrease the loss, and it does not require batchwise synchronizations. Consequently, our method can achieve desirable performance with significantly fewer communication rounds without a global backpropable model.

Assisted Learning Assisted Learning (AL) [21] is a decentralized collaborative learning framework for organizations to improve their learning quality. In that context, neither the organization being assisted nor the assisting organizations share their local models and data. Prior work on AL is limited to mean squared loss regression with only two organizations using a sequential exchange of information. Inspired by Gradient Boosting, our proposed Gradient Assisted Learning (GAL) is a general method for multiple organizations to assist each other in supervised learning scenarios.

Compared with existing works, our novelties include 1) generalization from squared loss to any differentiable loss for supervised learning, 2) allowing for local objective functions at each organization, 3) generalization from a sequential protocol between two organizations to parallel aggregation across multiple organizations, 4) introduction of deep model sharing for saving computation space, 5) introduction of the assisted learning rate for fast convergence, and 6) theoretical analysis of asymptotic convergence of GAL based on the classical Gradient Boosting algorithm.

3 Gradient Assisted Learning

3.1 Notation

Suppose that there are N data observations independently drawn from a joint distribution $p_{xy} = p_x p_{y|x}$, where $y \in \mathcal{Y}$ and $x \in \mathbb{R}^d$ respectively represent the task label and feature variables, and d is the number of features. For regression tasks, we have $\mathcal{Y} = \mathbb{R}$. For K-class classification tasks, $\mathcal{Y} = \{e_1, \ldots, e_K\}$, where e_k is the canonical vector representing the class k, $k = 1, \ldots, K$. Let $\mathbb{E}$ and $\mathbb{E}_N$ denote the expectation and empirical expectation, respectively. Thus, $\mathbb{E}_N g(y, x) \triangleq N^{-1} \sum_{i=1}^{N} g(y_i, x_i)$ for any measurable function g, where (y_i, x_i) are i.i.d. observations from p_{xy}. Suppose that there are M organizations. Each organization m only holds X_m, a sub-vector of X (illustrated in Figure 2). In general, we assume that the variables in $X_1, \ldots, X_M$ are disjoint in the presentation of our algorithm, although our method also allows for the sharing of some variables. For example, one organization may observe demographic features for a mobile user cohort, and another organization holds health-related features of that cohort. Without loss of generality, we suppose that Alice, the organization to be assisted, has local data x_1 and task label y_1, while other $M - 1$ organizations are collaborators which assist Alice and have local data $x_2 \ldots x_M$.

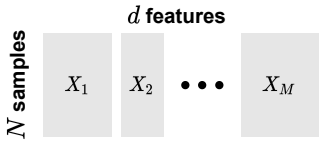


Figure 2: Illustration of organizations' vertically distributed data.

3.2 Problem

For $m = 1, \ldots, M$, let $\{x_{i,m}\}_{i=1}^N$ denote the available data to the organization m. Thus, N objects are simultaneously observed by M organizations, each observing a subset of features from the $x \in \mathbb{R}^d$. Alice also has local task labels $\{y_{i,1}\}_{i=1}^N$ for training purposes.

Let $\mathcal{F}_m$ and L_m respectively denote supervised function class (such as generalized linear functions or neural networks) and the local objective function of organization m. We will assume that L_m is differentiable. Without loss of generality, we assume that Alice denotes organization 1, who will be assisted. Without assistance from other organizations, Alice would learn a model that minimizes the following empirical risk,

$$F_{\text{Alone}} = \operatorname*{argmin}_{F_1 \in \mathcal{F}_1} \mathbb{E}_N L_1(y_1, F_1(x_1)). \tag{1}$$

Note that the above formulation only involves Alice's local data x_1 and local model (as represented by F_1 and L_1). In an oracle case, Alice would be able to operate on other organizations' data $x_2, \ldots, x_M$ as well. Recall that x represents the ensemble of all the available data variables. In general, the most ideal case for Alice is to minimize the following empirical risk,

$$F_{\text{Joint}} = \operatorname*{argmin}_{F \in \mathcal{F}} \mathbb{E}_N L_1(y_1, F(x)), \tag{2}$$

where $\mathcal{F}$ is a supervised function class defined on the space of x.

In reality, Alice has no access to the complete data and model resources of other organizations. In this light, she shall be happy to crowdsource the learning task to other organizations to cooperatively build a model in hindsight, without the need to share any organization's local data or models. In the prediction stage, Alice can collect the information needed to form a final prediction to hopefully achieve a performance that significantly improves over her single-organization performance. To this end, we will develop a solution for Alice to achieve such a goal.

We will include detailed derivations and discussions of our solution in Subsection 3.3. For readability, we summarize notations that will be frequently used in the exposition below. Our method will require Alice to occasionally send continuous-valued vectors $r_1 = [r_{i,1}]_{i=1}^N \in \mathbb{R}^{N \times K}$ to each organization m at each communication round. These residuals, to be elaborated in the next subsection, approximate the fastest direction of reducing the training loss in hindsight, namely a sample version of $\partial L_1(y_1, F(x))/\partial F(x)$ given Alice's estimation of F at a particular time (round). Upon the input of these residual vectors, the organization m will locally learn a supervised function f_m that maps from its feature space to the residual space. With a slight abuse of notation, we also refer to f_m as the learned model. To this end, the organization m will perform the empirical risk minimization

$$f_m = \operatorname*{argmin}_{f \in \mathcal{F}_m} \mathbb{E}_N \ell_m(r_1, f(x_m)) = \operatorname*{argmin}_{f \in \mathcal{F}_m} \frac{1}{N} \sum_{i=1}^{N} \ell_m\left(r_{i,1}, f(x_{i,m})\right) \tag{3}$$

to obtain a locally trained model f_m. Here, $\mathcal{F}_m$ and ℓ_m respectively denote the supervised function class and loss function of the organization m. We note that ℓ_m are local regression loss functions for fitting the pseudo-residual r_1 and may not necessarily be the same as L_1 for fitting true labels. For example, L_1 may be the cross-entropy loss for classification of label y, while $\ell_{1:M}$ could be the squared loss for regression of the response r_1. The above local training (optimization) is often performed using the stochastic gradient descent (SGD) algorithm. In our assisted learning context, L_1, $\mathcal{F}_m$, and ℓ_m are proprietary local resources that cannot be shared across organizations.

3.3 The GAL Algorithm

We first introduce the derivation of the GAL algorithm from a functional gradient descent perspective. Then, we cast the algorithm into pseudocode and discuss each step. Consider the unrealistic case that Alice has all the data x needed for a centralized supervised function $F : x \mapsto F(x)$. Recall that the goal of Alice is to minimize the population loss $\mathbb{E}_{p_{x,y}} L_1(y, F(x))$ over a data distribution $p_{x,y}$. If $p_{x,y}$ is known, starting with an initial guess $F^0(x)$, Alice would have performed a gradient descent step in the form of

$$F^1 \leftarrow F^0 - \eta \cdot \frac{\partial}{\partial F} \mathbb{E}_{p_{x,y}} L_1(y, F(x)) \mid_{F=F^0} = F^0 - \eta \cdot \mathbb{E}_{p_{x,y}} \frac{\partial}{\partial F} L_1(y, F(x)) \mid_{F=F^0}, \tag{4}$$

where the equality holds under the standard regularity conditions of exchanging integration and differentiation. Note that the second term in (4) is a function on $\mathbb{R}^d$. However, because Alice only has access to her own data x_1, the expectation $\mathbb{E}_{p_{x,y}}$ cannot be realistically evaluated. Therefore, we need to approximate it with functions in a pre-specified function set. In other words, we will find f from $\mathcal{F}_M$ that 'best' approximates $\mathbb{E}_{p_{x,y}} \frac{\partial}{\partial F} L_1(y, F(x))$. We will show that this is actionable without requiring the organizations to share proprietary data, models, and objective functions.

Recall that $\mathcal{F}_m$ is the function set locally used by the organization m, and x_m is correspondingly observed portion of x. The function class that we propose to approximate the second term in (4) is

$$\mathcal{F}_M = \left\{ f : x \mapsto \sum_{m=1}^{M} w_m f_m(x_m), \forall f_m \in \mathcal{F}_m, x \in \mathbb{R}^d, w \in P_M \right\}, \tag{5}$$

Algorithm 1 GAL: Gradient Assisted Learning (from the perspective of the service receiver, Alice)

Input: M decentralized organizations, each holding data $\{x_{i,m}\}_{i=1}^N$ (local) corresponding to N objects, the task label $\{y_{i,1}\}_{i=1}^N$ initially held by the service receiver (Alice local), model class $\mathcal{F}_m$ (local), gradient assistance weights w (Alice local), assistance rate η (Alice local), overarching loss function L_1 (Alice local), regression loss function ℓ_m to fit pseudo-residual (local), assistance rounds T.

Learning Stage:

  Intialization:

    Let $t = 0$, and initialize $F^0(x) = \mathbb{E}_N(y_1)$

  for assistance round t from 1 to T **do**

    Compute pseudo-residual

    $r_1^t = -\left[\frac{\partial L_1\left(y_1, F^{t-1}(x)\right)}{\partial F^{t-1}(x)}\right]$

    Broadcast pseudo-residual r_1^t to other organizations

    for organization m from 1 to M *in parallel* **do**

      $f_m^t = \text{argmin}_{f_m \in \mathcal{F}_m} \mathbb{E}_N \ell_m \left(r_1^t, f_m(x_m)\right)$

    end

    Gather predictions $f_m^t(x_m)$, $m = 1, \ldots M$, from all the organizations

    Optimize the gradient assistance weights

    $\hat{w}^t = \text{argmin}_{w \in P_M} \mathbb{E}_N \ell_1 \left(r_1^t, \sum_{m=1}^M w_m f_m^t(x_m)\right)$

    Line-search for the gradient assisted learning rate

    $\hat{\eta}^t = \text{argmin}_{\eta \in \mathbb{R}} \mathbb{E}_N L_1 \left(y_1, F^{t-1}(x) + \eta \sum_{m=1}^M \hat{w}_m^t f_m^t(x_m)\right)$

    $F^t(x) = F^{t-1}(x) + \hat{\eta}^t \sum_{m=1}^M \hat{w}_m^t f_m^t(x_m)$

  end

Prediction Stage:

  For each data observation x^*, of which x_m^* is held by organization m:

  Gather predictions $f_m^t(x_m^*)$, $t = 1, \ldots, T$ from each organization m, $m = 1, \ldots, M$

  Predict with

  $F^T(x^*) \triangleq F^0(x_1^*) + \sum_{t=1}^T \hat{\eta}^t \sum_{m=1}^M \hat{w}_m^t f_m^t(x_m^*)$

where $P_M = \{w \in \mathbb{R}^M : \sum_{m=1}^M w_m = 1, w_m \geq 0\}$ denotes the probability simplex. The gradient assistance weights w_m's are interpreted as the contributions of each organization at a particular greedy update step. The gradient assistance weights are constrained to sum to one to ensure the function space is compact, and the solutions exist.

