# Assisted Learning for Organizations with Limited Imbalanced Data

**Anonymous authors**
**Paper under double-blind review**

## Abstract

In the era of big data, many big organizations are integrating machine learning into their work pipelines to facilitate data analysis. However, the performance of their trained models is often restricted by limited and imbalanced data available to them. In this work, we develop an assisted learning framework for assisting organizations to improve their learning performance. The organizations have sufficient computation resources but are subject to stringent data-sharing and collaboration policies. Their limited imbalanced data often cause biased inference and sub-optimal decision-making. In assisted learning, an organizational learner purchases assistance service from an external service provider and aims to enhance its model performance within only a few assistance rounds. We develop effective stochastic training algorithms for both assisted deep learning and assisted reinforcement learning. Different from existing distributed algorithms that need to transmit gradients or models frequently, our framework allows the learner to only occasionally share information with the service provider, but still, obtain a model that achieves near-oracle performance as if all the data were centralized.

## 1 Introduction

Over the past decade, machine learning has demonstrated its great success in various engineering and science domains, e.g., robotic control (Kober & Peters, 2014; Deisenroth et al., 2013), natural language processing (Li et al., 2016; Bahdanau et al., 2017), computer vision (Liu et al., 2017; Brunner et al., 2017), finance (Lee et al., 2020; Lussange et al., 2021; Koratamaddi et al., 2021). Following this trend, many organizations, e.g., governments, hospitals, schools, and companies, are integrating machine learning models into their work pipelines to facilitate data analysis and improve decision-making. For example, according to a recent survey (Financesonline, 2021), about 49% of companies worldwide are considering to use machine learning, 51% of organizations claim to be early adopters of machine learning, and the estimated productivity improvement obtained by learning models can be as high as 40%.

Although machine learning techniques have been standardized and are relatively easy for organizations to implement with real-world data, the model performance critically depends on the quality of the training data they hold (Goodfellow et al., 2016). However, these training data may be of limited size or biased toward certain distributions. For example, for local banks collecting financial data from economic activities, the sample size is often limited by the size of the local population, and the data distribution is affected by the local economy. Consequently, machine learning models trained on such limited and imbalanced data may generalize poorly on test data. Therefore, there is a need to develop a modern machine learning framework that can assist organizations in improving their model performance.

A natural solution is to connect the organization to an external consulting agent, who has machine learning expertise and holds a large amount of data that is balanced or complementary to the organization's imbalanced data. For example, for a local small hospital, the consulting agent may be a large medical center at the state or national level. In particular, the organization can purchase some generic assistance service (e.g., information exchange) from the consulting agent to help improve its model performance. However, organizations' interactions with external consulting agents are usually subject to stringent regulations and policies. Two

common restrictions are: (i) neither the organization nor the consulting agent wants to share their raw data with the other side; and (ii) the organization often has a limited budget and can purchase very limited services from the agent. Considering these challenges for organizational machine learning users, it is desired to develop a machine learning framework for organizations to significantly improve their model performance by purchasing limited assistance services from external agents without data sharing. This constitutes the overarching goal of this work.

In this work, we develop an assisted learning framework in the 'horizontal-splitting' setting, where the organization and the consulting agent possess different sets of data samples that are utilized for training a common model. In our context, the organization's data are assumed to be limited and imbalanced, while the consulting agent's data are large or complement the organization's data. Our assisted learning framework is inspired by organizations' common characteristics: they may have a limited budget for the rounds of purchasing external assistance, yet they can exchange a large amount of side information per assistance round to maximize the performance gain. We summarize our contributions as follows.

### 1.1 Our Contributions

We formally define an assisted learning framework with 'horizontal-splitting' data. In this framework, an organization (referred to as the learner) is connected to an external service provider (referred to as the provider) for assistance, and their interaction is subject to the following rules: (i) Neither the learner nor the provider can share data with each other; (ii) The learner can interact with the provider for only a few assistance rounds due to limited budget, and it aims to maximize the performance gain; and (iii) In each assistance round, both the learner and the provider have access to sufficient computation and communication resources.

We then develop a fully-decentralized assisted learning algorithm named AssistDeep for *deep learning* tasks. This algorithm enables the learner to effectively interact with the provider without sharing data. Specifically, every assistance round of AssistDeep consists of two phases. In the first phase, the learner performs local training for multiple iterations and sends the generated trajectory of models together with their corresponding local loss values to the provider. In the second phase, the provider utilizes the learner's information to evaluate the global loss of the received models and picks the model that achieves the minimum global loss as the initialization. Then, the provider performs local training for multiple iterations and sends the generated trajectory of models together with their corresponding local loss values to the learner. Finally, the learner utilizes the provider's information to evaluate the global loss of the received models and outputs the best model that achieves the minimum global loss. Moreover, for *reinforcement learning* tasks, we further propose a fully-decentralized assisted learning algorithm named AssistPG, which is based on the policy gradient algorithm and has the same training logic as that of AssistDeep.

Through extensive experiments on deep learning and reinforcement learning tasks, we demonstrate that the learner can achieve a near-oracle performance with AssistDeep and AssistPG as if the model was trained on centralized data (i.e., the learner has full access to the provider's raw data). In particular, as the learner data's level of imbalance increases, AssistDeep can help the learner achieve a higher performance gain. Moreover, data are never exchanged in the assisted learning process for both participants.

### 1.2 Related Work

**Assisted learning**. The concept of assisted learning was originally proposed by the earlier work (Xian et al., 2020). However, they consider a very different assisted learning setting, i.e., the 'vertical-splitting' setting in which the organization and the consulting agent collect different features from the same cohort. This is in contrast with our setting where they hold the same features but different data samples. Moreover, their assisted learning algorithm was specially developed for regression-type tasks, whereas our algorithms apply to both classification and reinforcement learning tasks with general deep models. Consequently, our algorithm designs and application scenarios are substantially different from the prior work.

**Distributed learning**. Distributed learning frameworks such as federated learning (Shokri & Shmatikov, 2015; Konecny et al., 2016; McMahan et al., 2017) aim to improve the learning performance for a large number

of learners that have limited data and computation/communication resources. These learning frameworks are well suited for cloud systems and IoT systems (Ray, 2016; Gomathi et al., 2018) that manage numerous smart devices through wireless communication. In conventional distributed optimization, the data is evenly distributed among workers, which collaboratively solves a large-scale problem by frequently exchanging local information (gradients, models.) via either decentralized networks (Xie et al., 2016; Lian et al., 2017; 2018) or centralized networks (Ho et al., 2013; Li et al., 2014; Richtarik & Takavc, 2016; Zhou et al., 2016; 2018). As a comparison, our assist learning framework requires only a few interaction rounds between the learner and provider. This is particularly appealing for organizational learners, who can employ a sophisticated optimization process locally while restricting the rounds of assistance.

## 2 Assisted Deep Learning

In this section, we introduce the assisted learning framework for deep learning tasks. Throughout the paper, L denotes a learner who seeks assistance, and P denotes a service provider who provides assistance to L.

### 2.1 Problem Formulation

We consider the case where the learner L aims to train a machine learning model $\theta$ that performs well on its own dataset $\mathcal{D}^{(\mathrm{L})}$ and generalizes well to unseen data. In general, L can train a machine learning model by solving the empirical risk minimization problem $\min_{\theta\in\Theta} f(\theta;\mathcal{D}^{(\mathrm{L})})$, where $f(\cdot;\mathcal{D}^{(\mathrm{L})})$ is the loss on $\mathcal{D}^{(\mathrm{L})}$ and $\Theta$ is the parameter space. Standard statistical learning theories show that the obtained model can generalize well to intact test samples under suitable constraints of model parsimoniousness (Ding et al., 2018). However, when the user's data $\mathcal{D}^{(\mathrm{L})}$ contains a **limited** number of samples that are highly **imbalanced**, the learned model will suffer from overfitting or deteriorated generalization capability to the unseen test data.

To overcome data deficiency, the learner L intends to connect with an external service provider P (e.g., a commercialized data company), who possesses data $\mathcal{D}^{(\mathrm{P})}$ that are sufficient or complementary to the learner's data $\mathcal{D}^{(\mathrm{L})}$. Ideally, the user L would improve the model by solving the following data-augmented problem, where $\mathcal{D}^{(\mathrm{L,P})} := \mathcal{D}^{(\mathrm{L})} \cup \mathcal{D}^{(\mathrm{P})}$ denotes the centralized data.

$$\theta^{(\mathrm{L,P})} = \arg\min_{\theta\in\Theta} f(\theta;\mathcal{D}^{(\mathrm{L,P})}). \tag{1}$$

We note that $f(\theta;\mathcal{D}^{(\mathrm{L,P})}) = f(\theta,\mathcal{D}^{(\mathrm{L})}) + f(\theta,\mathcal{D}^{(\mathrm{P})})$. If $\mathcal{D}^{(\mathrm{P})}$ is generated from a distribution that is close to the underlying data distribution, it is expected that $\theta^{(\mathrm{L,P})}$ will achieve significantly improved performance on unseen data. However, it is unrealistic to centralize the data since the interactions between the learner L and the provider P are often restricted by stringent regulations. More specifically, in the assisted learning framework, we consider the following protocols listed below.

