Diffusion-Based Active Learning for Distributed Client Manifolds

Kwang In Kim

POSTECH
kimkin@postech.ac.kr

Abstract

In privacy-preserving distributed learning environments, data stored on local clients cannot be shared with other clients or servers. We consider a new active learning problem setup for these environments, where the server aims to build a centralized model by distributing labeling budgets across different clients. Our algorithm identifies which clients and their data points warrant annotation by estimating the global *impact* of the resulting labels. We evaluate this impact by embedding the clients into the manifold of learner parameters, formed by the task learner's predictions on unlabeled data, and diffusing the reduction in predictive uncertainties caused by labeling. The algorithm effectively selects clients with high estimated impact while achieving diversity in client selection, all without accessing local client data. In experiments, our approach demonstrates substantial advancements when compared to adaptations of existing active learning algorithms.

1 Introduction

Active learning (AL) attempts to optimize the use of limited labeling budgets by selecting and labeling the most *useful* data instances. Typically, AL is initialized with only a few training data points having ground-truth labels, while the remaining data points are unlabeled. Then, an AL algorithm iteratively identifies unlabeled subsets and queries their labels to an *oracle* based on the learning progress made by the learner network, e.g. how uncertain the learner is about its prediction on unlabeled data.

Beyond centralized learning scenarios, AL can also be beneficial in distributed collaborative learning environments, where data is divided among local clients and not shared externally to ensure data privacy and protection. In such environments, we envisage an AL scenario where initially, only a small set of clients possess labeled data. Subsequently, in each round, an AL algorithm selects a subset of clients and queries the labels of all or part of each client's data, while ensuring no data is shared between clients.

Adapting centralized AL strategies for distributed learning by selecting clients with the highest average uncertainties of their local data may appear to be a straightforward approach. However, this setting is sub-optimal as it does not account for the relationships between individual clients. The global model is trained on all labeled data from different clients, and since these local data are not statistically independent,[1] labeling points in one client can influence all clients. To improve the overall performance of the learner, it is sensible to select clients that most effectively contribute to reducing the uncertainty of *all* clients when labeled.

Assessing the *global impact* of labeling a client before selecting labels and training the learner is challenging as the underlying client relationships are not accessible. Ideally, the Kullback-Leibler divergence between local data distributions of clients would serve as a measure of these relationships, but this information is often not shared among clients. Instead, we use the learner parameters locally updated by each client as representations of individual clients and parameter similarities as a proxy for relationships. Using this structure, our algorithm embeds clients into a graph as a sample approximation of an underlying parameter manifold and assesses the effect of providing labels of a client node into the entire graph via conducting anisotropic diffusion therein. This enables us to trade-off between exploring less populated areas via diffusion and exploiting learners' progress via their uncertainty estimates at the client level.

To our knowledge, this represents the first instance of distributed active learning algorithms focusing on client-level selection. In experiments with five datasets, our algorithm demonstrated significant performance improvements over adaptations of existing AL algorithms.

2 Related Work

Active Learning (AL). Existing approaches can be categorized into three groups (Ren et al. 2022). *Distribution-based* methods aim to identify subsets of data that cover the underlying data-generating distributions well. For example, CoreSet algorithm by (Sener and Savarese 2018) selects data points that provide even coverage of data distributions, while (Nguyen and Smeulder 2004) proposed pre-clustering data to achieve diversity in data selection. (Sinha, Ebrahimi, and Darrell 2019)'s VAAL attempts to make the distributions of labeled and unlabeled data identical in a latent space, while (Guo 2010) maximizes the mutual information between labeled and unlabeled data. These methods are useful in iden-

[1] Otherwise, distributed learning becomes learning separate local models.

tifying sparsely labeled data areas but may not fully exploit the knowledge of the given learning problem. Evaluating the trained learner on unlabeled data at each round can provide insights into areas where additional labeling is desired, such as those close to decision boundaries.

Uncertainty sampling targets areas where the learner is least confident. Various uncertainty measures, such as entropy of class predictive distributions (Yun, Kim, and Kim 2020), predictive margin (Henter et al. 2015), and distances to decision boundaries (Tong and Koller 2001) are used to identify such areas. (Yoo and Kweon 2019)'s algorithm trains an auxiliary learner to estimate losses of the main task learner and selects data points with high estimated losses. Bayesian learning framework is employed by (Houlsby et al. 2011)'s BALD algorithm to maximize the gain of information about learner parameters, and (Kirsch, van Amersfoort, and Gal 2019)'s BatchBALD extends this for deep learning. However, pure uncertainty-based methods may fail to capture the relationship between data points, and the uncertainty predictions made by the learner can be unreliable in early AL rounds where labels are limited.