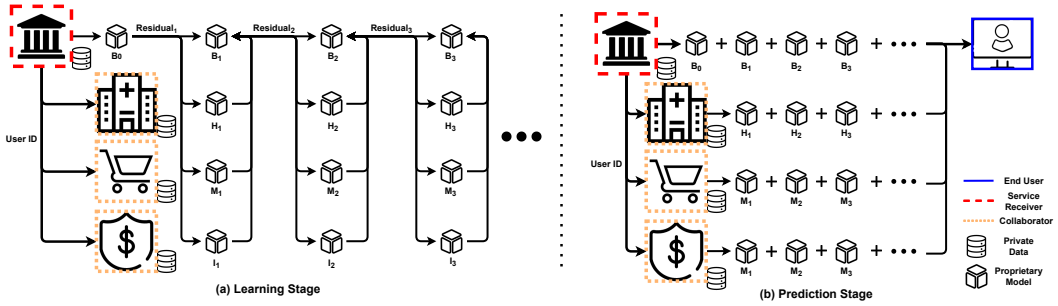


Figure 3: Learning and Prediction Stages for Gradient Assisted Learning (GAL).

We propose the following solution so that *each organization can operate on its own local data, model, and objective function*. Alice initializes with a startup model, denoted by $F^0(x) = F^0(x_1, y_1)$, based only on her local data and labels. Alice broadcasts r_1 (named 'pseudo residuals') to each organization m, $m = 2, \cdots, M$, who will then fit a local model f_m using r_1. Each organization will then send the fitted values from f_m to Alice, who will train suitable gradient assistance weights w_m. Subsequently, Alice finds the η in (4) that minimizes her current empirical risk. The above procedure is iterated for a finite number of rounds until Alice obtains a satisfactory performance (e.g., on validation data). The validation will be based on the same technique as the prediction stage to be described below.

This training stage is described under the 'learning stage' of Algorithm 1. Note that the pseudocode is from the perspective of Alice, the service receiver. For each organization m, it will only need to perform the empirical risk minimization using the label r_1^t sent by Alice at each round t.

In the Prediction/Inference stage (given above in Algorithm 1), other organizations send prediction results generated from their local models to Alice, who will calculate a prediction result $F^T(x)$ that is implicitly operated on x, where T is the number of iteration steps.

The idea of approximating functional derivatives with regularized functions was historically used to develop the seminal work of gradient boosting [9, 10]. When there is only one organization, the above method reduces to the standard gradient boosting algorithm.

Organizations in our learning framework form a shared community of interest. Each service-providing organization can provide end-to-end assistance for an organization without sharing anyone's proprietary data, models, and objective functions. In practice, the participating organizations may receive financial rewards from the one to assist. Moreover, every organization in this framework can provide its own task and seek help from others. As a result, all organizations become mutually beneficial to each other. We provide a realistic example in Figure 3 to demonstrate each step of Algorithm 1. We elaborate on the learning and prediction procedures in the Appendix.

We also provide an asymptotic convergence analysis for the GAL algorithm, where the goal is to minimize a loss $f \mapsto \mathcal{L}(f)$ over a function class through step-wise function aggregations. Because of the greedy nature of GAL, we consider the function class to be the linear span of organization-specific $\mathcal{F}_m$. The following result states that the GAL can produce a solution that attains the infimum of $\mathcal{L}(f)$. More technical details are included in the Appendix.

Theorem 1 *Assume that the loss (functional) $f \mapsto \mathcal{L}(f)$ is convex and differentiable on $\mathcal{F}$, the function $u \mapsto \mathcal{L}(f + ug)$ has an upper-bounded second-order derivative $\partial^2\mathcal{L}(f + ug)/\partial u^2$ for all $f \in span(\mathcal{F}_1, \ldots, \mathcal{F}_M)$ and $g \in \cup_{m=1}^M \mathcal{F}_m$, and the ranges of learning rates $\{a_t\}_{t=1,2,\ldots}$ satisfy $\sum_{t=1}^\infty a_t = \infty$, $\sum_{t=1}^\infty a_t^2 < \infty$. Then, the GAL algorithm satisfies $\mathcal{L}(F^t) \to \inf_{f \in span(\mathcal{F}_1,\ldots,\mathcal{F}_M)} \mathcal{L}(f)$ as $t \to \infty$.*

4 Experimental Studies

Baselines Our experiments are performed with four baselines, including 'Interm', 'Late', 'Joint', 'Alone', and 'AL'. 'Interm' and 'Late' refer to intermediate and late data fusions [7, 8], respectively. The intermediate data fusion ('Interm') sums up the intermediate features (output before the last layer) from each separate feature extractor, and then the aggregated feature is passed into a shared last layer to output the final prediction. The late data fusion ('Late') sums up the final prediction of each separate local model. Thus, 'Interm' works only for deep learning models such as CNN and LSTM by averaging the hidden representation of each local model, while 'Late' also works for Linear models as it aggregates the output of each local model. 'Joint' is the case where all the data are held by Alice and trained with the Gradient Boosting reduced from GAL. 'Alone' is the single-agent scenario, where only Alice's data are used for learning and prediction. 'AL' represents the performance of Assisted Learning [21]. GAL is expected to perform close to the centralized baselines including 'Interm', 'Late', and 'Joint' cases while significantly outperforming the 'Alone' and 'AL' cases. The summary statistics of each dataset are elaborated in Table 5 of the Appendix. Details of learning hyper-parameters are included in Table 7 of the Appendix. We conducted four random experiments for all datasets with different seeds, and the standard errors are shown in the brackets of all tables.

4.1 Model Autonomy

Recall that GAL allows each organization to choose its own local model. We demonstrate the performance of autonomous local models with UCI datasets downloadable from the *scikit-learn* package [22], including Diabetes [23], Boston Housing [24], Blob [22], Iris [25], Wine [26], Breast Cancer [27], and QSAR [28] datasets, where we randomly partition the features into 2, 4, or 8 subsets. For all the UCI datasets, we train on 80% of the available data and test on the remaining.

We experiment with Linear, Gradient Boosting (GB), and Support Vector Machine (SVM) for local models $f_m(\cdot)$ with the UCI datasets. We also demonstrate the performance of the scenario (GB-SVM) where half of the organizations use GB and the other half uses SVM. The experimental results are shown in Tables 1. Our method significantly outperforms the baselines 'Alone' and 'AL.' The results

Table 1: Results of the UCI datasets ($M = 8$) with Linear, GB, SVM and GB-SVM models. The Diabetes and Boston Housing (regression) are evaluated with Mean Absolute Deviation (MAD), and the rest (classification) are evaluated with Accuracy. More results for $M = 2$ and 4 can be found in the Appendix.

Dataset	Model	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)
Late	Linear	136.2(0.1)	8.0(0.0)	100.0(0.0)	100.0(0.0)	96.9(0.4)	76.9(0.8)
Joint	Linear	43.4(0.3)	3.0(0.0)	100.0(0.0)	100.0(0.0)	98.9(0.4)	84.0(0.2)
Alone	Linear	59.7(9.2)	5.8(0.9)	41.3(10.8)	63.9(15.6)	92.5(3.4)	68.8(3.4)
AL	Linear	51.5(4.6)	4.7(0.6)	97.5(2.5)	95.1(3.6)	97.7(1.1)	70.6(5.2)
GAL	Linear	42.7(0.6)	3.2(0.2)	100.0(0.0)	96.5(3.0)	98.5(0.7)	82.5(0.8)
GAL	GB	56.5(2.8)	3.8(0.5)	96.3(2.2)	95.8(1.4)	96.1(1.0)	84.8(0.9)
GAL	SVM	46.6(1.4)	2.9(0.2)	96.3(4.1)	96.5(1.2)	99.1(1.1)	85.5(0.7)
GAL	GB-SVM	49.8(2.6)	3.4(0.8)	70.0(7.9)	95.8(1.4)	93.2(1.6)	82.9(1.5)

also demonstrate that with GAL, an organization with little informative data and free choice of its local model (model autonomy) can leverage others' local data and models and even achieve near-oracle performance. We point out that although 'Interm,' 'Late,' and 'Joint' marginally outperform our method, they require training from centralized data. Our GAL algorithm replaces the true labels used in 'Interm,' 'Late,' and 'Joint' centralized cases with pseudo-residuals. The results from both regression and classification datasets with various model settings lead to similar conclusions.

4.2 Deep Model Sharing

We demonstrate our method is effective for deep models by using MNIST [29] and CIFAR10 [30] image datasets, where we split each image into patches as depicted in Figure 6. We use Convolutional Neural Networks (CNN) for both datasets, and the model architecture can be found in Table 6 of the Appendix. We visualize the performance of CIFAR10 at each assistance round in Figure 4 (a-c). The number of assistance rounds needed to approach the centralized performance is small (e.g., often within ten). The experimental results are shown in Tables 2. GAL significantly outperforms the bottom line 'Alone' in all the settings. This is expected since the first organization holds partial data and does not receive any assistance under 'Alone'. Interestingly, the performance of MNIST for $M = 8$ drops significantly under 'Alone' because the organization only holds the left upper image patch, which is usually completely dark, as shown in Figure 6.