**Assisted Learning Protocols**

1. *No data sharing:* Neither the learner L nor the provider P will share data with each other.
2. *Limited assistance:* The learner L has a limited budget for purchasing assistance services. The learner desires to maximize the performance gain within only a few assistance rounds.
3. *Unlimited computation and communication:* In each assistance round, both the learner and the provider can perform unlimited computation and exchange unlimited information.

To elaborate on the above protocols, first note that data sharing is often considered sensitive and prohibited in modern distributed learning. Also, assistance service between organizations is usually costly and time-consuming in reality, and organizational learners usually have a limited budget to purchase and manage such service. Moreover, we generally assume that both the organizational learner and the service provider have sufficient computation resources, and they can exchange unlimited information in each interaction round, e.g., the learner (resp. provider) can send an employee (resp. technician) to deliver a large-capacity hard drive to the other side.

**Algorithm 1** AssistDeep

**Input:** Initialization model $\theta^0$, learning rate $\eta$, assistance rounds $R$, local iterations $T$
**for** assistance rounds $r = 1, \ldots, R$ **do**

**Learner L :**
- ▶ Initialize $\theta_0^{(\mathrm{L})} = \theta^{r-1}$
- ▶ Local training to generate $\{\theta_t^{(\mathrm{L})}\}_{t=0}^{T-1}$
- ▶ Send $\{\theta_t^{(\mathrm{L})}, f(\theta_t^{(\mathrm{L})}; \mathcal{D}^{(\mathrm{L})})\}_{t\in\mathcal{T}}$ to provider P

**Provider P :**
- ▶ Initialize $\theta_0^{(\mathrm{P})} = \arg\min_{\theta\in\{\theta_t^{(\mathrm{L})}\}_{t\in\mathcal{T}}} f(\theta; \mathcal{D}^{(\mathrm{L,P})})$
- ▶ Local training to generate $\{\theta_t^{(\mathrm{P})}\}_{t=0}^{T'-1}$
- ▶ Send $\{\theta_t^{(\mathrm{P})}, f(\theta_t^{(\mathrm{P})}; \mathcal{D}^{(\mathrm{P})})\}_{t\in\mathcal{T}'}$ to learner L

**Learner L :**
- ▶ Output $\theta^r = \arg\min_{\theta\in\{\theta_t^{(\mathrm{P})}\}_{t\in\mathcal{T}'}} f(\theta; \mathcal{D}^{(\mathrm{L,P})})$

**end**
**Output:** The best model in $\{\theta^r\}_{r=1}^{R}$

Conventional distributed learning algorithms cannot be applied to enable assisted learning, as they are developed for distributed systems involving a large number of agents having access to very limited computation and communication resources, and they often require frequent information exchange among the agents. Hence, we need to develop a training algorithm specifically for assisted learning that can substantially improve the learner's model performance via limited interactions with the service provider. Next, we present a generic assisted learning algorithm for solving deep learning tasks.

### 2.2 AssistDeep for Assisted Deep Learning

We propose *AssistDeep* in Algorithm 1 for assisted deep learning. The learning process consists of $R$ rounds, each consisting of the following interactions between the learner L and the provider P.

(1) First, the learner L initiates a local learning process. It initializes a model $\theta_0^{(\mathrm{L})}$ and applies any standard deep learning optimizer (e.g., SGD, Adam, etc.) with learning rate $\eta$ to update it for $T$ iterations using the local dataset $\mathcal{D}^{(\mathrm{L})}$. Then, the learner evaluates the local loss $f(\cdot; \mathcal{D}^{(\mathrm{L})})$ in a subset $\mathcal{T}$ of the iterations $t = 0, 1, ...T-1$. Lastly, the learner sends this subset of models and their corresponding local loss to the provider P .

(2) Upon receiving the information from the learner L, the provider P first evaluates the global loss $f(\cdot; \mathcal{D}^{(\mathrm{L,P})})$ of the received set of models $\{\theta_t^{(\mathrm{L})}, t \in \mathcal{T}\}$ and picks the one that achieves the minimum global loss as the initialization model $\theta_0^{(\mathrm{P})}$. Note that the global loss can be evaluated because the local loss $\{f(\theta_t^{(\mathrm{L})}; \mathcal{D}^{(\mathrm{L})}), t \in \mathcal{T}\}$ are provided by the learner L , and the provider P just needs to evaluate the local loss $\{f(\theta_t^{(\mathrm{L})}; \mathcal{D}^{(\mathrm{P})}), t \in \mathcal{T}\}$ on its local data. After that, the provider applies any standard optimizer with learning rate $\eta$ to update the model for $T'$ iterations on the local dataset $\mathcal{D}^{(\mathrm{P})}$. Then, the provider evaluates the local loss $f(\cdot; \mathcal{D}^{(\mathrm{P})})$ in a subset $\mathcal{T}'$ of the iterations $t = 0, 1, ...T'-1$, and sends the subset of models and their corresponding local loss to the learner L.

(3) Once the learner L receives the information sent by the provider P, it evaluates the global loss $f(\cdot; \mathcal{D}^{(\mathrm{L,P})})$ of the received set of models $\{\theta_t^{(\mathrm{P})}, t \in \mathcal{T}'\}$ and picks the one that achieves the minimum global loss as the output model of this assistance round.

**Discussions**. The above assisted learning algorithm works for general deep learning tasks. It does not require data sharing between the learner and the provider. In particular, the interaction between the learner and the provider has the following two prominent features.

- Both the learner and the provider need to store a number of models sampled from the trajectory of models generated in their local training process and evaluate the corresponding local loss value of these sampled models. Then, these sampled models and their local loss values are sent to the other side. Normally, this information exchange may require a large amount of communication. But as we show later in the experimental studies, it suffices to sample the training trajectory at a low frequency.
- After receiving the models and loss values sent by the other side, both the learner and the provider will evaluate the global loss of these models on the centralized data, and then will pick the one that achieves the minimum global loss as the initialization/output model. Here, the global loss of these models can be evaluated on one side, as the loss values of the models on the other side's local data are evaluated and sent by the other side.

## 3 Assisted Reinforcement Learning

We further extend our assisted learning framework to Reinforcement Learning (RL) scenarios to enhance the model's generalizability. We first introduce some basic setups of RL.

**Markov Decision Process (MDP)**. We consider a standard finite-horizon MDP that is denoted by a tuple $M = (\mathcal{S}, \mathcal{A}, \mathbf{P}, r, \pi, \rho_0, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ corresponds to the action space, $\mathbf{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the underlying state transition kernel that drives the new state given the previous state and action, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\pi : \mathcal{S} \to \mathcal{A}$ is the policy, $\rho_0$ denotes the initial state distribution, and $T$ is the episode length. Given a policy $\pi_\theta$ parameterized with $\theta$, the goal of RL, also known as on-policy learning, is to learn an optimal policy parameter $\theta^*$ that maximizes the expected accumulated reward, namely $\theta^* = \arg\max_\theta J(\theta) := \mathbb{E}[\sum_{t=1}^T \gamma^{t-1} r_t]$.

### 3.1 Problem Formulation

We assume that an RL learner L has collected a small amount of Markovian data $\mathcal{D}^{(\mathrm{L})}$ by interacting with a certain environment. It wants to train a policy that generalizes well to other similar environments. However, the data and environment that the learner L can access are limited. In assisted reinforcement learning, the learner L aims to enhance its policy's generalizability to unseen environments by querying assistance from a service provider P. For example, autonomous-driving startup companies typically own limited data that are insufficient for training good autonomous driving models that perform well in heterogeneous environments, and they can purchase assistance services from big companies (who own massive data) to improve the model performance and generalizability.

Formally, we assume that there is an underlying distribution of transition kernel that models the variability of the environment. Specifically, denote $E_\beta$ as an environment with the transition kernel $\mathbf{P}_\beta$ parameterized by $\beta$, which follows an underlying distribution $q$. Let $J_\beta(\theta)$ denote the expected accumulated reward collected from the environment $E_\beta$ following the policy $\pi_\theta$. The learner L's ultimate goal is to learn a good policy that applies to the underlying distribution of environment, namely, $\max_\theta \mathbb{E}_{\beta\sim q}\big[J_\beta(\theta)\big]$. In practice, the learner L only has training data collected from a limited number of environment instances, say $\beta^{(\mathrm{L})} = \{\beta_1^{(\mathrm{L})}, \ldots, \beta_{n_\mathrm{L}}^{(\mathrm{L})}\}$. On the other hand, the service provider may have rich experience interacting with a more diverse set of environments, say $\beta^{(\mathrm{P})} = \{\beta_1^{(\mathrm{P})}, \ldots, \beta_{n_\mathrm{P}}^{(\mathrm{P})}\}$. Consequently, the learner aims to solve the following RL problem with centralized data.

$$\max_\theta J_{\beta^{(\mathrm{L},\mathrm{P})}}(\theta) := \sum_{\beta\in\beta^{(\mathrm{L})}} J_\beta(\theta) + \sum_{\beta'\in\beta^{(\mathrm{P})}} J_{\beta'}(\theta). \tag{2}$$

### 3.2 AssistPG for Assisted Reinforcement Learning

Policy gradient (PG) is a classic RL algorithm for policy optimization. The PG algorithm estimates the policy gradient $\nabla J(\theta)$ via the policy gradient theorem, and applies it to update the policy. Specifically, given one episode $\tau$ with length $T$ that is collected under the current policy $\pi_\theta$, the corresponding policy gradient takes the following form, where $R(\tau) = \sum_{t=1}^T \gamma^{t-1} r_t$ is the discounted accumulated reward over this episode.