State-of-the-art *hybrid* algorithms combine both uncertainty sampling and distribution-based methods. For example, (Kim et al. 2021) extended VAAL to incorporate predicted learner losses. (Caramalau, Bhattarai, and Kim 2021) developed a sequential graph convolutional network (GCN)-based algorithm to build a graph embedding of data instances to identify points with high uncertainty which are sufficiently different from labeled data, and (Ash et al. 2020) proposed BADGE which analyzes the gradients of point losses to promote label selection diversity and uncertainty reduction. (Liu et al. 2021) improved uncertainty sampling by considering the expected model performance change caused by labeling selected points. However, hybrid and distribution-based approaches require explicit data representations, making them challenging to apply in distributed learning where relationships between individual clients are not directly available. Our algorithm adapts the concept of hybrid active learning algorithms to distributed learning in a non-trivial manner.

Distributed AL. Applying AL to distributed learning has only recently been studied. Existing methods focus on selecting data points from local client data using a global model trained on all labeled data, leading to local uncertainty sampling (Ahmed et al. 2020; van Bommel 2021; Qian, Gochhayat, and Hansen 2019) and hybrid selection (Ahn et al. 2022; Kim et al. 2023). (Kim et al. 2022) proposed an approach that enhances label selection efficiency within each client by combining a global model with local models trained on local labeled data. However, this method is not applicable in our scenario where most clients lack labeled data, and therefore local models are not defined.

Existing distributed AL methods can be adapted to our setting by combining them with uncertainty-based (e.g. based on the average entropies) or random client selection methods. However, our proposed method outperforms these adaptations by explicitly modeling client relationships.

Complementary to our work, (Goetz et al. 2019)'s *federated AL* uses selected portions of labeled local data to accelerate training, while (Huang et al. 2021)'s *asynchronous AL* optimizes label querying for asynchronous annotators. (Chen and Wujek 2020)'s *distributed AL* strategically distributes unlabeled data to different annotator clients to improve computation and communication efficiency. These approaches assume that the AL algorithm has access to all data.

3 Problem Definition: Active Learning in Distributed Environments

We focus on classification, aiming to construct a neural network classifier $f : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X}$ and $\mathcal{Y}$ being the space of input images and labels. In standard supervised learning, f is trained by minimizing the sum J of losses l on a training set $\mathcal{L} \subset \mathcal{X} \times \mathcal{Y}$: $J(\mathbf{w}) = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{L}} l(f(\mathbf{x}_i), \mathbf{y}_i)$, where $\mathbf{w}$ is the parameter vector of f. We use the cross-entropy loss l, assuming that $f(\mathbf{x})$ forms a class-conditional distribution while other losses can be employed.

Distributed Learning. In this setting, the training data is distributed across individual clients $\mathcal{C} = \{C_z\}_{z=1}^{M}$, each with its own data $\mathcal{L}_z$ sampled from the corresponding local distribution P_z. By defining $\mathcal{L}$ as the union of local data, the primary learning objective J can be reformulated as

$$J(\mathbf{w}) = \sum_{z=1}^{M} J_z(\mathbf{w}), \qquad \text{where} \tag{1}$$

$$J_z(\mathbf{w}) = \sum_{((\mathbf{x}_z)_i,(\mathbf{y}_z)_i)\in\mathcal{L}_z} l(f((\mathbf{x}_z)_i), (\mathbf{y}_z)_i).$$

The energy J can be minimized using distributed stochastic gradient descent (McMahan et al. 2017):

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \sum_{z=1}^{M} d(\mathbf{w}_z(t)). \tag{2}$$

Here, the parameter update $d(\mathbf{w}_z(t))$ at step t is the difference between the locally updated parameters $\mathbf{w}_z(t)$ stored in C_z and the global parameters $\mathbf{w}(t)$ shared by a central server. $\mathbf{w}_z(t)$ is obtained by performing local epochs in C_z:

$$\mathbf{w}_z^{k+1}(t) = \mathbf{w}_z^k(t) - \eta\nabla_{\mathbf{w}} J_z(\mathbf{w}^k(t)), \tag{3}$$

where η is the learning rate and $\mathbf{w}_z^1(t) = \mathbf{w}(t)$.