Because local deep learning models such as CNN can consume extensive computation space, we propose Deep Model Sharing (DMS) to allow sharing feature extractors of deep models across all iterations to save memory. In particular, we propose to jointly train the feature extractors with residuals from previous iterations as well as from the current iteration by adding an additional last prediction layer. The local deep model $f_m^t(\cdot)$ is composed of $f_{m,o}^t(f_{m,e}(\cdot)))$, where $f_{m,o}^t(\cdot)$ is the last output layer at assistance round t and $f_{m,e}(\cdot)$ is the deep feature extractor shared across multiple assistance rounds. For each assistance round, local organizations fits pseudo-residuals across previous assistance rounds $r_1^{1:t}$ with $f_m^{1:t} = \text{argmin}_{f_m \in \mathcal{F}_m} \mathbb{E}_N \ell_m \left(r_1^{1:t}, f_{m,o}^{1:t}(f_{m,e}(x_m))\right)$. It is worth mentioning that we do not expect such trade-off GAL_{DMS} to consistently outperform AL and GAL because a single feature extractor may not well fit residuals across many iterations. The results in Table 2 show that sharing the feature extractor across multiple assistance rounds can still outperform the 'Alone' case. Thus, DMS can provide a trade-off between predictive performance and computation space. The detailed comparisons can be found in Table 12.

Table 2: Results of the MNIST and CIFAR10 ($M = 8$) datasets with CNN model. The MNIST and CIFAR10 are evaluated with Accuracy. GAL_{DMS} represents the results with Deep Model Sharing. More results for $M = 2$ and 4 are in the Appendix.

Dataset	MNIST(↑)	CIFAR10(↑)
Interm	99.4(0.0)	81.1(0.3)
Late	99.0(0.0)	81.0(0.2)
Joint	99.4(0.0)	80.1(0.2)
Alone	96.7(0.2)	72.7(0.2)
AL	96.4(0.1)	74.7(0.3)
GAL	98.5(0.2)	**78.7(0.4)**
GAL_{DMS}	**98.7(0.1)**	74.3(0.5)

4.3 Comparison with AL

As demonstrated in our extensive experiments, the proposed GAL outperforms AL in terms of predictive performance. In particular, AL converges not only worse but also slower than GAL. This is due to that AL uses a constant assisted learning rate and trains participating organizations in a sequential manner. Moreover, sequentially training participating organizations also requires much more computation and communication overhead. We compare the computation and communication complexity between AL and GAL under the constraint of the same communication cost as demonstrated in Table 12. Because AL sequentially trains each organization while GAL allows

organizations to train locally in parallel, the computation time and communication round of AL is $M\times$ those of GAL. The GAL with Deep Model Sharing (GAL_{DMS}) saves $T\times$ computation space by sharing the feature extractor of deep models. In summary, GAL generalizes the problem scope, reduces the computation and communication complexity, and achieves significantly better results.

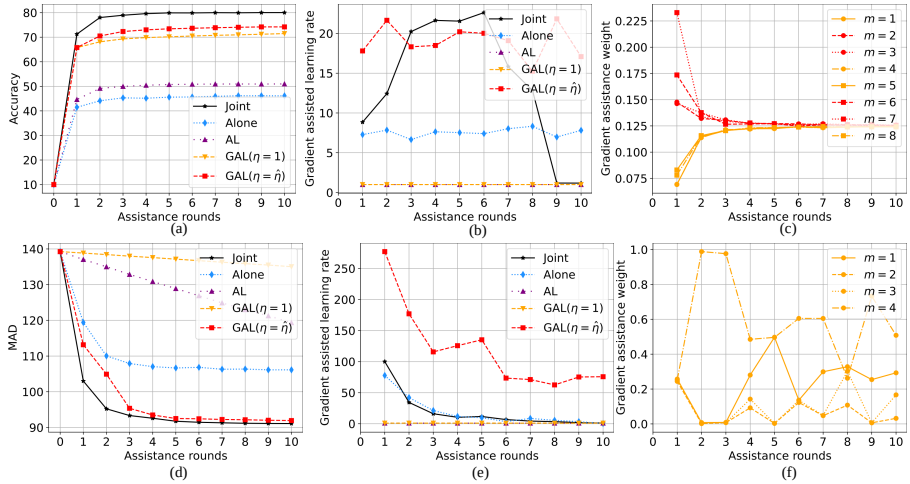


Figure 4: Results of the CIFAR10 (a-c) ($M = 8$) and MIMICL (d-f) ($M = 4$) datasets. GAL significantly outperforms 'Alone' and 'AL'. Our method also performs close to the centralized baselines. The gradient assisted learning rate diminishes to zero as the overarching loss converges. A constant gradient assisted learning rate ($\eta = 1$) converges much slower. The gradient assistance weights exhibits interpretability of the importance of organizations as the weights of the central image patches ($m = \{2, 3, 6, 7\}$) of CIFAR10 dataset are larger than the boundary patches ($m = \{1, 4, 5, 8\}$) in the first few rounds. More results can be found in the Appendix.

Table 3: Results of case studies of 3D object recognition and medical time series forecasting.

Dataset	ModelNet40(↑)	ShapeNet55(↑)	MIMICL(↓)	MIMICM(↑)
Interm	75.3(18.2)	88.6(0.1)	64.6(0.9)	0.90(0.0)
Late	86.6(0.2)	88.4(0.1)	71.4(0.2)	0.91(0.0)
Joint	46.3(1.4)	16.3(0.0)	91.1(0.7)	0.82(0.0)
Alone	76.4(1.1)	81.3(0.6)	106.1(0.3)	0.78(0.0)
AL	77.3(2.8)	83.8(0.0)	119.3(0.3)	0.86(0.0)
GAL	83.0(0.2)	84.1(0.6)	**91.9(2.3)**	**0.88(0.0)**
GAL_{DMS}	**83.2(0.3)**	**85.3(0.2)**	97.7(2.9)	0.81(0.0)

4.4 Case Studies

The results in Table 3 demonstrate the utility of GAL in various practical applications. We illustrate the results across multiple assistance rounds of MIMICL in Figure 4 (d-f). GAL significantly outperforms 'Alone' and 'AL.' Our method also performs close to the centralized baselines.

Three-dimensional object recognition We use the shape representation of 3D objects for recognition from a collection of rendered views on 2D images. We generate $M = 12$ 2D camera views of ModelNet40 [31] and ShapeNet55 [32] datasets following [33]. Cameras are treated as decentralized learners in our experiments. We adopt the same CNN architecture used for MNIST and CIFAR10 datasets. It is worth mentioning that 'Joint' of ModelNet40 and ShapeNet55 performs considerably worse than other baselines because 'Joint' uses a single feature a single CNN feature extractor to process all images from 12 angles [33].

Medical time series forecasting We use the in-hospital dataset MIMIC3 [34], where the task aims to predict the Length-of-stay (MIMICL) and Mortality rate (MIMICM) of the ICU stays of patients. We follow the benchmark work [35] to process raw data and split the features into four organizations, including 1) microbiology measurement, 2) demographic information, 3) body measurement, and 4) International Classification of Diseases (ICD). We use MAD to evaluate the result of MIMICL (regression) and AUCROC to evaluate the result of MIMICM (imbalanced binary classification). Our backbone model (LSTM) is the same as the one used in the benchmark work [36]. It is worth noting that the data features among four organizations are aligned with time stamps which is a natural identifier for time series forecasting.

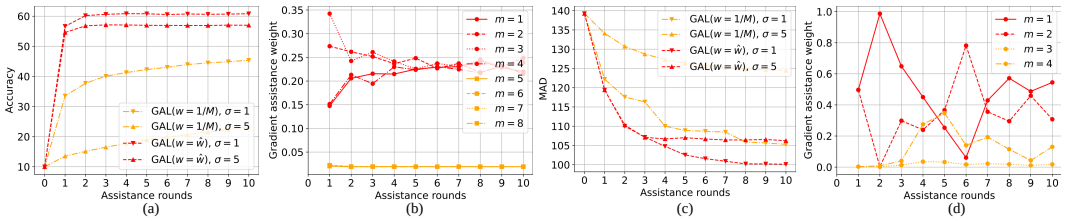


Figure 5: Ablation study results on CIFAR10 (a-b) ($M = 8$) and MIMICL (c-d) ($M = 4$) datasets. Plots (a,c) show that the GAL equipped with gradient assistance weight significantly outperforms the GAL with direct average under noise injections ($\mathcal{N}(0, \sigma^2)$, $\sigma = \{1, 5\}$) to the transmitted pseudo-residual to half of the organizations during learning and prediction. Plots (b,d) show the gradient assistance weight of noisy (in orange, $\sigma = 1$) and noise-free organizations (in red).

Table 4: Ablation study on the local objective function ($M = 8$). More results are in the Appendix.

Dataset	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)
Alone	59.7(9.2)	5.8(0.9)	41.3(0.0)	63.9(0.0)	92.5(0.4)	68.8(0.2)	24.2(0.1)	46.3(0.3)
ℓ_1	**42.7(0.6)**	3.2(0.2)	42.5(12.5)	95.1(4.1)	97.4(1.1)	64.3(3.6)	90.5(1.9)	27.8(1.7)
$\ell_{1.5}$	43.4(1.0)	**2.9(0.1)**	100.0(0.0)	95.8(4.2)	97.4(0.6)	80.2(1.3)	94.5(0.1)	70.2(0.8)
ℓ_2	44.8(1.9)	3.0(0.1)	**100.0(0.0)**	**96.5(3.0)**	98.5(0.7)	**82.5(0.8)**	96.3(0.6)	**74.3(0.2)**
ℓ_4	45.8(1.3)	3.2(0.2)	97.5(4.3)	98.6(1.4)	**99.1(0.9)**	81.3(0.7)	**98.1(0.1)**	73.2(0.3)
(ℓ_1, ℓ_2)	43.3(1.5)	3.2(0.3)	100.0(0.0)	96.5(3.0)	96.7(0.7)	80.0(1.0)	94.6(1.1)	65.0(0.2)

4.5 Ablation Studies

Local objective functions Our method does not require local regression loss functions ℓ_m to be shared with other organizations, but they are supposed to be beneficial for assisting Alice. We conduct ablation studies on the choice of various local regression loss functions. In this study shown in Table 4, we use different regression functions $\ell_q(y, \hat{y}) = |y - \hat{y}|^q$ where $q \in \{1, 1.5, 2, 4\}$. (ℓ_{q_1}, ℓ_{q_2}) allows half of organizations including to use ℓ_{q_1} while the other half to use l_{q_2}. The results show that classification task generally works better with $q > 1$.