In practice, a mini-batch of episodes is used to estimate the policy gradient (Sutton & Barto, 2018), which is approximated as follows.

$$\nabla J(\theta) \approx R(\tau) \sum_{t=1}^{T} \nabla \log \pi_\theta\big(a_t^{(i)} | s_t^{(i)}\big).$$

**Algorithm 2** AssistPG

**Input:** Initialization model $\theta^0$, learning rate $\eta$, assistance rounds $R$, local iterations $T$.
**for** assistance rounds $r = 1, \ldots, R$ **do**

**Learner L :**
- ▶ Initialize $\theta_0^{(\mathrm{L})} = \theta^{r-1}$
- ▶ Local PG training to generate $\{\theta_t^{(\mathrm{L})}\}_{t=0}^{T-1}$
- ▶ Send $\{\theta_t^{(\mathrm{L})}, \sum_{\beta \in \beta^{(\mathrm{L})}} J_\beta(\theta_t^{(\mathrm{L})})\}_{t \in \mathcal{T}}$ to provider P

**Provider P :**
- ▶ Initialize $\theta_0^{(\mathrm{P})} = \arg\max_{\theta \in \{\theta_t^{(\mathrm{L})}\}_{t \in \mathcal{T}}} J_{\beta^{(\mathrm{L,P})}}(\theta)$
- ▶ Local PG training to generate $\{\theta_t^{(\mathrm{P})}\}_{t=0}^{T'-1}$
- ▶ Send $\{\theta_t^{(\mathrm{P})}, \sum_{\beta \in \beta^{(\mathrm{P})}} J_\beta(\theta_t^{(\mathrm{P})})\}_{t \in \mathcal{T}'}$ to learner L

**Learner L :**
- ▶ Output $\theta^r = \arg\max_{\theta \in \{\theta_t^{(\mathrm{P})}\}_{t \in \mathcal{T}'}} J_{\beta^{(\mathrm{L,P})}}(\theta)$

**end**
**Output:** The best model in $\{\theta^r\}_{r=1}^{R}$

In Algorithm 2, we present **Assisted Policy Gradient (AssistPG)**–a policy gradient-type algorithm for solving the assisted RL problem in Equation (2). The main logic of the AssistPG algorithm is the same as that of the AssistDeep for assisted deep learning.

## 4 Experiments

In this section, we first visualize AssistDeep training to help understand the mechanism of assisted learning. Then, we provide extensive experiments of deep learning and reinforcement learning to demonstrate the effectiveness of the proposed assisted learning algorithms.

### 4.1 Visualization of AssistDeep Training

**Regression Example.** We apply AssistDeep to solve a regression problem with simulated data $a = [-1, -1]$, $b = [1, -1.25]$ and hyperparameters $T = 10$, $\eta = 0.9^r$ for both the learner and the provider. We run the algorithm for $R = 10$ assistance rounds. Fig. 1 shows the learning trajectory of $\theta^r$ for $r = 0, 1, \ldots, 9$. It can be seen that at the beginning, the learner L's learning trajectory moves toward the oracle solution since the directions of two local optima are roughly the same; then, it oscillates in between two opposite directions and converges to the oracle solution.

**Classification Example.** We apply AssistDeep to solve a simple logistic regression problem for binary classification. We generate two classes of data samples: Class A data contains 50 points drawn from $\mathcal{N}([-1, 1], 1.5^2 I_2)$ , and Class B data contains 50 points drawn from $\mathcal{N}([1, -1], 1.5^2 I_2)$, where $\mathcal{N}$ and $I$ denote Gaussian distribution and identity matrix, respectively. Suppose that a learner L observes 90% class A samples and 10% class B samples. Another service provider P observes a similar number of data samples consisting of 10% class A samples and 90% class B samples. The learning process of AssistDeep is illustrated in Fig. 2 (Left), and we also show the oracle solution trained by SGD with centralized data in Fig. 2 (Right).

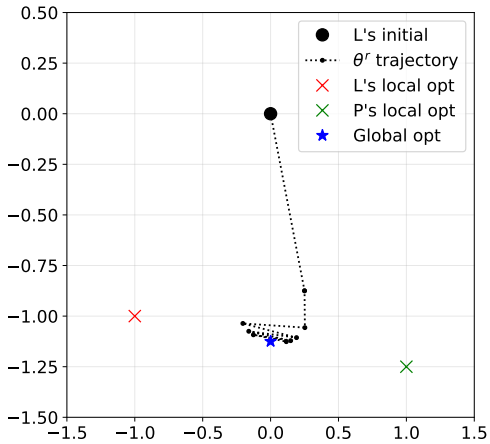


Figure 1: Learning trajectory of AssistDeep in a synthetic regression example.

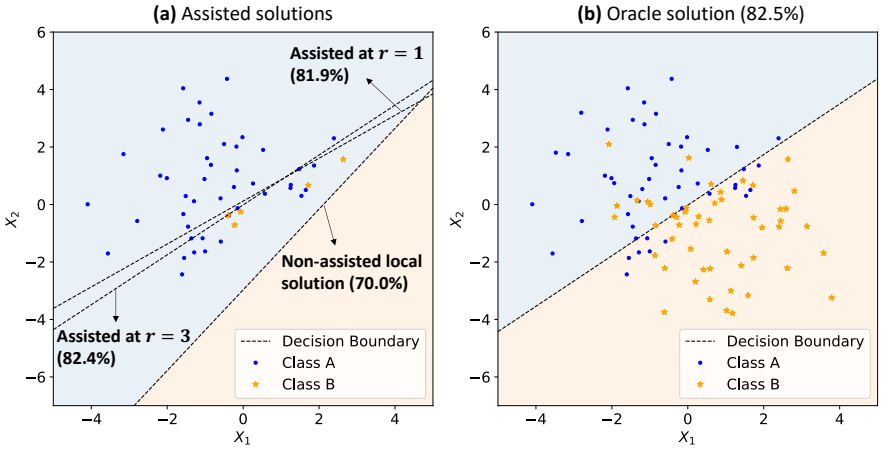


Figure 2: Visualization of AssistDeep in classification: (a) the learner's classifiers after being assisted by the provider at different rounds, and (b) oracle classifier obtained by using centralized data. The test accuracies are shown in the parentheses.

It can be seen that without any assistance, the learner can only achieve 70% accuracy and the corresponding classifier performs poorly on the samples in class B. In comparison, after a single round of assistance, the classification performance is significantly improved to 81.9% accuracy. After three rounds of assistance, the corresponding classifier is relatively close to the oracle classifier, which achieves an accuracy of 82.5%. Hence, it can be seen that AssistDeep has the potential to achieve a near-oracle performance.

### 4.2 Assisted Deep Learning Experiments

We test the performance of AssistDeep by comparing it with two baselines: SGD (using centralized data $\mathcal{D}^{(\mathrm{L,P})}$) and Learner-SGD (using only the learner's data $\mathcal{D}^{(\mathrm{L})}$). We consider two popular datasets CIFAR-10 (Krizhevsky, 2009) and SVHN. We implement all these algorithms using the Adam optimizer with learning rate 0.001 and batch size 256 to train AlexNet and ResNet-18 on CIFAR-10 and SVHN, respectively.

To test AssistDeep, we split the classification dataset into two parts and assign them to the learner and the provider, respectively. Specifically, the provider's data consists of half of the total samples of each classification class, and therefore is balanced and of a large size. On the other hand, the learner's data is sampled from the rest half of the data (excluding the provider's data) and is determined by the data imbalance level parameter $\gamma_L := |\mathcal{D}^{(\mathrm{LMINOR})}|/|\mathcal{D}^{(\mathrm{LMAJOR})}| \in [0, 1]$, where $\mathcal{D}^{(\mathrm{LMAJOR})}$ and $\mathcal{D}^{(\mathrm{LMINOR})}$ denote the major class and minor class of the learner's local data, respectively, and we fix the major class to include the remaining data samples of

the smallest class. Then, all the other classes are minor class of the learner's data and their sizes are given by $\gamma_L|\mathcal{D}^{(\text{LMAJOR})}|$. Intuitively, $\gamma_L = 1$ means that learner's data is balanced and large, whereas $\gamma_L = 0$ means that learner's data is extremely imbalanced and small.