This update process facilitates privacy-preserving learning, as C_z does not share its local data $\mathcal{L}_z$ directly. Instead, it shares the local update $d(\mathbf{w}_z(t))$ an encapsulation of $\mathcal{L}_z$.

Distributed Active Learning (AL). Each client C_z maintains local labeled data $\mathcal{L}_z^r$ and unlabeled data $\mathcal{U}_z^r$. Initially in *round* $r = 0$, only a small fraction of clients have labeled data, and thus most $\mathcal{L}_z^0$'s are empty. In each subsequent AL round, the classifier f^r is trained on $\{\mathcal{L}_z^r\}_{z=1}^{M}$ by iterating Eq. 2. Based on the evaluation of f^r on $\{\mathcal{U}_z^r\}_{z=1}^{M}$, an AL algorithm first selects a subset of clients, referred to as a *client batch* $\mathcal{C}^r \subset \mathcal{C}$. Thereafter, for each selected client $C_z \in \mathcal{C}^r$, the algorithm constructs a *local data batch* $\mathcal{B}_z^r \subset \mathcal{U}_z^r$ of data points to label. Once $\mathcal{B}_z^r$ is labeled, $\mathcal{L}_z^{r+1}$ and $\mathcal{U}_z^{r+1}$ are subsequently obtained as $\mathcal{L}_z^r \cup \mathcal{B}_z^r$ and $\mathcal{U}_z^r \setminus \mathcal{B}_z^r$, respectively, adding total $|\mathcal{C}^r| \times |\mathcal{B}_z^r|$ labels per round. This problem is

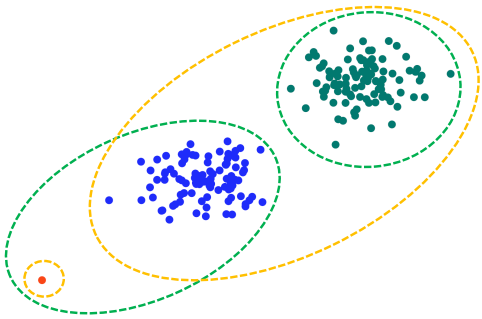

Figure 1: Example two-cluster clusterings. An isolated client (the red point) was added to points sampled from two Gaussian distributions (green and blue points). Orange and green lines indicate spectral clusterings obtained using the unnormalized Laplacian and random-walk normalized Laplacian ($L_{\text{rw}} = I - D^{-1}W$), respectively.

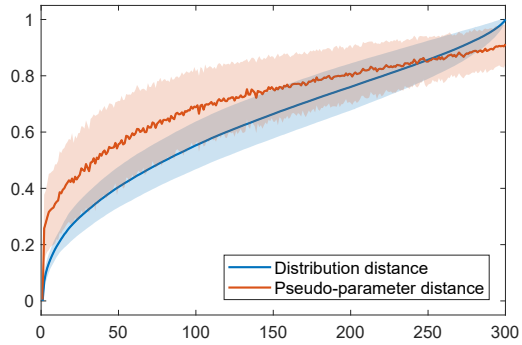

Figure 2: The alignment of client distances: CINIC-10 dataset is distributed to 300 clients based on Dirichlet distributions on their class labels (see Sec. 5). The (average) *distribution distance* is calculated by first building a matrix of pairwise KL-divergences, then for each column z (corresponding to client C_z), sorting the corresponding row in increasing order, and finally averaging all sorted columns. The *pseudo-parameter distance* is obtained by arranging each column of the pseudo-parameter distance matrix $[Q]_{zy} = q(C_z, C_y)$ (Eq. 5) based on the corresponding sorting indices of the distribution distances. The shaded area represents $\pm$ the standard deviations. For visualization, the distances are scaled to $[0, 1]$. Distances are scaled to $[0, 1]$ for visualization. The average Pearson correlation coefficient between the two distance types is 0.82 on a scale of $[-1, 1]$. Note that in practice, the ground-truth local data distributions are not shared.

equivalent to standard AL in each client when all clients are selected in each round, i.e. $\mathcal{C}^r = \mathcal{C}$. However, considering the overhead of recruiting annotators and preparing the labeling environments at clients, and considering the constraints posed by proprietary or privacy-sensitive data which might restrict internet-based labeling, it is more feasible for only a select few clients to collect labels, rather than expecting uniform label contributions from all clients.