Gradient assisted learning rate We illustrate the gradient assisted learning rate of CIFAR10 and MIMICL datasets at each assistance round in Figure 4(b,e). We perform a line-search for the gradient assisted learning rate with the Limited-Memory BFGS optimizer which improves the convergence rates compared with SGD and Adam. We conduct ablation study using a constant gradient assisted learning rate ($\eta = 1$). As shown in Figure 4(a,d), the constant gradient assisted learning rate leads to a convergence much slower than the line-search method. Fast convergence is desirable since the computation and communication cost increases with the number of assistance rounds. To determine the maximal number of assistance rounds T for the service receiver, we can run the GAL procedure until the gradient assisted learning rate becomes small. When the gradient assistance rate is low, as shown in Figure 4(e), the overarching loss converges to zero. In this light, an organization may stop receiving assisted learning when the gradient assisted learning rate is below a threshold.

Gradient assistance weights We show the gradient assistance weights of CIFAR10 and MIMICL datasets at each assistance round in Figure 4(c,f). The results of MNIST and CIFAR10 show that the gradient assistance weights exhibit interpretability of the importance of organizations because the image patches with dominant contributions are $m = (2, 3, 6, 7)$ (colored in red). These image patches correspond to the center of the original image, which matches our intuition appealingly. The weights converge to uniform as the residuals of later iterations are small. We also conduct ablation study of the gradient assistance weights in Figure 5 by adding noises (Gaussian with zero mean and σ^2 variance, $\sigma \in \{1, 5\}$) to the output of a randomly chosen half of the clients during learning and prediction. The purpose of adding noise is to simulate realistic scenarios where some assisting organizations are uninformative or inject adversarial behavior. We summarize the ablation study results in Tables 16-18 in the Appendix. The results show that the GAL equipped with gradient assistance weights is more robust than the GAL with a direct average.

5 Conclusion

We proposed Gradient Assisted Learning, a decentralized learning method for multiple organizations to collaborate without sharing data, models, and objective functions. The proposed method can significantly outperform the local learning baselines and achieve near-oracle performance as if data were centralized on various datasets. All participants form a shared community of interest by autonomously building its own model and iteratively fitting the gradients of the overarching objective function. We also demonstrate asymptotic convergence analysis and practical case studies of GAL.

References

[1] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Comput. Netw.*, 57(10):2266–2279, 2013.

[2] John T Farrar, Andrea B Troxel, Kevin Haynes, Ian Gilron, Robert D Kerns, Nathaniel P Katz, Bob A Rappaport, Michael C Rowbotham, Ann M Tierney, Dennis 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®*, 155(8):1622–1631, 2014.

[3] Bernard Lo. Sharing clinical trial data: maximizing benefits, minimizing risk. *JAMA*, 313(8):793–794, 2015.

[4] Lin Zhu, Dafeng Qiu, Daji Ergu, Cai Ying, and Kuiyi Liu. A study on predicting loan default based on the random forest algorithm. *Procedia Computer Science*, 162:503–513, 2019.

[5] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In *Proc. ICMLA*, pages 896–902. IEEE, 2015.

[6] Xinran Wang, Yu Xiang, Jun Gao, and Jie Ding. Information laundering for model privacy. *Proc. ICLR*, 2021.

[7] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.

[8] Dana Lahat, Tülay Adali, and Christian Jutten. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015.

[9] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent in function space. 1999.

[10] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, pages 1189–1232, 2001.

[11] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proc. CCS*, pages 1310–1321, 2015.

[12] Jakub Konecny, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

[14] Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

[15] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[16] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *in Proc. TIST*, 10(2):12, 2019.

[17] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient vertical federated learning framework. *arXiv preprint arXiv:1912.11187*, 2019.

[18] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *Proc. ICML*, pages 3973–3983, 2020.

[19] Chandra Thapa, Mahawaga Arachchige Pathum Chamikara, and Seyit Camtepe. Splitfed: When federated learning meets split learning. *arXiv preprint arXiv:2004.12088*, 2020.

[20] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. Vafl: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

[21] Xun Xian, Xinran Wang, Jie Ding, and Reza Ghanadan. Assisted learning: A framework for multi-organization learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *Ann. Stat.*, 32(2):407–499, 2004.

[24] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *J. Environ. Econ. Manag.*, 5(1):81–102, 1978.

[25] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[26] Stefan Aeberhard, Danny Coomans, and Olivier De Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, 27(8):1065–1077, 1994.

[27] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. International Society for Optics and Photonics, 1993.

[28] Kamel Mansouri, Tine Ringsted, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. Quantitative structure–activity relationship models for ready biodegradability of chemicals. *J. Chem. Inf. Model.*, 53(4):867–878, 2013.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[32] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015.

[33] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[34] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Sci. Data*, 3:160035, 2016.

[35] Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking deep learning models on large healthcare datasets. *J. Biomed. Inform.*, 83:112–134, 2018.

[36] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Sci. Data*, 6(1):1–18, 2019.

[37] Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.

[38] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[39] Jie Ding and Bangjun Ding. Interval privacy: A privacy-preserving framework for data collection. *arXiv preprint arXiv:2106.09565*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4.2 for computation space and Section B for theoretical analysis.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not foresee any negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section B.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide source codes in the supplementary material. We use public available datasets.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section C in Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The numerical standard error are shown in brackets of tables.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] One Nvidia 1080TI is enough for one experiment run.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the public available datasets we use.
   (b) Did you mention the license of the assets? [No] They should be available publicly.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use public available datasets.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use public available datasets.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix for "Gradient Assisted Learning"

The **Appendix** contains additional discussions, technical results, and further experimental results.

A Application Scenario

As shown in Figure 1, Alice, the service receiver (bank) squared in red dashed line, is the organization to be assisted. Before learning, it broadcasts identification (ID) to locate and align vertically distributed data held by other organizations. At the beginning of the Learning Stage, the bank deterministically initializes the values of $F^0(x)$ to be the unbiased estimate of y_1, namely $F^0(x) = \mathbb{E}_N(y_1^0)$. For the regression task, $F^0(x)$ is a single scalar. For classification task, $F^0(x)$ is a point in the K-dimensional simplex P_K.

During the first assistance round in the Learning Stage, the bank computes pseudo-residual r_1^1 and broadcasts it to other organizations (e.g., hospital, mall, and insurance company). Then, all the organizations, including the bank, will fit a new local model with 1) their local data, 2) the pseudo-residual r_1^1, and 3) their local regression loss function ℓ_m (e.g., ℓ_2-loss) to fit the pseudo-residual. We note that organizations have complete autonomy on model fitting. In particular, they can choose their own learning algorithms and models by considering their resources (e.g., computation power). Next, the bank will aggregate all the predictions from each organization's local models by optimizing a weight vector $w_{1:M}$ referred to as gradient assistance weights. As previously discussed in Equation (5), we approximate the oracle gradient (operated on centralized data, in hindsight) with a weighted average of those predictions from organizations. We then numerically search for the learning rate η. This process can be iterated multiple times until the learning rate is low or the validation loss is satisfactory.

During the Prediction Stage, organizations will predict with trained models at every assistance round and transmit their predictions to the bank. Similar to the Learning Stage, *the synchronization of each organization is unnecessary*. The bank computes the final prediction with gradient assistance weights, learning rates, and received predictions.

B Theoretical Analysis

To develop a convergence analysis of the GAL algorithm, we use the following notations. We still let $\mathcal{F}_m$ (for each $m = 1, \ldots, M$) denote a set of real-valued functions defined on organization m's data x_m. For notational simplicity, for each $f_m \in \mathcal{F}_m$, we also treat it as a function of the (artificially) extended variable $x = [x_1, \ldots, x_M]$. So, we may write a function in the form of $f_1 + f_2$, which basically means $[x_1, x_2] \mapsto f_1(x_1) + f_2(x_2)$. Let $\mathcal{L}$ denote the overarching loss function to minimize (for the agent to assist), and P_M the probability simplex.

As a summary, we abstract the core steps in Algorithm 1 below. For each organization m at assistance round t, it optimizes each local model by solving

$$(\alpha_t, f_m^t) = \operatorname*{argmin}_{\alpha \in [-a_t, a_t], f_m \in \mathcal{F}_m} \mathcal{L}(F^{t-1} + \alpha f_m). \tag{6}$$

Alice then gathers predictions f_m^t, $m = 1, \ldots M$, from all the organizations, and optimize the gradient assistance weights and learning rate by solving

$$(\hat{w}^t, \hat{\eta}^t) = \operatorname*{argmin}_{w \in P_M, \eta \in [-a_t, a_t]} \mathcal{L}\left(F^{t-1} + \eta \sum_{m=1}^{M} w_m f_m^t\right). \tag{7}$$

At round $t = 0$, we initialize with any $F^0 \in \mathcal{F}_1$. At each round t, each organization first runs a greedy boosting step to obtain (α_t, f_m^t). The f_m^t will be sent to us (the organization to assist). Then, we run another greedy step to optimize the assistance weights $\hat{w}^t$ and learning rate $\hat{\eta}^t$, with fixed f_m^t, $m = 1, \ldots, M$. The weighted function will be added to F^{t-1} to generate the latest F^t.

For each m, we let

$$\text{span}(\mathcal{F}_m) = \left\{ \sum_{j=1}^{K_m} \mu_j f_j : \mu_j \in \mathbb{R}, f_j \in \mathcal{F}_m, K_m \in \mathbb{N}^+ \right\},$$

which is the function space formed from linear combinations of elements in $\mathcal{F}_m$. Let

$$\text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M) = \left\{ \sum_{m=1}^{M} w_m f_m : w_m \in \mathbb{R}, f_m \in \text{span}(\mathcal{F}_m) \right\}$$

denote the linear span of the union of $\mathcal{F}_1, \ldots, \mathcal{F}_M$. An equivalent way to write it is $\text{span}(\cup_{m=1}^M \mathcal{F}_m)$.

We will show the following convergence result. With a suitable choice of step parameters a_t and regularity conditions of the loss $\mathcal{L}$, the abstract form of GAL can produce F^t that asymptotically attains the minimum loss within the function class $\text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. We make the following technical assumptions.