We fix the number of assistance rounds to be 10. The total number of local optimization iterations in each assistance round is fixed to be 2000. We assign these iteration budget to the learner and provider in proportion to their local sample sizes. Both the learner and provider record their local training models and local loss values for every $I = 50$ iteration, which is referred to as the sampling period. In addition, we conducted repeated experiments with different random seeds to estimate the standard deviation of the above results. We find that the training loss and the test accuracy of the output model are highly consistent across the repeated experiments. Specifically, the standard deviation of the training loss and test accuracy are 4e-3 and 3e-3, respectively. Therefore, in all the figures corresponding to deep learning experiments, we only plot the curves using the data obtained from one particular experiment, and the curves of all the other repeated experiments are nearly identical.

#### 4.2.1 Effect of Sample Size and Data Imbalance Level

**CIFAR-10 Dataset.** We first compare these algorithms with balanced learner's data ($\gamma_L = 1$) and imbalanced learner's data ($\gamma_L = 0, 0.3, 0.7$) in training an AlexNet on the CIFAR-10 dataset. In Fig. 3, we plot the training loss (on centralized data $\mathcal{D}^{(\text{L,P})}$) and the test accuracy (on the 10k test data) against the number of assistance rounds. Here, one assistance round on the x-axis is interpreted as 2k local iterations for SGD and Learner-SGD. The training loss of Learner-SGD is not reported as it is trained on $\mathcal{D}^{(\text{L})}$ only. It can be seen that AssistDeep achieves a comparable performance to that of SGD with centralized data. In particular, when $\gamma_L = 0$, meaning that the learner has limited and extremely imbalanced data, the test performance of AssistDeep is significantly better than Learner-SGD, demonstrating the effectiveness of querying assistance from the service provider. When $\gamma_L = 0.3, 0.7, 1$ and the learner has more data, AssistDeep still achieves a near-oracle performance, and its test performance is better than Learner-SGD.

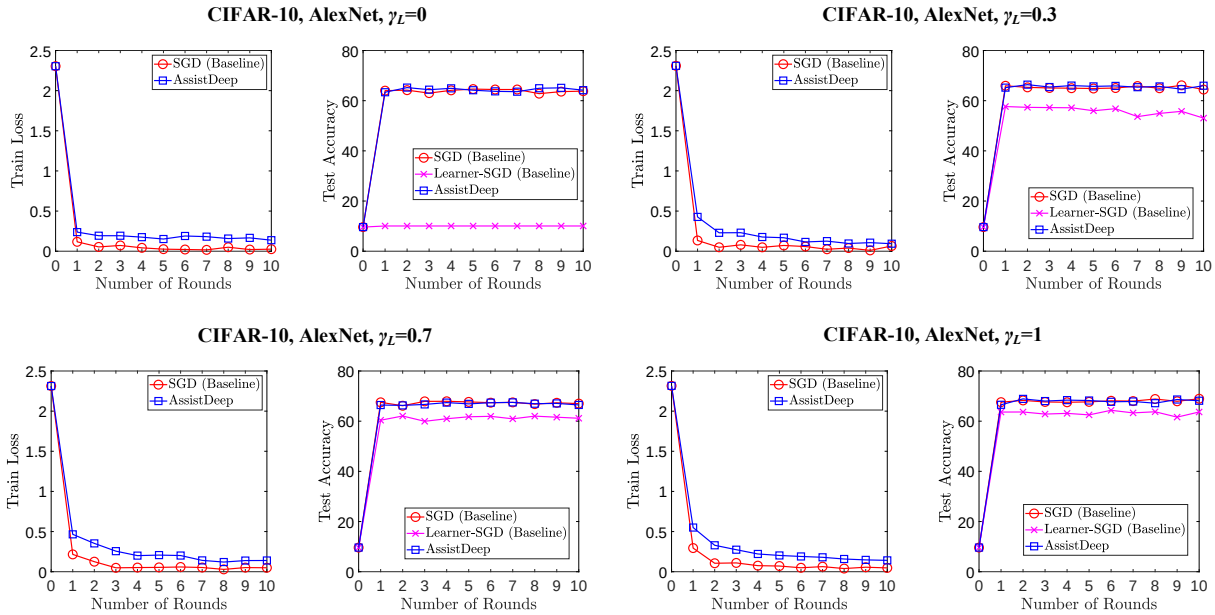


Figure 3: Comparison of AssistDeep, SGD, and Learner-SGD with $\gamma_L = 0, 0.3, 0.7, 1$ in training an AlexNet on CIFAR-10.

We further test and compare these algorithms with balanced learner's data $\gamma_L = 1$ and imbalanced learner's data $\gamma_L = 0, 0.3, 0.7$ in training a ResNet-18 on CIFAR-10. The results on ResNet-18 are shown in Fig. 4. It can be seen that in all the results, AssistDeep achieves a comparable test performance to that of SGD. Also, its test performance is better than Learner-SGD, especially when the learner has limited and imbalanced data

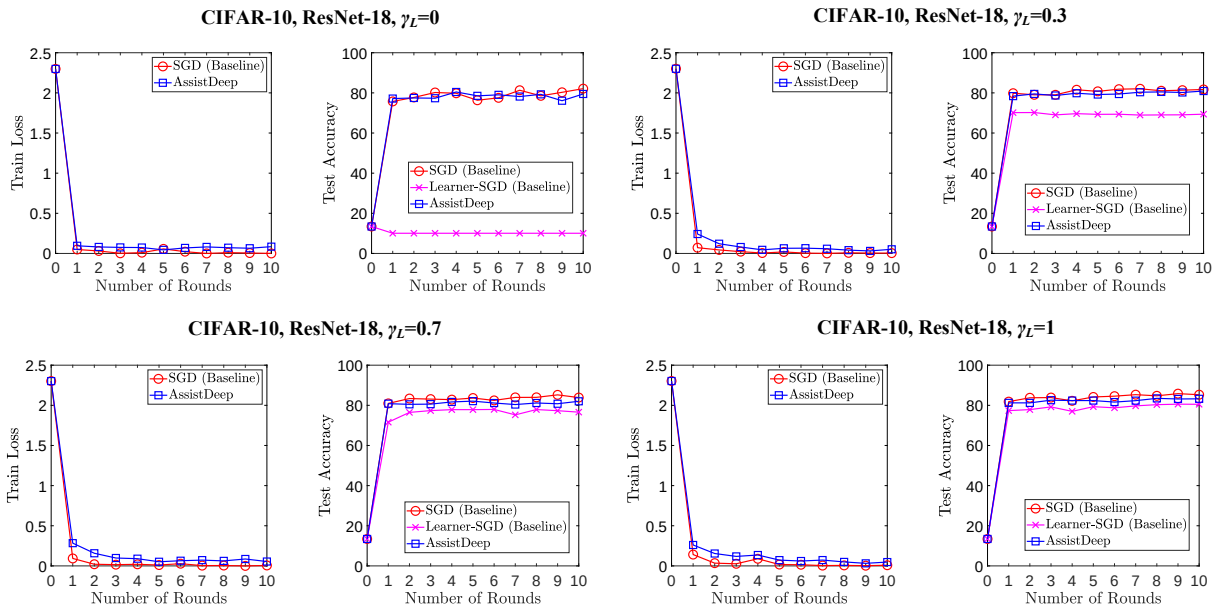


Figure 4: Comparison of AssistDeep, SGD, and Learner-SGD with $\gamma_L = 0, 0.3, 0.7, 1$ in training a ResNet-18 on CIFAR-10.

$(\gamma_L = 0, 0.3, 0.7)$. Combining these results with those in Fig. 3, we conclude that AssistDeep improves the test performance more (compare to Learner-SGD) when the learner's data is more imbalanced and limited.

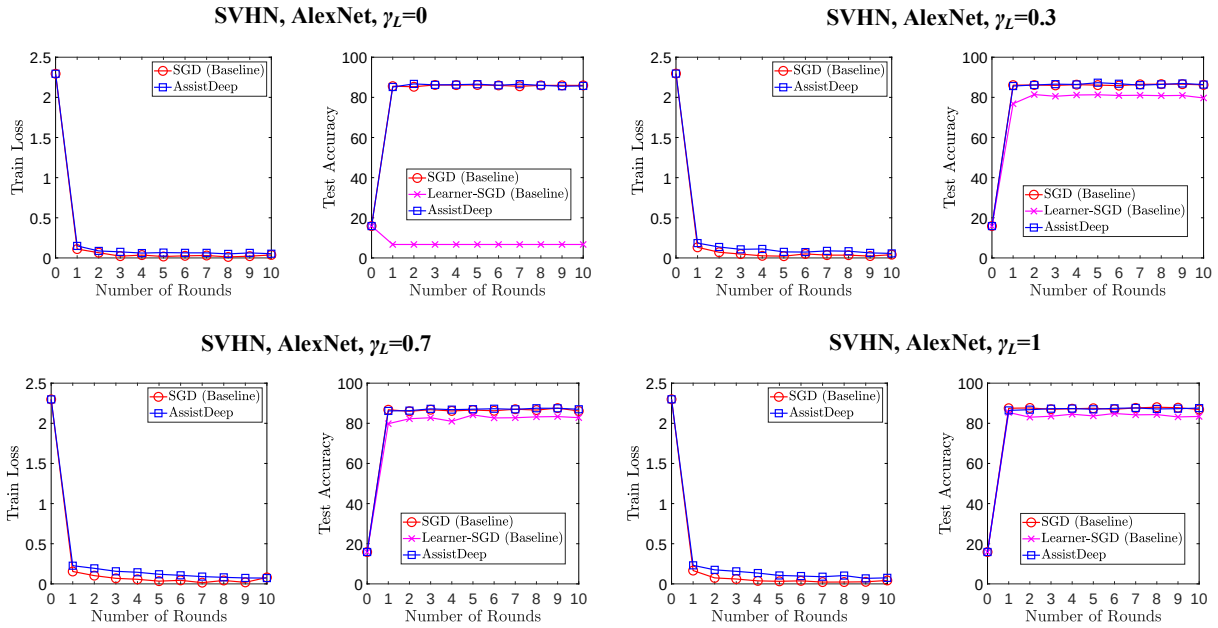


Figure 5: Comparison of AssistDeep, SGD, and Learner-SGD with $\gamma_L = 0, 0.3, 0.7, 1$ in training an AlexNet on SVHN.