4 Distributed Active Learning Algorithm

Typically, a centralized AL algorithm defines a *score* function $g : \mathcal{X} \to \mathbb{R}$ and constructs a batch $\mathcal{B}^r$ as the data points with the largest score values. A common choice is the entropy $g(\mathbf{x}) = -\sum_j [f^r(\mathbf{x})]_j \log([f^r(\mathbf{x})]_j)$ of $f^r(\mathbf{x})$ where $[\mathbf{a}]_j$ is the j-th component of $\mathbf{a}$, measuring how *uncertain* the f^r prediction is on $\mathbf{x}$.

Straightforwardly adapting this approach to client selection in distributed AL, one could attempt to define a client-level score $g(C_z)$ as the average entropy of f^r on $\mathcal{U}_z^r$. However, this does not take account of how *impactful* C_z is when selected: As clients' local data distributions are not entirely independent of each other, labeling points in a client C_z and accordingly updating the learner f^r would reduce the uncertainty of not just $\mathcal{U}_z^r$ but the entire unlabeled set $\{\mathcal{U}_z^r\}_{z=1}^M$. In this case, it is preferable to assess scores based on these global impacts of selecting clients, as the goal of AL is to enhance the overall performance of f^r. Figure 1 illustrates this with an example where we consider each point as a client. Suppose an isolated client (the red point) has the highest average entropy. Naïvely selecting this client would not substantially contribute to resolving the uncertainties of other clients. In this case, it would be more effective to select points in densely populated yet unlabeled areas (green and blue points).

Modeling impact also promotes diversity within a batch of clients: As shown shortly, once a client is selected in a region e.g. in the cluster of blue points in Fig. 1, our algorithm takes its impact into account (that the entropies of neighboring clients will be suppressed) and selects the next client from a distinct region (green cluster). To encapsulate the global impact of selecting individual clients, our approach employs the simulation of a diffusion process across the client manifold.

4.1 Forming Client Manifolds Using Pseudo-Parameters

Assessing the impact of labeling (points of) clients would require *similarities* (or relationships) between clients. A natural (dis-)similarity measure would be the Kullback-Leibler (KL) divergence between the respective local data distributions. Alternatively, when samples from these distributions are available, their empirical maximum mean discrepancy (MMD) can be used (Gretton et al. 2012). However, in privacy-preserving and data-protecting environments, neither the local distributions nor their samples are shared outside individual clients.

We consider the local parameter updates $\{d(\mathbf{w}_z^r)\}_{z=1}^M$ as surrogate representations of clients. They encapsulate how the global learner parameters $\mathbf{w}^r$ are affected by the information provided by each local dataset, and their distances can indicate how diverse these clients are. Evaluating these parameters involves labeled data that might not be defined in early AL rounds where most clients do not have labeled data. Instead, we use pseudo-labels $\{\mathbf{1}[f^r((\mathbf{x}_z)_i)]\}$ constructed as the one-hot-encoding of the classifications made by f^r on $\mathcal{U}_z^r$. Defining the local energy accordingly

$$\widetilde{J}_z^r(\mathbf{w}) = \sum_{(\mathbf{x}_z)_i \in \mathcal{U}_z^r} l(f^r((\mathbf{x}_z)_i), \mathbf{1}[f^r((\mathbf{x}_z)_i)]) \qquad (4)$$

and replacing J_z^r in Eq. 3 with $\widetilde{J}_z^r$, we obtain pseudo-

parameters (update) $\widetilde{d}(\mathbf{w}_z^r)$. The embedding m of clients into a manifold $\mathcal{M}$ is then defined as

$$m(C_z^r) = \frac{\widetilde{d}(\mathbf{w}_z^r)}{\|\widetilde{d}(\mathbf{w}_z^r)\|}$$

whose L_2-metric induces a metric q on $\mathcal{C}$:[2]

$$q(C_z^r, C_y^r) = \left\| m(C_z^r) - m(C_y^r) \right\|. \tag{5}$$

The local epoch index t is omitted for simplicity. As pseudo-labels are what f^r considers to be the correct predictions, $m(C_z^r)$ provides the direction in the parameter space that (f^r believes) improves the performance of f^r on $\mathcal{U}_z^r$. While pseudo-labels by themselves might not be suitable for updating f^r as they are erroneous, the resulting client distances can provide an indication of how diverse the clients are (see Fig. 2). As shown shortly, our final client distances will be obtained by combining q with the distances of entropy reductions observed at individual clients (Eq. 10).