(A1) The loss (functional) $f \mapsto \mathcal{L}(f)$ is convex and differentiable on $\mathcal{F}$, with gradient $\nabla \mathcal{L}$. Also, for all $f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$ and $g \in \cup_{m=1}^M \mathcal{F}_m$, the function $u \mapsto \mathcal{L}(f + ug)$ has a second order derivative $\partial^2 \mathcal{L}(f + ug)/\partial u^2$, and it is upper bounded by a fixed constant C.

(A2) The ranges of learning rates $\{a_t\}_{t=1,2,\ldots}$ satisfy $\sum_{t=1}^\infty a_t = \infty$, $\sum_{t=1}^\infty a_t^2 < \infty$.

Theorem 1: Under Assumptions (A1) and (A2), the GAL algorithm satisfies $\mathcal{L}(F^t) \to \inf_{f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f)$ as $t \to \infty$.

Remarks on Theorem 1: The result says that with suitable control of the learning rates, the greedy procedure in Algorithm 1 can converge to the oracle one could obtain within $\text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. Suppose that an organization, say the one indexed by $m = 1$, does not collaborate with others. Likewise, we have the convergence for that particular organization, $\lim_{t \to \infty} \mathcal{L}(F^t) = \inf_{f^* \in \text{span}(\mathcal{F}_1)} \mathcal{L}(f^*)$. It can be seen that the GAL will produce a significant gain for this organization as long as

$$\inf_{f^* \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f^*) < \inf_{f^* \in \text{span}(\mathcal{F}_1)} \mathcal{L}(f^*). \tag{8}$$

It is conceivable that (8) is easy to meet in many practical scenarios since each $\mathcal{F}_m$ is operated on a particular modality of data that belongs to organization m. On the other hand, a skeptical reader may wonder how the GAL solution compares with a function learned from the pulled data. It is possible that the global minimum of $\mathcal{L}$ (over functions that operate on the pulled data) does not belong to $\text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. If that is the case, the best we can do is to find f that attains the limit $\inf_{f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f)$. This is a limitation due to the constraint that organizations cannot share data and the additive structure of $\text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$. Fortunately, in various real-data experiments we performed, the GAL often performs close to the centralized learning within only a few assistance rounds.

In the technical result, we could allow the approximate minimization of (6) and (7), meaning that the loss of the produced solution is δ_t-away from the optimal loss. In that case, it can be verified that $\sum_{t=1}^\infty \delta_t < \infty$ is sufficient to derive the same asymptotic result in Theorem 1.

The proof of Theorem 1 uses the same technique as was used in [37]. The technical result here is nontrivial, because f_m^t $(m = 1, \ldots, M)$ in each round t are not jointly minimized with $\hat{w}^t$ and $\hat{\eta}^t$ in (7), and thus their linear combination may not be the most greedy solution of minimizing $\mathcal{L}(F^{t-1} + f)$ within $f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$.

A limitation of our theoretical result is that it does not explain our experimental observations that GAL requires only a few rounds of assistance to obtain excellent performance. We leave a more sophisticated convergence analysis as future work.

Proof of Theorem 1:

Let $f^* \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$ be an arbitrary fixed function. It is introduced for technical convenience and can be treated as the function that (approximately) attains the infimum of $L(f)$.

For every $f \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)$, we define the following norm with respect to the basis functions,

$$\|f\|_1 = \inf\Big\{ \|\mu\|_1 : \sum_{m=1}^{M} \sum_{j=1}^{K_m} \mu_{m,j} f_{m,j} : \mu_{m,j} \in \mathbb{R}, f_{m,j} \in \mathcal{F}_m, K_m \in \mathbb{N}^+ \Big\}$$

where $\|\mu\|_1$ denotes the abstract sum of its entries, namely $\sum_{m=1,\ldots,M,j=1,\ldots,K_m} |\mu_{m,j}|$.

For each t, let $S_t \subset \cup_{m=1}^{M} \mathcal{F}_m$ denote the finite set of functions such that
1) $f_m^\tau \in S_t$ for all $0 < \tau < t$, and
2) $f^* = \sum_{g \in S_t} \mu_{f^*}^g g$ ($\mu^g \in \mathbb{R}$), with $\|\mu_{f^*}\|_1 \leq \|f^*\|_1 + \varepsilon$.
Note that S_t exists due to the definition of $\|\cdot\|_1$ and the construction of each f_m^τ. Suppose that F^{t-1} admits the representation $F^{t-1} = \sum_{g \in S_t} \mu_{F^{t-1}}^g g$.

From (7), we have

$$\mathcal{L}(F^t) \leq \mathcal{L}\big(F^{t-1} + \hat{\eta}_t f_m^t\big), \quad \forall m = 1, \ldots, M. \tag{9}$$

Meanwhile, it follows from (6) that for each m, and each $g \in S_t \cap \mathcal{F}_m$,

$$\mathcal{L}\big(F^{t-1} + \hat{\eta}_t f_m^t\big) \leq \mathcal{L}\big(F^{t-1} + a_t s^g g\big). \tag{10}$$

where $s^g \triangleq \text{sign}(\mu_{f^*}^g - \mu_{F^{t-1}}^g)$. Combining (9) and (10), we obtain

$$\mathcal{L}(F^t) \leq \mathcal{L}\big(F^{t-1} + a_t s^g g\big), \quad \forall g \in S_t. \tag{11}$$

Applying Taylor expansion to $f \mapsto \mathcal{L}(f)$ at $f = F^{t-1}$, and invoking (11) and Assumption (A1), we have

$$\mathcal{L}(F^t) - \mathcal{L}(F^{t-1}) \leq \mathcal{L}(F^{t-1} + a_t s^g g) - \mathcal{L}(F^{t-1}) \leq a_t s^g \nabla \mathcal{L}(F^{t-1})^{\mathrm{T}} g + \frac{C}{2} a_t^2 \tag{12}$$

for all sufficiently small $a_t > 0$. Let $\|\mu_{f^*} - \mu_{F^{t-1}}\|_1 \triangleq \sum_{g \in S_t} |\mu_{f^*}^g - \mu_{F^{t-1}}^g|$. Multiplying both sides by $|\mu_{f^*}^g - \mu_{F^{t-1}}^g|$, and add up all the $g \in S_t$, we have

$$\begin{aligned} \|\mu_{f^*} - \mu_{F^{t-1}}\|_1 \cdot \{\mathcal{L}(F^t) - \mathcal{L}(F^{t-1})\} &\leq a_t \nabla \mathcal{L}(F^{t-1})^{\mathrm{T}} (f^* - F^{t-1}) + \frac{C}{2} a_t^2 \\ &\leq a_t \{\mathcal{L}(f^*) - \mathcal{L}(F^{t-1})\} + \frac{C}{2} a_t^2 \end{aligned} \tag{13}$$

where the last inequality is due to the convexity of $\mathcal{L}$. If $\|\mu_{f^*} - \mu_{F^{t-1}}\|_1 = 0$, F^{t-1} already converges to f^*. Otherwise, we rearrange (13) to obtain

$$\mathcal{L}(F^t) - \mathcal{L}(f^*) \leq \left(1 - \frac{a_t}{\|\mu_{f^*} - \mu_{F^{t-1}}\|_1}\right) \{\mathcal{L}(F^{t-1}) - \mathcal{L}(f^*)\} + \frac{C}{2} a_t^2 \tag{14}$$

$$\leq \left(1 - \frac{a_t}{\|\mu_{f^*}\|_1 + 1 + \sum_{\tau=0}^{t-1} a_\tau}\right) \{\mathcal{L}(F^{t-1}) - \mathcal{L}(f^*)\} + \frac{C}{2} a_t^2, \tag{15}$$

where the last inequality is due to the triangle inequality, the way F^{t-1} is constructed, and the fact that ε can be arbitrarily chosen. Here, we defined $a_0 \triangleq 0$. Let $a_{1:t} = \sum_{\tau=1}^{t} a_\tau$ for each $t \geq 1$. Applying (15) and the Lemma 4.2 in [37], we have

$$\max(0, \mathcal{L}(F^t) - \mathcal{L}(f^*)) \leq \frac{\|\mu_{f^*}\|_1 + 1}{\|\mu_{f^*}\|_1 + a_{1:t}} + \frac{C}{2} \sum_{\tau=1}^{t} \frac{\|\mu_{f^*}\|_1 + a_{1:\tau}}{\|\mu_{f^*}\|_1 + a_{1:t}} a_\tau^2. \tag{16}$$

Since f^* is arbitrarily chosen, it can be seen from (16) and Assumption (A2) that $\lim_{t \to \infty} \mathcal{L}(F^t) = \inf_{f^* \in \text{span}(\mathcal{F}_1, \ldots, \mathcal{F}_M)} \mathcal{L}(f^*)$.

C Experimental Setup

C.1 Dataset

In Table 5, we illustrate the statistics of datasets used in our experiments. In Figure 6, we show how MNIST and CIFAR10 images are split into 2, 4, and 8 image patches. The left upper image patch (labeled as $[1]$) of MNIST image is less informative which demonstrates that an organization with little informative data can leverage other organizations' local data and models. The central image patches (labeled $[2, 3, 6, 7]$) of MNIST and CIFAR10 images are more informative than others, which leads to larger corresponding gradient assistance weights.

Table 5: Detailed statistics used in each data experiment. The variables d and K respectively denote the number of features (or the shape of the image) and the length of the prediction vector (or equivalently, the number of classes in the classification task).