**SVHN Dataset.** In Fig. 5 and 6, we repeat the above experiments with a different SVHN dataset using the same hyper-parameter settings. One can make the same observations from these results, which show that our proposed AssistDeep works well on diverse types of datasets.

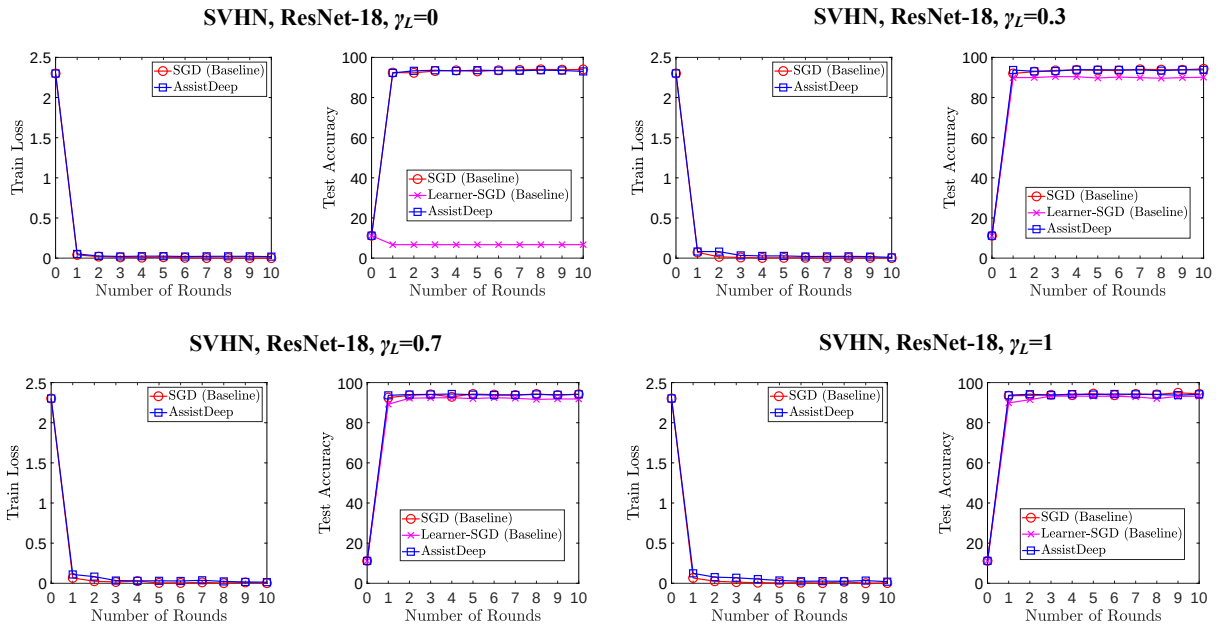


Figure 6: Comparison of AssistDeep, SGD, and Learner-SGD with $\gamma_L = 0, 0.3, 0.7, 1$ in training a ResNet-18 on SVHN.

### 4.2.2 Effect of Sampling Period

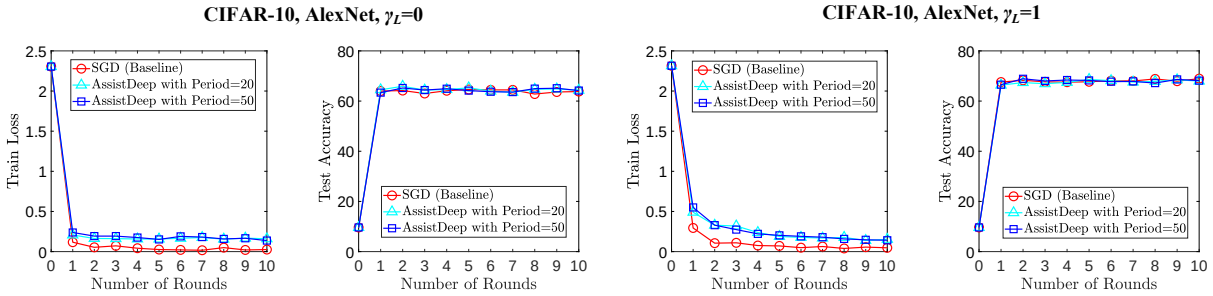


Figure 7: Comparison of AssistDeep and SGD with $\gamma_L = 0, 1$ under different sampling periods in training an AlexNet on CIFAR-10.

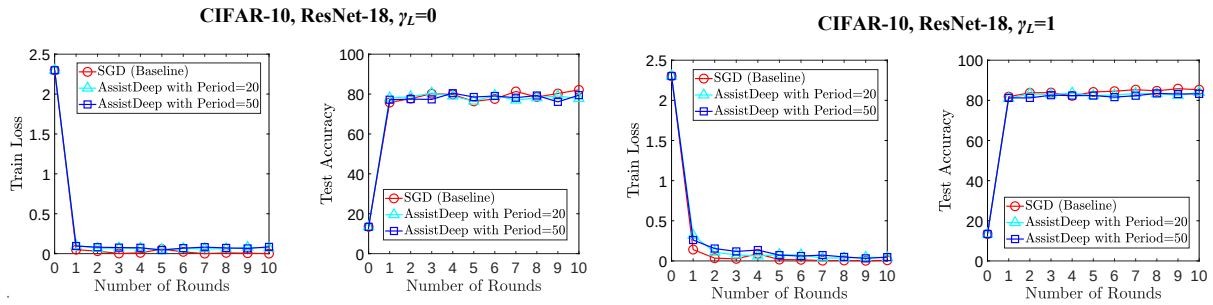


Figure 8: Comparison of AssistDeep and SGD with $\gamma_L = 0, 1$ under different sampling periods in training a ResNet-18 on CIFAR-10.

On CIFAR-10, we explore whether increasing the sampling frequency of the model and loss value can improve the performance of AssistDeep. We consider $\gamma_L = 0, 1$ and compare AssistDeep with centralized SGD under different sampling periods 20 and 50. The comparison results in training AlexNet and ResNet-18 are shown in Fig. 7 and 8, respectively. It can be seen that using a low sampling frequency for AssistDeep already achieves the baseline performance of SGD. It implies that AssistDeep does not require much information exchange between the learner and provider. This helps save computation resources and reduce information leakage.

### 4.3 Assisted Reinforcement Learning Experiments

We demonstrate the effectiveness of AssistPG via two RL applications: CartPole (Barto et al., 1983) and LunarLander (Brockman et al., 2016). In the CartPole problem, a controller aims to stabilize a pole attached to a cart by applying left or right force to the cart (see the first figure in Fig. 9), and we show that AssistPG can help the controller stabilize the pole with different pole lengths. For the LunarLander problem, a lander initializes its landing from top left of the sky and aims to land on a landing pad by controlling its engine (see the first figure in Fig. 10). We show that AssistPG can help land the lander with different engine powers.

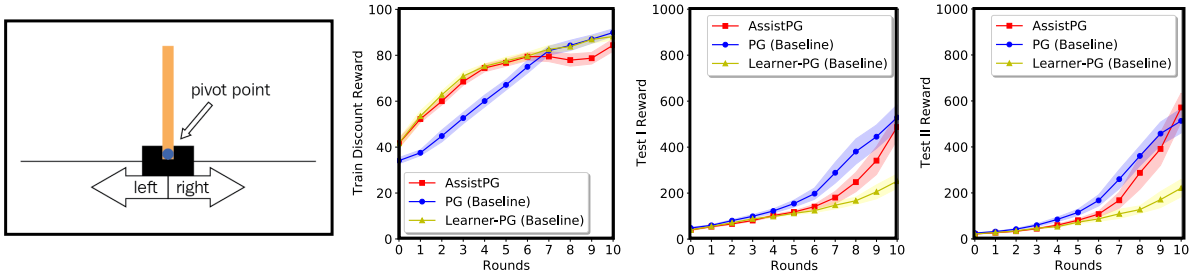


Figure 9: Comparison of AssistPG, PG, and Learner-PG in the CartPole game.