Normalization is necessary for m: As $\widetilde{J}_z^r(\mathbf{w})$ linearly combines pointwise losses l, without normalization, even for two clients sharing the same data distribution, the distance can be large if the sizes of their unlabeled sets differ. The predictions $\{f^r(t)((\mathbf{x}_z)_i)\}_{(\mathbf{x}_z)_i \in \mathcal{U}_z^r}$ can be unstable at early epochs t. Therefore, we evaluate q based on the pseudo-parameters determined in later epochs: Our final distance is constructed by taking an average of q values for the last 25% of the total epochs in each AL round. Pseudo-labels have been widely applied to semi-supervised learning and AL (Hu et al. 2021; Lee 2013; Liu et al. 2021; Ash et al. 2020).

Since in practice, we do not have direct access to the underlying manifold $\mathcal{M}$, we take the graph $\mathcal{G}^r$ formed by $\{m(C_z^r)\}_{z=1}^M$ as a discrete approximation of $\mathcal{M}$. We will assess the impact of a client via simulating anisotropic diffusion on $\mathcal{G}^r$. First, we introduce the anisotropic diffusion structure on general graphs.

4.2 Client Selection Algorithm

Anisotropic Diffusion on a General Graph $\mathcal{G}$. For a given point set $\mathcal{P} = \{\mathbf{p}_z\}_{z=1}^M$ with a similarity structure $\widehat{w}$, a graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ is constructed by defining an edge $e(z, y) \in \mathcal{E}$ if

$$\mathbf{p}_z \in \mathcal{N}_K(\mathbf{p}_y) \text{ and } \mathbf{p}_y \in \mathcal{N}_K(\mathbf{p}_z),$$

where $\mathcal{N}_K(\mathbf{p})$ is the K-nearest neighbors of $\mathbf{p}$ within $\mathcal{P}$ induced by $\widehat{w}$.

An anisotropic diffusion of the evaluation of a function g on $\mathcal{G}$ ($\mathbf{g} = [g(\mathbf{p}_1), \dots, g(\mathbf{p}_M)]^\top$) is described as (Szlam, Maggioni, and Coifman 2008; Kim et al. 2015)

$$\frac{\partial \mathbf{g}}{\partial t} = -L^\gamma \mathbf{g}, \quad \text{where} \quad L^\gamma = D^\gamma - W^\gamma, \tag{6}$$

$$[W^\gamma]_{zy} = w(z, y)\gamma(z, y), \tag{7}$$

$$w(z, y) = \begin{cases} \widehat{w}(\mathbf{p}_z, \mathbf{p}_y), & \text{if } e(z, y) \in \mathcal{E} \\ 0, & \text{otherwise,} \end{cases}$$

and D^γ is a diagonal matrix of column-sums of W^γ: $[D^\gamma]_{zz} = \sum_{y=1}^M [W^\gamma]_{zy}$ with $[A]_{zy}$ being the (z, y)-th element of A. The anisotropicity γ can be defined as a positive function on $\mathcal{E}$ as specified later.

[2]More precisely, this is the *ambient* metric of $\mathcal{M}$ (Lee 1997).

Time-discretizing Eq. 6 (Griffiths and Higham 2010) obtains an iterative algorithm instantiating this process:

$$\mathbf{g}(s+1) = \mathbf{g}(s) - \delta L^\gamma \mathbf{g}(s) \tag{8}$$

with a positive step size δ. A simple interpretation of this algorithm is that at each point $\mathbf{p}_z \in \mathcal{P}$, the corresponding function value $g(\mathbf{p}_z)$ is gradually *spread* over $\mathcal{P}$ regulated by the *diffusivity* W^γ: If $\mathbf{p}_z$ and $\mathbf{p}_y$ are similar (as $\widehat{w}(\mathbf{p}_z, \mathbf{p}_y)$ specifies) and the anisotropicity $\gamma(z, y)$ is high, the diffusion tends to make their function values $g(\mathbf{p}_z)$ and $g(\mathbf{p}_y)$ similar.