Dataset	N_{train}	N_{test}	d	K	M
Diabetes	353	89	10	1	{2, 4, 8}
BostonHousing	404	102	13	1	{2, 4, 8}
Blob	80	20	10	10	{2, 4, 8}
Iris	120	30	4	3	{2, 4}
Wine	142	36	13	3	{2, 4, 8}
BreastCancer	455	114	30	2	{2, 4, 8}
QSAR	844	211	41	2	{2, 4, 8}
MNIST	60000	10000	(1, 28, 28)	10	{2, 4, 8}
CIFAR10	50000	10000	(3, 32, 32)	10	{2, 4, 8}
ModelNet40	3163	800	(12, 3, 32, 32,3 2)	40	{12}
ShapeNet55	35764	5159	(12, 3, 32, 32, 32)	55	{12}
MIMICL	34387	6057	22	1	{4}
MIMICM	17902	3236	22	1	{4}

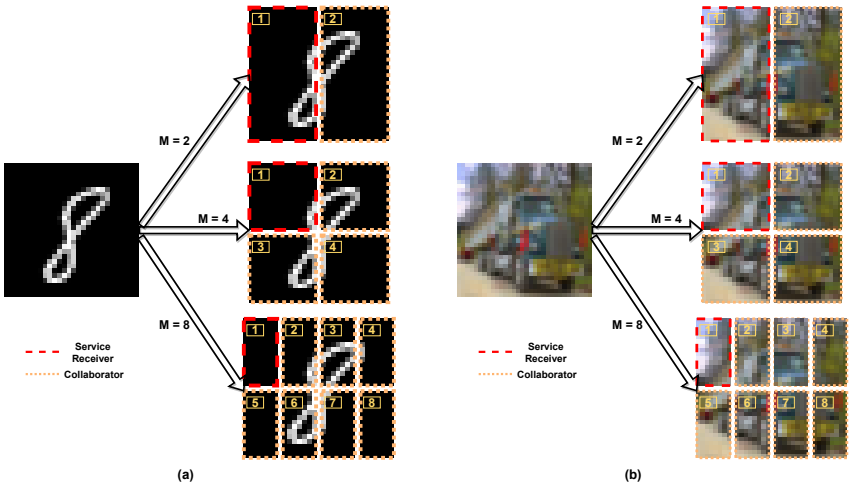


Figure 6: An illustration of (a) MNIST and (b) CIFAR10 data split into 2, 4, and 8 image patches. The left upper image patch (labeled $[1]$) of MNIST images is less informative in general. In contrast, the central image patches (labeled $[2, 3, 6, 7]$) of MNIST and CIFAR10 images are more informative.

C.2 Model and Hyperparameters

Table 6 summarizes the deep neural network architecture used for the MNIST, CIFAR10, ModelNet40, and ShapeNet55 datasets. Table 7 shows the hyperparameters used in our experiments.

Table 6: The model architecture of Convolutional Neural Networks (CNN) used in our experiments of the MNIST, CIFAR10, ModelNet40, and ShapeNet55 datasets. The n_c, H, W represent the shape of images, namely the number of image channels, height, and width, respectively. K is the number of classes in the classification task. The BatchNorm and ReLU layers follow Conv(output channel size, kernel size, stride, padding) layers. The MaxPool(output channel size, kernel size) layer reduces the height and width by half.

Image $x \in \mathbb{R}^{n_c \times M \times M}$
Conv(64, 3, 1, 1)
MaxPool(64, 2)
Conv(128, 3, 1, 1)
MaxPool(128, 2)
Conv(256, 3, 1, 1)
MaxPool(256, 2)
Conv(512, 3, 1, 1)
MaxPool(512, 2)
Global Average Pooling
Linear(512, K)

Table 7: Hyperparameters used in our experiments for training local models, gradient assisted learning rates, and gradient assistance weights.

Dataset		UCI	MNIST	CIFAR10	ModelNet40	ShapeNet55	MIMICL	MIMICM
Architecture		Linear	CNN				LSTM	
Local	Epoch	100	10					
	Batch size	1024	512		64		8	
	Optimizer	SGD					Adam	
	Learning rate	1.0E-01					1.0E-03	
	Weight decay	5.0E-04						
Gradient assisted learning rate	Epoch	10						
	Batch size	Full						
	Optimizer	L-BFGS						
	Learning rate	1						
Gradient assistance weights	Epoch	100						
	Batch size	1024						
	Optimizer	Adam						
	Learning rate	1.0E-01						
	Weight decay	5.0E-04						
Assistance rounds		10						

D Further Experimental Results

D.1 Model Autonomy

In Tables 8 and 9, we demonstrate the results of our experiments related to model autonomy for $M = 2$ and 4 respectively. Our method significantly outperforms the baselines 'Alone' and 'AL.' The results also demonstrate that with GAL, an organization with little informative data and free choice of its local model (model autonomy) can leverage other organizations' local data and models and even achieve near-oracle performance.

Table 8: Results of the UCI datasets ($M = 2$) with Linear, GB, SVM and GB-SVM models. The Diabetes and Boston Housing (regression) are evaluated with Mean Absolute Deviation (MAD), and the rest (classification) are evaluated with Accuracy.

Dataset	Model	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Iris(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)
Late	Linear	120.2(0.1)	3.6(0.1)	100.0(0.0)	100.0(0.0)	100.0(0.0)	99.3(0.4)	81.4(0.4)
Joint	Linear	43.4(0.3)	3.0(0.0)	100.0(0.0)	99.2(1.4)	100.0(0.0)	99.1(0.4)	84.0(0.2)
Alone	Linear	46.8(3.5)	4.1(0.7)	100.0(0.0)	92.5(6.0)	93.1(6.4)	98.9(0.6)	79.9(1.0)
AL	Linear	63.7(1.5)	3.9(0.6)	98.8(2.2)	95.0(2.9)	95.1(2.3)	97.6(0.7)	80.6(1.6)
GAL	Linear	43.2(0.8)	2.9(0.1)	100.0(0.0)	99.2(1.4)	96.5(2.3)	98.9(0.4)	83.8(0.4)
GAL	GB	49.1(2.7)	3.0(0.3)	97.5(2.5)	95.8(1.4)	98.6(1.4)	95.6(1.1)	85.1(1.0)
GAL	SVM	42.6(1.9)	2.5(0.1)	100.0(0.0)	96.7(0.0)	95.1(1.2)	99.6(0.4)	87.3(1.0)
GAL	GB-SVM	50.9(2.9)	3.1(0.5)	96.3(6.5)	96.7(0.0)	93.7(4.6)	94.7(0.6)	82.7(0.4)

Table 9: Results of the UCI datasets ($M = 4$) with Linear, GB, SVM and GB-SVM models. The Diabetes and Boston Housing (regression) are evaluated with Mean Absolute Deviation (MAD), and the rest (classification) are evaluated with Accuracy.

Dataset	Model	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Iris(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)
Late	Linear	129.5(0.1)	4.7(0.0)	100.0(0.0)	100.0(0.0)	100.0(0.0)	98.5(0.7)	79.7(1.2)
Joint	Linear	43.4(0.3)	3.0(0.0)	100.0(0.0)	99.2(1.4)	100.0(0.0)	98.9(0.4)	84.0(0.2)
Alone	Linear	56.6(8.2)	4.8(0.6)	80.0(6.1)	79.2(13.0)	84.7(1.4)	97.1(1.0)	73.0(1.0)
AL	Linear	58.3(2.4)	5.2(0.3)	100.0(0.0)	88.3(8.3)	92.4(2.3)	98.9(1.1)	76.8(2.5)
GAL	Linear	43.3(1.1)	3.0(0.1)	100.0(0.0)	100.0(0.0)	97.9(2.3)	99.1(0.6)	83.3(0.5)
GAL	GB	56.8(3.9)	3.2(0.4)	98.8(2.2)	96.7(0.0)	94.4(2.0)	95.2(0.8)	84.8(1.1)
GAL	SVM	44.7(2.6)	2.7(0.1)	100.0(0.0)	96.7(0.0)	96.5(2.3)	99.8(0.4)	86.6(1.1)
GAL	GB-SVM	50.0(2.9)	3.3(0.4)	92.5(4.3)	97.5(1.4)	88.9(3.9)	95.0(1.8)	84.2(1.5)

D.2 Deep Model Sharing

In Tables 10 and 11, we demonstrate the results of our experiments related to deep model sharing for $M = 2$ and 4 respectively. The results show that sharing the feature extractor across multiple assistance rounds can still outperform the 'Alone' case. Thus, DMS can provide a trade-off between predictive performance and computation space.

Table 10: Results of the MNIST and CIFAR10 ($M = 2$) datasets with CNN model. The MNIST and CIFAR10 are evaluated with Accuracy. GAL_{DMS} represents the results with Deep Model Sharing.

Dataset	MNIST(↑)	CIFAR10(↑)
Interm	99.4(0.0)	81.1(0.3)
Late	99.0(0.0)	81.0(0.2)
Joint	99.4(0.0)	80.1(0.2)
Alone	96.7(0.2)	72.7(0.2)
AL	96.4(0.1)	74.7(0.3)
GAL	98.5(0.2)	**78.7(0.4)**
GAL_{DMS}	**98.7(0.1)**	74.3(0.5)

Table 11: Results of the MNIST and CIFAR10 ($M = 4$) datasets with CNN model. The MNIST and CIFAR10 are evaluated with Accuracy. GAL_{DMS} represents the results with Deep Model Sharing.

Dataset	MNIST(↑)	CIFAR10(↑)
Interm	99.1(0.0)	79.8(0.1)
Late	98.4(0.1)	77.5(0.2)
Joint	99.4(0.0)	80.1(0.2)
Alone	81.2(0.1)	60.0(0.4)
AL	82.5(0.1)	64.8(0.3)
GAL	96.6(0.2)	**77.3(0.2)**
GAL_{DMS}	**96.7(0.3)**	71.3(0.2)

D.3 Comparison with AL

In Table 12, we demonstrate the comparison of computation and communication complexity between GAL and AL. We compare the computation and communication complexity between AL and GAL under the constraint of the same communication cost as demonstrated. Because AL sequentially trains each organization while GAL allows organizations to train locally in parallel, the computation time and communication round of AL is $M\times$ those of GAL. The GAL with Deep Model Sharing (GAL_{DMS}) saves $T\times$ computation space by sharing the feature extractor of deep models. In summary, GAL generalizes the problem scope, reduces the computation and communication complexity, and achieves significantly better results.

Table 12: Comparison of computation and communication complexity between GAL and AL. M and T represent the number of organizations and assistance rounds, respectively.

Complexity	Computation Time	Computation Space	Communication Round	Communication Cost
AL	$M\times$	$T\times$	$M\times$	$1\times$
GAL	$1\times$	$T\times$	$1\times$	$1\times$
GAL_{DMS}	$1\times$	$1\times$	$1\times$	$1\times$

D.4 Ablation Studies

D.4.1 Privacy enhancement

Our learning framework does not require the organizations to share local data, model, and objective function. One potential limitation of our approach is that assisting organizations may infer Alice's information based on shared the pseudo-residuals. Therefore, we suggest to further enhance the privacy by adopting the framework of Differential Privacy (DP) [38] or Interval Privacy (IP) [39]. We use the Laplace mechanism with $\alpha = 1$ for DP and set the number of intervals of IP to be 1. We add a moderate amount of noise to the pseudo-residuals in hindsight. In Tables 13-15, we demonstrate that privacy-enhanced GAL can still outperform the 'Alone' case.