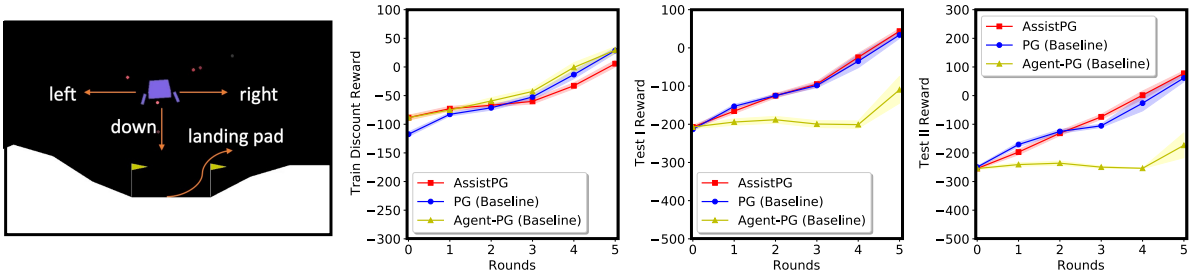


Figure 10: Comparison of AssistPG, PG, and Learner-PG in the LunarLander game.

We assume that the learner and the provider can query episode data by interacting with diverse environments. Specifically, for the CartPole problem, we parameterize the environment using the pole length. Both the learner and the provider train their control policies by playing 5 Cartpole games with the pole length randomly generated from Uniform$(4, 5)$ (for the learner) and Uniform$(0, 1)$ (for the provider). For the LunarLander problem, we parameterize the environment using the engine power. Both the learner and provider train their control policies by playing 10 LunarLander games with the engine power randomly generated from Uniform$(10, 15)$ (for the learner) and Uniform$(35, 40)$ (for the provider). Moreover, we consider two sets of testing environments that are uniform ("Test I") and non-uniform ("Test II"), respectively. For the CartPole problem, Test I environments randomly generate the pole length from Uniform$(0, 5)$, and Test II environments randomly generate the pole length from Beta$(1, 5)$ with probability 0.2 and Uniform$(0, 5)$ with probability 0.8. For the LunarLander problem, Test I environments randomly generate the engine power from Uniform$(10, 40)$, and Test II environments randomly generate the engine power as $30r + 10$, where $r \sim$ Beta$(5, 1)$.

We test AssistPG on both RL problems and compare its performance with two baselines: the standard PG (using centralized episode data), and the Learner-PG (using only learner's episode data). All these algorithms are implemented with learning rate $5 \times 10^{-3}$ and episode batch size 32. We model the policy using a three-layer feed-forward neural network with 4 and 32 hidden neurons for CartPole and LunarLander,

respectively. Moreover, for AssistPG, we fix the total number of assistance rounds to be 10 and 5 for CartPole and LunarLander, respectively. The total number of local PG iterations in each assistance round is fixed to be 20 for both problems. We also set the sampling period to be four, namely, the learner and the provider record their local model and discounted training reward for every four local PG iterations. Fig. 9 and 10 plot the discounted training rewards (collected in local environment only), Test I cumulative rewards, and Test II cumulative rewards against the assistance round obtained by all these algorithms, for solving the CartPole and LunarLander problems, respectively. Here, one assistant round is interpreted as 20 local PG iterations for algorithms other than AssistPG.

Fig. 9 indicates that AssistPG outperforms Learner-PG, when the testing environments include diverse lengths of poles. Also, AssistPG can achieve a comparable performance to that of the PG with centralized data. Moreover, Fig. 10 indicates that AssistPG can swiftly adapt to scenarios out of their comfort zone (namely the training environments) in only a few rounds. These experiments demonstrate that our assisted learning framework can help the learner significantly improve the quality of the policy for handling complex RL problems.

### 4.4 Visualization of LunarLander Experiment

In this section, we visualize the landing trace of the LunarLander trained by AssistPG and Leaner-PG in different test environments.

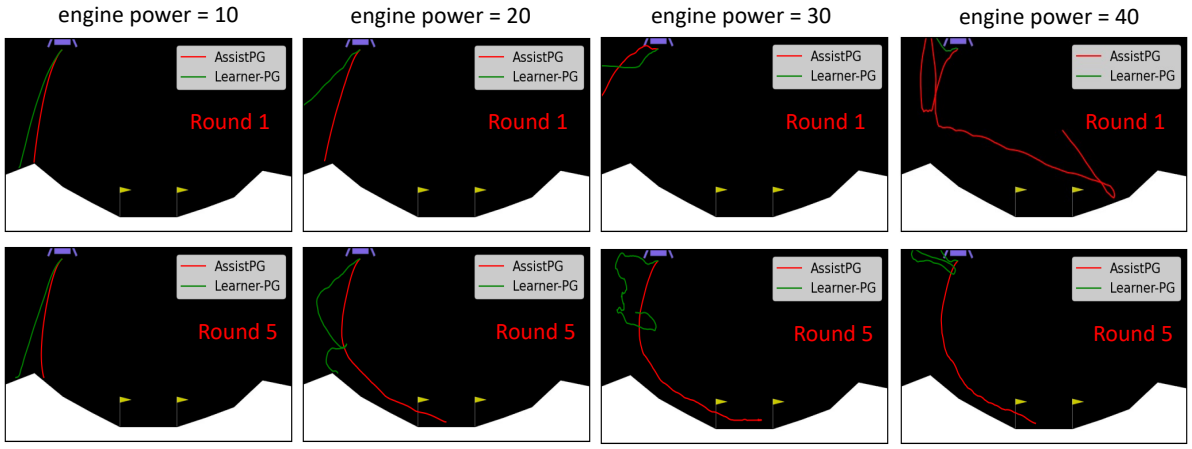


Figure 11: Landing traces of LunarLander with engine power = 10, 20, 30, 40 trained by AssistPG and Learner-PG.

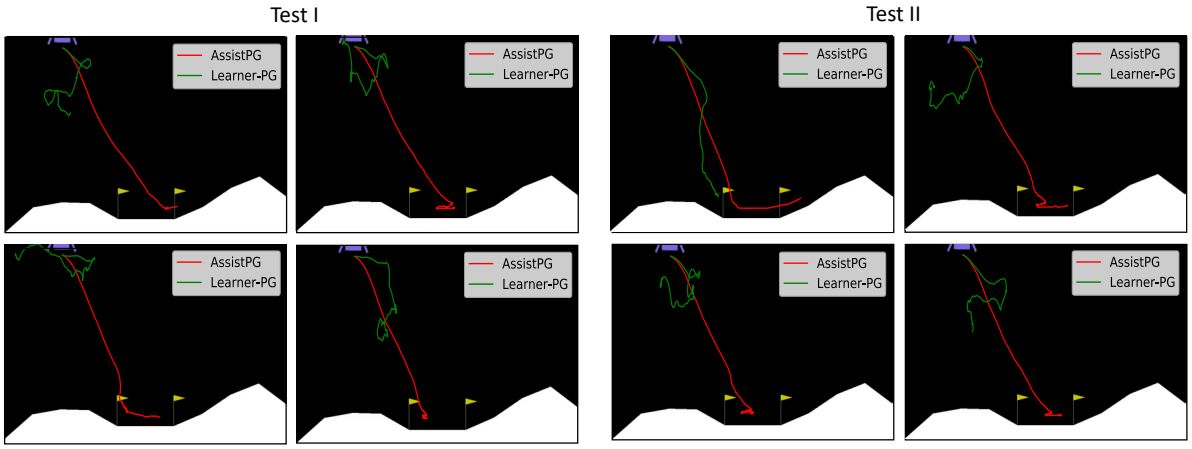


Figure 12: Landing traces with engine power $\sim$ Uniform$(10, 40)$ and $\sim 30 \cdot \text{Beta}(5, 1) + 10$ trained by AssistPG and Learner-PG.

Specifically, we consider a fixed map and set the engine power of the lander to be 10, 20, 30, and 40, respectively. In each setting, we train the lander using both AssistPG and Learner-PG for $R = 5$ rounds. After each round of training, we let the lander play an episode using the trained model and plot the corresponding landing trace. These traces are plotted in Fig. 11. From these figures, it can be seen that the lander with engine power 20-40 trained by AssistPG can successfully land on the landpad after 5 rounds of assisted learning. As a comparison, the lander trained by Learner-PG cannot even land after 5 rounds of training. This demonstrates the advantage of AssistPG. On the other hand, when the lander has a small engine power 10, it is challenging for both algorithms to land the lander properly, as the engine cannot provide sufficient acceleration. Moreover, after 5 rounds of training (using both AssistPG and Learner-PG), we test the lander in both the test environment I ("Test I") and II ("Test II"), and plot the landing traces in Fig. 12. Here, for each test, we consider a fixed map and randomly generate 4 different engine powers from Uniform$(10, 40)$ (for Test I) and $30 \cdot \text{Beta}(5, 1) + 10$ (for Test II).