Assessing Client Scores via Diffusion on Client Manifolds. Hypothetically when local points in a client C_z^r are labeled, and the classifier f^{r+1} is subsequently trained, the entropies of its prediction on the entire unlabeled set $\{\mathcal{U}_z^r\}_{z=1}^M$ will be reduced. We model such global impact of selecting C_z^r by simulating a diffusion of the average entropy values $\mathbf{g}^r = [g(C_1^r), \dots, g(C_M^r)]^\top$. First, we build a client graph $\mathcal{G}^r = (\mathcal{P}^r, \mathcal{E}^r)$ by assigning the client embeddings $\{m(C_z^r)\}_{z=1}^M$ to $\mathcal{P}^r$ and defining (see Eq. 5)

$$\widehat{w}(m(C_z^r), m(C_y^r)) = \exp\left(-\frac{q\left(C_z^r, C_y^r\right)^2}{\sigma_w^2} \right) \tag{9}$$

with a scale hyperparameter σ_w^2. The anisotropicity $\gamma(c, b)$ is defined based on the difference between the reduction of the average entropies in C_z and C_y:

$$\delta_g(C_z^r) = g(C_z^r) - g(C_z^{r-1}).$$

The diffusivity between C_z and C_y is strengthened when adding labels leads to similar degrees of uncertainty reduction for a parameter σ_γ^2:

$$\gamma(c, b) = \exp\left(-\frac{\left(\delta_g(C_z^r) - \delta_g(C_y^r)\right)^2}{\sigma_\gamma^2} \right). \tag{10}$$

The impact $v(C_z^r)$ of labeling C_z^r is assessed by first assuming that C_z^r's entropy is suppressed (caused by labeling) and then estimating the resulting reduction of the overall entropies in $\mathcal{C}$ via diffusing C_z^r's suppressed entropy: We build the initial entropy vector $\mathbf{g}_z^r(1)$ by replacing the z-th component of the original entropy evaluation $\mathbf{g}^r$ with 0. Thereafter $\mathbf{g}_z^r(h)$ is iteratively updated using Eq. 8. When the iteration terminates at step H, the impact $v(C_z^r)$ is calculated as the sum of the reduced entropies on $\mathcal{C} \setminus \{C_z\}$

$$v(C_z^r) = \sum_{y \neq z} [\mathbf{g}_z^r(1)]_y - \sum_{y \neq z} [\mathbf{g}_z^r(H)]_y. \tag{11}$$

It is important to exclude $[\mathbf{g}_z^r(H)]_z$ in the evaluation of $v(C_z^r)$. By construction of L^γ, the diffusion in Eq. 8 preserves the *total mass*: The total reduction of entropies in $\{C_y\}_{y \neq z}$ caused by diffusion is exactly the same as the increase of entropy on C_z (from zero; caused by diffusion) and therefore, $\sum_{y=1}^M [\mathbf{g}_z(1)]_y = \sum_{y=1}^M [\mathbf{g}_z(h)]_y$ for any h. Intuitively, v helps identify clients which lie in high-density areas (of clients) where labels are sparse.

Our final score is obtained by combining v and g:

$$s(C_z^r) = \frac{[\mathbf{g}^r]_z}{\text{std}(\mathbf{g}^r)} + \frac{[\mathbf{v}^r]_z}{\text{std}(\mathbf{v}^r)}, \tag{12}$$

Algorithm 1: Distributed AL. In each round r, a client batch $\mathcal{C}^r$ of size V is constructed and for each client in $\mathcal{C}^r = \{C_z\}$, a local data batch $\mathcal{B}_z^r$ of size W is selected.

Initial data: Unlabeled data $\{\mathcal{U}_z^0\}$ for each client and labeled data $\{\mathcal{L}_z^0\}$ for some clients $\{C_z^0\} \subset \mathcal{C}$;

Parameters: The number of diffusion steps H and its step size δ, nearest neighbor size K (Eq. 13), and scale parameters σ_w^2 (Eq. 6) and σ_γ^2 (Eq. 10).