Table 13: Ablation study on the privacy enhancement ($M = 2$). GAL_{DP} and GAL_{IP} represent privacy-enhanced by DP and IP, respectively.

Dataset	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Iris(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)
Alone	46.8(3.5)	4.1(0.7)	100.0(0.0)	92.5(6.0)	93.1(6.4)	98.9(0.6)	79.9(1.0)	96.7(0.2)	72.7(0.2)
GAL_{DP}	52.1(1.1)	3.5(0.2)	66.3(5.4)	83.3(4.1)	88.2(9.3)	93.9(1.2)	81.9(1.4)	95.7(0.4)	59.0(0.9)
GAL_{IP}	52.1(0.9)	3.4(0.1)	100.0(0.0)	92.5(2.8)	99.3(1.2)	96.3(0.7)	86.3(1.3)	97.2(0.4)	71.7(0.4)

Table 14: Ablation study on the privacy enhancement ($M = 4$). GAL_{DP} and GAL_{IP} represent privacy-enhanced by DP and IP, respectively.

Dataset	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Iris(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)
Alone	56.6(8.2)	4.8(0.6)	80.0(6.1)	79.2(13.0)	84.7(1.4)	97.1(1.0)	73.0(1.0)	81.2(0.1)	60.0(0.4)
GAL_{DP}	52.0(0.8)	3.4(0.1)	47.5(14.8)	85.8(6.4)	91.0(6.3)	95.2(1.0)	81.9(3.2)	94.7(0.4)	57.6(0.1)
GAL_{IP}	52.0(0.2)	3.4(0.1)	97.5(4.3)	89.2(3.6)	97.9(1.2)	96.1(0.8)	86.0(1.9)	95.6(0.4)	71.1(0.3)

Table 15: Ablation study on the privacy enhancement (maximal M). GAL_{DP} and GAL_{IP} represent privacy-enhanced by DP and IP, respectively.

Dataset	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)	ModelNet40(↑)	ShapeNet55(↑)	MIMICL(↓)	MIMICM(↑)
Alone	59.7(9.2)	5.8(0.9)	41.3(10.8)	63.9(15.6)	92.5(3.4)	68.8(3.4)	24.2(0.1)	46.3(0.3)	76.4(1.1)	81.3(0.6)	106.1(0.3)	0.78(0.0)
GAL_{DP}	52.2(1.0)	4.3(1.1)	51.3(10.8)	88.2(7.9)	94.7(1.1)	80.5(1.8)	94.3(0.6)	56.8(0.7)	46.6(2.8)	46.7(7.5)	94.9(3.2)	0.59(0.0)
GAL_{IP}	51.8(0.7)	4.2(1.1)	100.0(0.0)	95.8(3.1)	96.1(0.8)	84.8(0.9)	94.7(0.5)	69.2(0.1)	59.8(0.7)	58.0(2.5)	95.5(4.9)	0.59(0.0)

D.4.2 Gradient assistance weights

To optimize gradient assistance weights, we use the Adam optimizer and enforce the parameters to sum to one by using the softmax function. The cost to optimize the gradient assistance weights w and gradient assisted learning rate η is often negligible compared with the cost to fit the pseudo-residuals since the number of parameters involved in $w \in \mathbb{R}^M$ and $\eta \in \mathbb{R}^1$ is small. In Tables 16-18, we demonstrate the results of our ablation studies o gradient assistance weights.

Table 16: Ablation study ($M = 2$) of gradient assistance weights by adding noises to the transmitted pseudo-residuals to half of the organizations.

Noise	Weight	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Iris(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)
$\sigma = 1$	✗	50.1(1.9)	4.4(0.2)	62.5(2.5)	80.8(6.4)	86.8(2.3)	89.9(3.1)	73.2(1.3)	79.7(0.3)	48.8(0.3)
	✓	**47.8(2.4)**	**3.5(0.5)**	**97.5(4.3)**	**95.0(3.7)**	**96.5(3.0)**	**98.7(1.0)**	**80.2(0.5)**	**96.8(0.1)**	**71.4(0.1)**
$\sigma = 5$	✗	58.8(1.3)	6.1(0.2)	25.0(9.4)	52.5(10.9)	63.9(3.4)	73.2(1.0)	63.3(0.5)	34.8(0.5)	22.0(0.2)
	✓	**46.5(3.1)**	**4.1(0.8)**	**83.8(7.4)**	**90.0(4.1)**	**93.1(4.2)**	**97.6(1.1)**	**78.3(1.0)**	**96.3(0.1)**	**65.9(0.3)**

Table 17: Ablation study ($M = 4$) of gradient assistance weights by adding noises to the transmitted pseudo-residuals to half of the organizations.

Noise	Weight	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Iris(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)
$\sigma = 1$	✗	46.7(1.0)	4.1(0.1)	46.3(6.5)	80.0(5.3)	85.4(3.0)	91.2(1.4)	72.6(2.2)	78.7(0.1)	47.6(0.3)
	✓	**45.0(2.8)**	**3.7(0.5)**	**90.0(5.0)**	**95.8(4.3)**	**94.4(3.4)**	**97.8(1.0)**	**79.1(1.1)**	**94.1(0.1)**	**65.4(0.3)**
$\sigma = 5$	✗	59.4(1.1)	5.7(0.4)	13.8(4.1)	54.2(7.6)	61.1(7.1)	75.9(2.9)	64.1(1.8)	38.4(0.3)	22.6(0.5)
	✓	**49.6(3.7)**	**4.1(0.7)**	**66.3(9.6)**	**93.3(2.4)**	**93.7(3.6)**	**97.8(0.4)**	**76.7(1.6)**	**93.0(0.2)**	**59.9(0.6)**

Table 18: Ablation study (maximal M) of gradient assistance weights by adding noises to the transmitted pseudo-residuals to half of the organizations.

Noise	Weight	Diabetes(↓)	BostonHousing(↓)	Blob(↑)	Wine(↑)	BreastCancer(↑)	QSAR(↑)	MNIST(↑)	CIFAR10(↑)	ModelNet40(↑)	ShapeNet55(↑)	MIMICL(↓)	MIMICM(↑)
$\sigma = 1$	✗	49.0(1.6)	4.3(0.2)	46.3(6.5)	81.2(5.3)	90.8(2.5)	73.2(1.0)	75.1(0.4)	45.4(0.3)	55.8(1.0)	67.6(0.8)	105.3(0.6)	0.54(0.0)
	✓	**46.4(2.3)**	**4.0(0.2)**	**78.8(8.2)**	**88.9(2.0)**	**96.7(1.0)**	**78.9(1.2)**	**92.7(0.1)**	**61.0(0.4)**	**78.3(0.4)**	**80.7(0.3)**	**100.1(0.6)**	**0.75(0.0)**
$\sigma = 5$	✗	61.0(2.4)	5.8(0.2)	12.5(2.5)	54.2(6.9)	78.5(2.0)	61.8(0.5)	33.8(0.3)	23.3(0.6)	24.5(0.9)	39.6(0.9)	124.5(0.0)	0.52(0.0)
	✓	**49.7(3.1)**	**4.7(0.5)**	**62.5(9.0)**	**84.7(1.4)**	**96.9(1.3)**	**77.1(0.8)**	**92.1(0.2)**	**57.3(0.3)**	**77.5(0.4)**	**79.8(0.4)**	**108.3(3.5)**	**0.67(0.0)**

D.5 Additional Results

In Figure 7-19, we illustrate results of all datasets (maximal M).

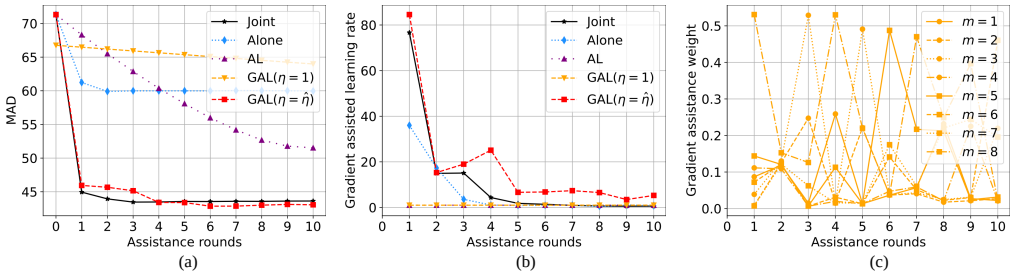


Figure 7: Results of the Diabetes ($M = 8$) dataset.

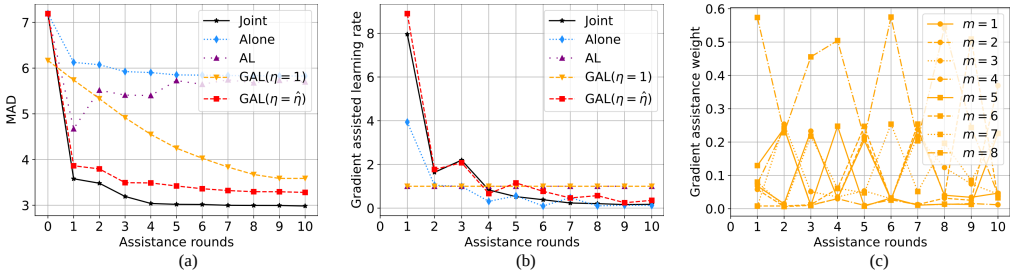


Figure 8: Results of the BostonHousing ($M = 8$) dataset.