From both figures, it can be seen that the lander trained by the AssistPG lands more smoothly in all test environments under diverse engine powers than that trained by the Learner-PG. The video version for the CartPole and LunarLander games can be accessed from the anonymous link `https://www.dropbox.com/sh/oz2jswj36li4lkh/AADaQn4Nj67v9mdIHKDLN6nAa?dl=0`. In the CartPole game, four videos record the performance of AssistPG and Learner-PG against the first five rounds with pole length equaling 1, 2, 3, and 4, respectively. Another two videos record 10 plays in the test environment I and II, respectively. In all the plays, both AssistPG and Learner-PG use the model trained from the fifth round. In the LunarLander game, four videos record the performance of AssistPG and Learner-PG against the first five rounds with engine power equaling 10, 20, 30, and 40, respectively. Another two videos record 10 plays in the test environment I and II, respectively. In all the plays, both AssistPG and Learner-PG use the model trained from the fifth round. The videos show that with the assistance from the provider, the user can quickly generalize its model to more diverse environments.

## 5 Conclusion

This work develops a learning framework for assisting organizational learners to improve their learning performance with limited imbalanced data. In particular, the proposed AssistDeep and AssistPG allow the provider to assist the learner's training process and significantly improve its model quality within only a few assistance rounds. We demonstrate the effectiveness of both assisted learning algorithms through experimental studies. In the future, we expect that this learning framework can be integrated with other learning frameworks such as meta-learning and multi-task learning. A limitation of this study is that it only considers a pair of learner and provider. An interesting future direction is to emulate the current assisted learning framework to allow multiple learners or service providers.

## References


Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *Proc. International Conference on Learning Representations*, 2017.

Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Gino Brunner, Oliver Richter, Yuyi Wang, and Roger Wattenhofer. Teaching a machine to read maps with deep reinforcement learning. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI*, 11 2017.

M. P. Deisenroth, G. Neumann, and J. Peters. *A Survey on Policy Search for Robotics*. 2013.

Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. *Proc. ICLR*, 2021.

Enmao Diao, Jie Ding, and Vahid Tarokh. SemiFL: Semi-supervised federated learning for unlabeled clients with alternate training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=1GAjC_FauE`.

Jie Ding, Vahid Tarokh, and Yuhong Yang. Model selection techniques–an overview. *IEEE Signal Process. Mag.*, 35(6):16–34, 2018.

Financesonline. Market share & data analysis. `https://financesonline.com/machine-learning-statistics/`, 2021. Accessed: 2021-01-18.

RM Gomathi, G Hari Satya Krishna, E Brumancia, and Y Mistica Dhas. A survey on iot technologies, evolution and architecture. In *Proc. ICCCSP*, pp. 1–5. IEEE, 2018.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ML via a stale synchronous parallel parameter server. In *Proc. NeurIPS*, pp. 1223–1231. 2013.

Jens Kober and Jan Peters. *Reinforcement Learning in Robotics: A Survey*, pp. 9–67. Springer, 2014.

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.

Prahlad Koratamaddi, Karan Wadhwani, Mridul Gupta, and Dr. Sriram G. Sanjeevi. A multi-agent reinforcement learning approach for stock portfolio allocation. In *ACM IKDD CODS*, pp. 410, 2021.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Jinho Lee, Raehyun Kim, Seok-Won Yi, and Jaewoo Kang. Maps: Multi-agent reinforcement learning-based portfolio management system. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4520–4526, 7 2020.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pp. 1192–1202, 2016.

Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proc. OSDI*, pp. 583–598, 2014.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *Proc. ICLR*, 2020.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Proc. NeurIPS*, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *Proc. ICML*, volume 80, pp. 3043–3052, 2018.

F. Liu, S. Li, L. Zhang, C. Zhou, R. Ye, Y. Wang, and J. Lu. 3dcnn-dqn-rnn: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds. In *Proc. International Conference on Computer Vision (ICCV)*, pp. 5679–5688, 2017.

Johann Lussange, Ivan Lazarevich, Sacha Bourgeois-Gironde, Stefano Palminteri, and Boris Gutkin. Modelling stock markets by multi-agent reinforcement learning. *Computational Economics*, 57, 01 2021.

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*, pp. 1273–1282. PMLR, 2017.

Partha Pratim Ray. A survey of iot cloud platforms. *Future Computing and Informatics Journal*, 1(1-2): 35–46, 2016.

P. Richtarik and M. Takavc. Distributed Coordinate Descent Method for Learning with Big Data. *J. Mach. Learn. Res.*, 2016.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proc. CCS*, pp. 1310–1321. ACM, 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Xun Xian, Xinran Wang, Jie Ding, and Reza Ghanadan. Assisted learning: A framework for multi-organization learning. *Proc. NeurIPS*, 2020.

Pengtao Xie, Jin Kyu Kim, Yi Zhou, Qirong Ho, Abhimanu Kumar, Yaoliang Yu, and Eric Xing. Lighter-communication distributed machine learning via sufficient factor broadcasting. In *Proc. UAI*, pp. 795–804, 2016.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *ArXiv:1806.00582*, 2018.

Y. Zhou, Y.L. Yu, W. Dai, Y.B. Liang, and E.P. Xing. On convergence of model parallel proximal gradient algorithm for stale synchronous parallel system. In *Proc. AISTATS*, 2016.

Yi Zhou, Yingbin Liang, Yaoliang Yu, Wei Dai, and Eric P. Xing. Distributed proximal gradient algorithm for partially asynchronous computer clusters. *J. Mach. Learn. Res.*, 19(19):1–32, 2018.

# A Appendix

## A.1 Federated Learning (FL)

Federated learning is an emerging distributed learning framework (Shokri & Shmatikov, 2015; Konecny et al., 2016; McMahan et al., 2017; Zhao et al., 2018; Li et al., 2020; Diao et al., 2021; 2022) that aims to learn a global model using the average of local models trained by numerous smart devices with possibly imbalanced/heterogeneous data. The existing federated learning algorithms require frequent transmissions of local model parameters. This is different from our solution designed for the organizational learning scenarios, where each learner is an organization that often has unconstrained communication and computation resources, but is restricted to interacting with external service providers. Our solution aims to help the learner improve learning performance within ten rounds, while federated learning needs many more rounds.

### A.1.1 Comparison between FedAvg and AssistDeep

In this subsection, we compare the standard FedAvg algorithm (McMahan et al., 2017)for federated learning with centralized SGD (baseline), Learner-SGD (baseline) and our AssistDeep algorithm. We use the same datasets (CIFAR-10 and SVHN) and models (AlexNet and ResNet-18) as in Section 4.2 and keep all settings and hyperparameters unchanged. We implement all algorithms using the SGD optimizer with a learning rate of 0.01 for AlexNet and a learning rate of 0.1 for ResNet-18. The provider's data are uniformly sampled from all classes and are therefore balanced. The learner's data are also sampled with the same size as the provider's, but are taken from a single class and are therefore extremely imbalanced. For FedAvg, we treat the learner and provider as two federated learning clients/agents and use the same local data and local SGD iteration budgets as AssistDeep.

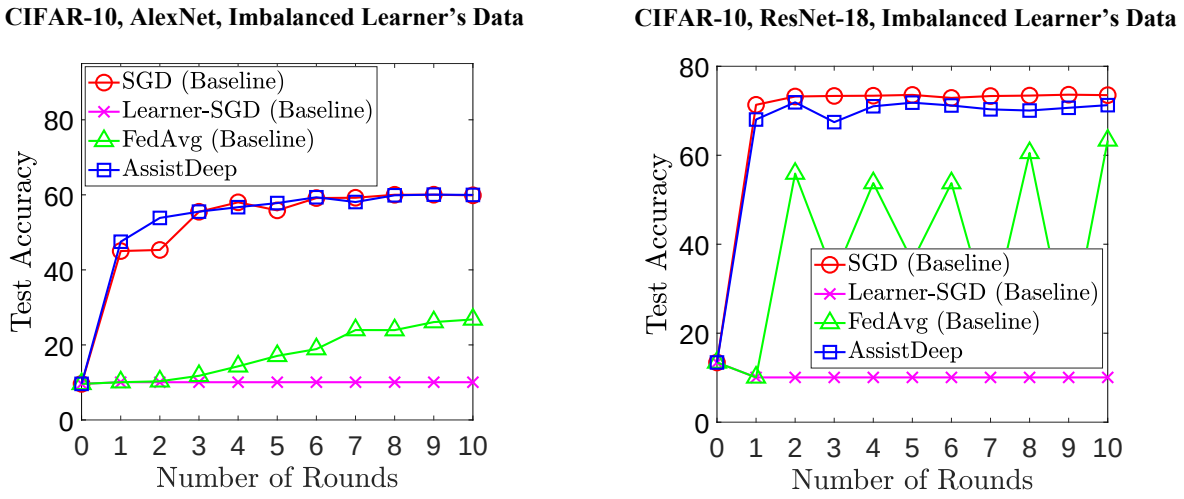


Figure 13: Comparison of AssistDeep, FedAvg, SGD, and Learner-SGD with imbalanced learner's data in training an AlexNet (left) and ResNet-18 (right) on CIFAR-10.