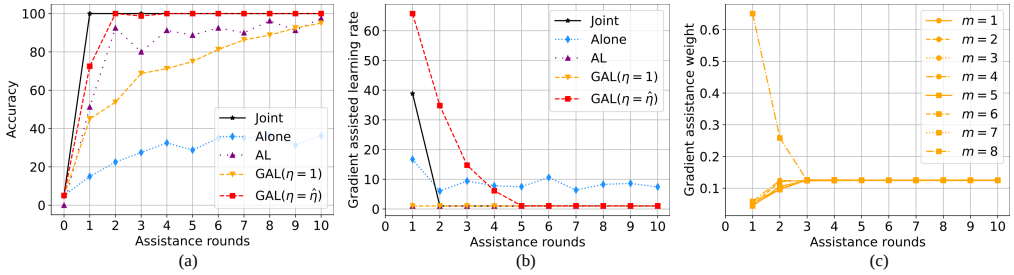
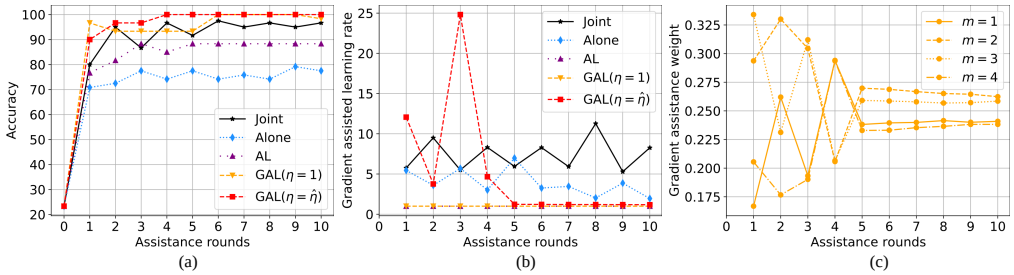
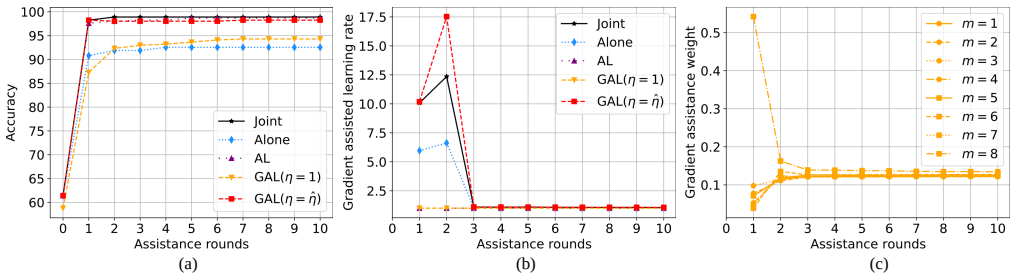


Figure 9: Results of the Blob ($M = 8$) dataset.



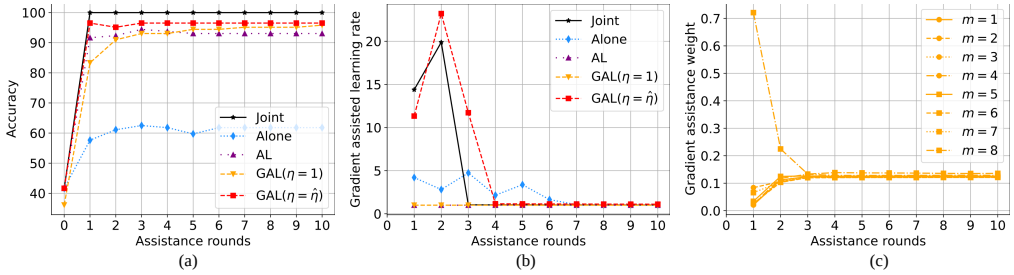
(a) (b) (c)

Figure 10: Results of the Iris ($M = 4$) dataset.



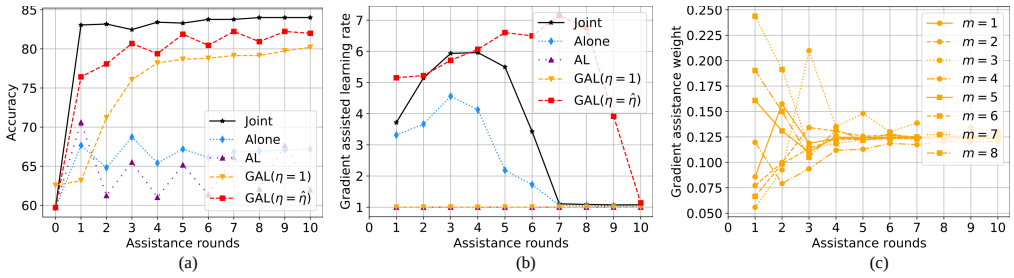
(a) (b) (c)

Figure 11: Results of the BreastCancer ($M = 8$) dataset.



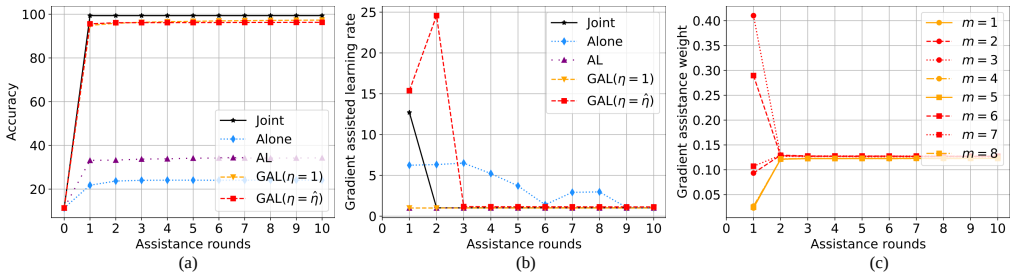
(a) (b) (c)

Figure 12: Results of the Wine ($M = 8$) dataset.



(a) (b) (c)

Figure 13: Results of the QSAR ($M = 8$) dataset.



(a) (b) (c)

Figure 14: Results of the MNIST ($M = 8$) dataset.

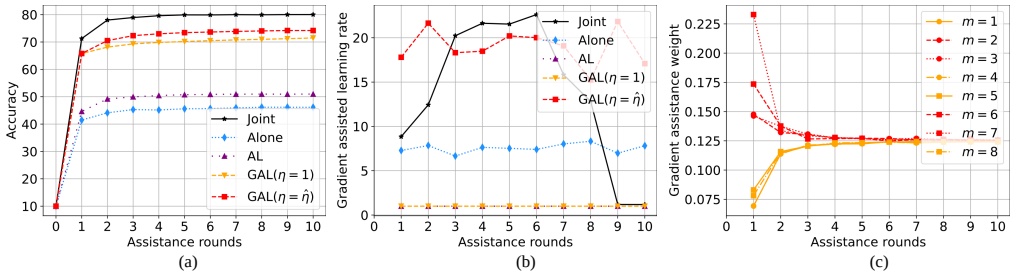


Figure 15: Results of the CIFAR10 ($M = 8$) dataset.

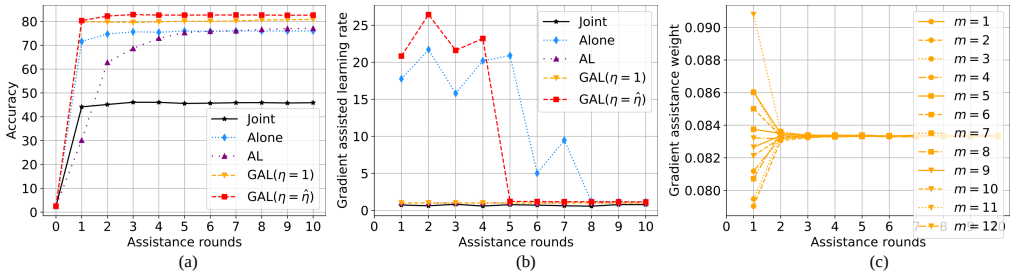


Figure 16: Results of the ModelNet40 ($M = 12$) dataset.

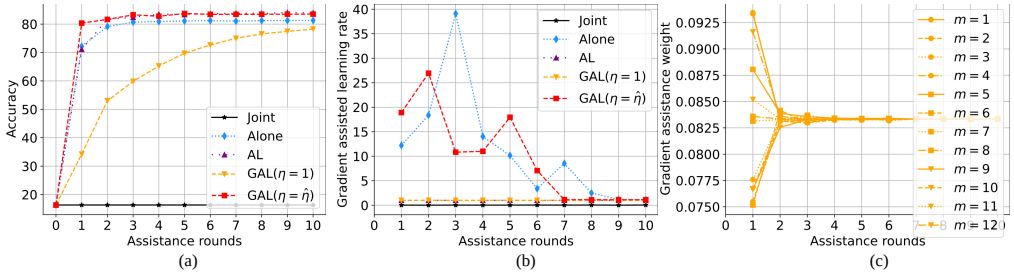


Figure 17: Results of the ShapeNet55 ($M = 12$) dataset.

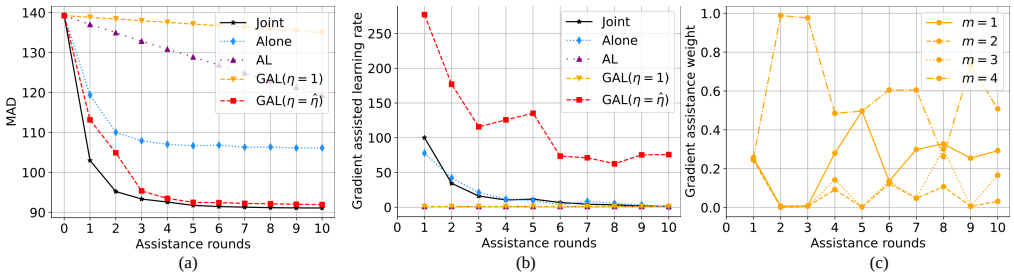


Figure 18: Results of the MIMICL ($M = 4$) dataset.

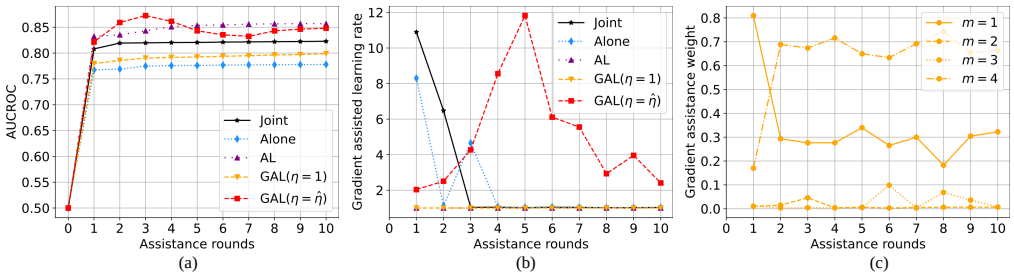


Figure 19: Results of the MIMICM ($M = 4$) dataset.