**CIFAR-10 Dataset.** We first compare these algorithms with extremely imbalanced learner's data in training an AlexNet and ResNet-18 on the CIFAR-10 dataset, respectively. In Fig. 13, we plot the test accuracy against the number of assistance rounds for AlexNet (left) and ResNet-18 (right), respectively. For SGD, Learner-SGD and AssistDeep, almost the same results can be observed as those shown in Fig. 3 (top left) and the same conclusions can be drawn as those made in subsection 4.2.1 (when the learner has extremely imbalanced data). Additionally, it can be observed that our AssistDeep algorithm converges much faster and achieves a much better test performance than FedAvg for both AlexNet and ResNet-18. This is because FedAvg simply averages the last local models generated by the two clients, and one of them is trained on highly imbalanced learner's data, which causes a significant bias. These results demonstrate that our proposed AssistDeep algorithm is more robust to data imbalance compared to standard federated learning.

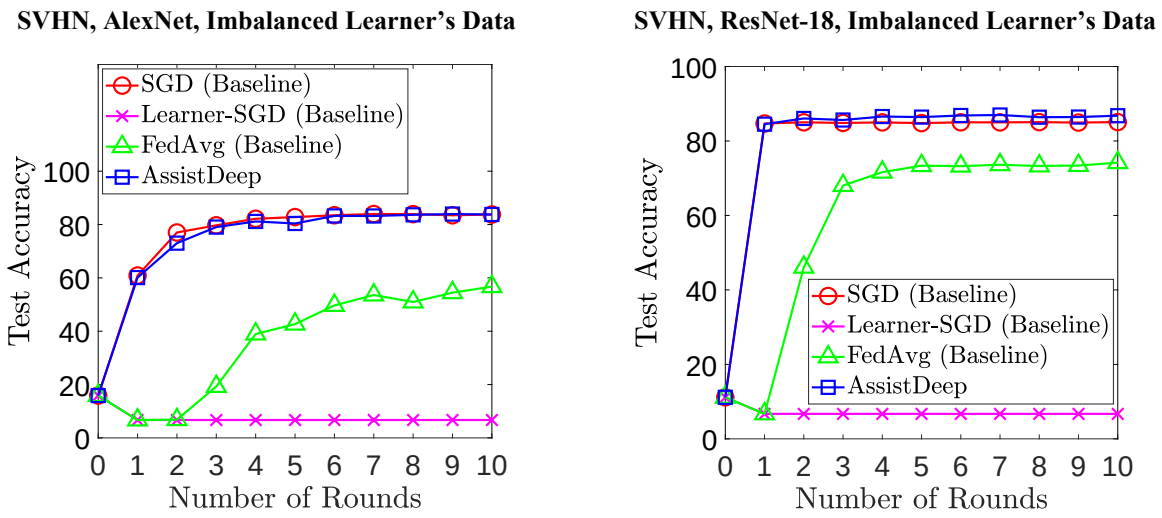


Figure 14: Comparison of AssistDeep, FedAvg, SGD, and Learner-SGD with imbalanced learner's data in training an AlexNet (left) and ResNet-18 (right) on SVHN.

**SVHN Dataset.** In Fig. 14, we repeat the above experiments with the SVHN dataset using AlexNet (left) and ResNet-18 (right), respectively. From these results, one can make the same observations as the previous CIFAR-10 results, which show that our proposed AssistDeep significantly outperforms FedAvg on diverse types of datasets.

## A.2 AssistKD: Assisted Learning with Knowledge Distillation

### A.2.1 Knowledge Distillation

Knowledge distillation (KD) is a machine learning technique that trains one classifier, called the student, using the outputs (also known as soft labels, which is a vector of probability scores assigned to each class) of another classifier, called the teacher (Hinton et al., 2015). It has been found to be often more efficient to train classifiers using soft labels rather than ground truth labels. The basic idea behind KD is to train a student model to match the output logits of a teacher model.

When defining the logit label of the teacher's model, a higher temperature $t$ represents the softer distribution of output classes. Suppose the output has $c$ classes. The $i$-th entry in the teacher's logit label is defined as

$$q_{T,i}(\mathbf{x}, t) = \frac{\exp(z_{T,i}(\mathbf{x})/t)}{\sum_{i=1}^{c} \exp(z_{T,i}(\mathbf{x})/t)},$$

where $z_{T,i}$ represents the pre-softmax layer associated with the $i$-th class. Similarly, we define the $i$-th entry in the student's logit label as

$$q_{S,i}(\mathbf{x}, \beta, t) = \frac{\exp(z_{S,i}(\mathbf{x}, \beta)/t)}{\sum_{i=1}^{c} \exp(z_{S,i}(\mathbf{x}, \beta)/t)},$$

where $\beta$ is the training parameters in student's model, and $z_{S,i}(\mathbf{x}, \beta)$ is the pre-softmax layer associated with the $i$-th class given training model parameters $\beta$. We use the following loss function (Hinton et al., 2015) for knowledge distillation:

$$\mathfrak{L}(y, \mathbf{x}, \beta) = -\alpha \sum_{i=1}^{c} q_{T,i}(\mathbf{x}, t) \log q_{S,i}(\mathbf{x}, \beta, t) - (1-\alpha) \sum_{i=1}^{c} \mathbf{1}(y = i) \log q_{S,i}(\mathbf{x}, \beta, 1), \tag{3}$$

where $\alpha$ is a hyper-parameter that decides the weighting of soft labels and hard labels, and $y$ is the ground truth label. Here, the first term in the above loss corresponds to the distillation loss between the teacher's soft label and the student's soft label, while the second term corresponds to the student's cross-entropy loss.

We adapt KD to the assisted learning framework and call it AssistKD. In AssistKD, knowledge distillation occurs in two stages: 1) When a learner learns its local models, it will distill every model generated in the local trajectory of models to the shared model architecture. To save computation, we use each distilled model as a warm start for distilling the next in the trajectory. 2) When a learner picks the model that yields the lowest global loss, the learner will distill the chosen model to its local model architecture to initialize the training of its local models

#### A.2.2 Experiments

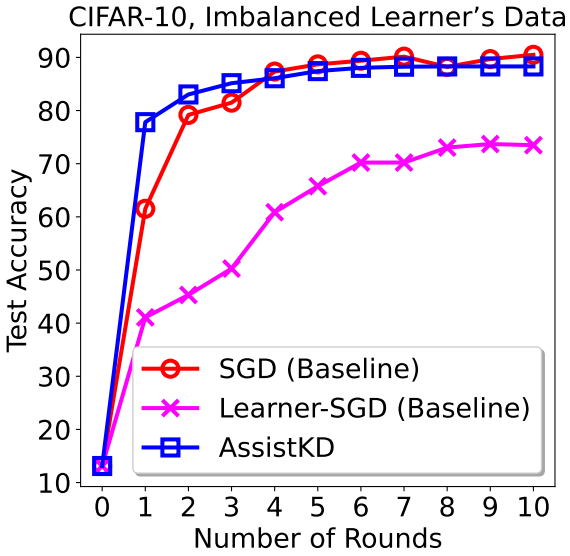


Figure 15: Comparison of AssistKD (CNN as the communication model), SGD (using model ResNet-18), and Learner-SGD (using model ResNet-18) on CIFAR-10.

We conduct an experiment where the learner and the provider train a ResNet-18 as their local training model. They communicate with each other by transmitting a smaller-scale CNN model (shared between learner and provider). In this example, the shared model consists of six convolutional layers and three fully connected layers. It only has half of the trainable parameters compared with the trainable parameters in ResNet-18.

We test the performance of AssistKD on CIFAR-10 by comparing it with two baselines: SGD (using centralized data $\mathcal{D}^{(\mathrm{L},\mathrm{P})}$) and Learner-SGD (using only the learner's data $\mathcal{D}^{(\mathrm{L})}$). We implement all these algorithms using the SGD optimizer with a learning rate of 0.1 and batch size of 256 to train CNN and ResNet-18 on CIFAR-10. We assign different SGD iterations to the aforementioned two-stage knowledge distillation. In particular, we let the SGD iterations in the first stage be 10. In the second stage, we choose the SGD iterations that traverse the local training data for 60 times, i.e., 60 epochs. We choose hyperparameter $t = 1$ and $\alpha = 0.9$ in (3).

We compared these algorithms with imbalanced learner data. In particular, the provider's data are uniformly sampled from all classes and are therefore balanced. The size of the learner's data is only 20% of the provider's sample size. Moreover, half of the learner data is taken from a single class, and the other half is uniformly sampled from other classes. We fix the number of assistance rounds to be 10. The total number of local SGD iterations in each assistance round is fixed to be 2000. We assign the SGD iteration budget to the learner and provider in proportion to their data samples' sizes. Both the learner and provider record their local training models and local loss values for every $I = 50$ SGD iteration, which is the sampling period.

The test performance is shown in Fig. 15. From the plot, the performance of AssistKD approaches that of the standard SGD, which trains a larger model (ResNet-18) on the centralized data. Moreover, the AssistKD significantly outperforms the Learner-SGD, which trains on the learner's data alone. This demonstrates that

our AssistKD enables the learner and provider to transmit models of different sizes/scales while keeping a similar learning performance to the original assisted learning.