Output: Labeled local data subsets $\{\mathcal{L}_z^R\}_{z=1}^M$.

for $r = 1, \ldots, R$ **do**
  Train f^r on $\{\mathcal{L}_z^r\}$ and evaluate pairwise q values (Eq. 5);
  Calculate $\mathbf{g}^r$ by evaluating f^r on $\{\mathcal{U}_z^r\}$ and construct the graph Laplacian L^γ (Eq. 6);
  for $v = 1, \ldots, V$ **do**
    for $z = 1, \ldots, M$ **do**
      Construct $\mathbf{g}_z^r(1)$ from $\mathbf{g}^r$;
      for $h = 1, \ldots, H$ **do**
        $\mathbf{g}_z^r(h+1) = \mathbf{g}_z^r(h) - \delta L^\gamma \mathbf{g}_z^r(h)$;
      end for
    end for
    Calculate $\mathbf{v}^r$ from $\{\mathbf{g}_z^r(H)\}_{z=1}^M$ (Eq. 11) and $\{s(C_z^r)\}$ from $\mathbf{v}^r$ and $\mathbf{g}^r$ (Eq. 12);
    Select C_o as the maximizer of s; $[\mathbf{g}^r]_o = 0$;
    Select $\mathcal{B}_z^r \subset \mathcal{U}_z^r$ of size W for labeling in C_o;
    $\mathcal{L}_z^{r+1} = \mathcal{L}_z^r \cup \mathcal{B}_z^r$; $\quad \mathcal{U}_z^{r+1} = \mathcal{U}_z^r \setminus \mathcal{B}^r$;
  end for
end for

where std($\mathbf{a}$) represents the standard deviation of elements in vector $\mathbf{a}$ and $\mathbf{v}^r = [v(C_1^r), \ldots, v(C_M^r)]^\top$.

Once the scores s of all clients are evaluated in round r, its minimizer C_o^r is selected for labeling. To prevent C_o from being selected again in the same round, $[\mathbf{g}^r]_o$ is set to zero. This process is repeated until a batch $\mathcal{C}^r$ of desired size V is constructed. Section 4.2 summarizes our AL procedure.

Building Local Data Batch $\{\mathcal{B}_z^r\}$. Our score function s for client selection can be adapted to building local batches within selected clients. In $C_z^r \in \mathcal{C}^r$, we first evaluate the predictive point-wise entropies $\mathbf{k}_z^r = [k^r((\mathbf{x}_z)_1), \ldots, k^r((\mathbf{x}_z)_U)]^\top$ with $k^r((\mathbf{x}_z)_i)$ being the entropy of $f^r((\mathbf{x}_z)_i)$. $g(C_z^r)$ is the average of the elements of $\mathbf{k}_z^r$. Then, similarly to the client graph $\mathcal{G}^r$, a local graph $\mathcal{G}_z^r = (\mathcal{X}_z^r, \mathcal{E}_z^r)$ on $\mathcal{X}_z = \mathcal{L}_z^r \cup \mathcal{U}_z^r$ is constructed by $e_z(i,j) \in \mathcal{E}_z$ if

$$(\mathbf{x}_z)_i \in \mathcal{N}_K((\mathbf{x}_z)_j) \text{ and } (\mathbf{x}_z)_j \in \mathcal{N}_K((\mathbf{x}_z)_i). \tag{13}$$

The corresponding local Laplacian L_z^r is constructed based on (see Eq. 6)

$$[W_z^r]_{ij} = \begin{cases} \exp\left(-\frac{\|(\mathbf{x}_z)_i - (\mathbf{x}_z)_j\|^2}{(\sigma_w^c)^2}\right), & \text{if } e_z(i,j) \in \mathcal{E}_z^r \\ 0, \quad \text{otherwise} \end{cases}$$

with a parameter σ_w^c. Finally, the local score s_z is defined similarly to Eq. 12 by combining $\mathbf{k}_z^r$ and the diffused entropy constructed based on L_z^r.

Discussion. Unlike the common use of graph Laplacian, our Laplacian (Eq. 6) is not normalized. In spectral clustering, for example, normalization is commonly employed to achieve balanced clusterings as exemplified in Fig. 1: Clustering with unnormalized Laplacian led to a trivial solution (orange dashed lines) assigning a cluster to a single isolated point (red). This is caused by its weak association (diffusivities) with other points. Normalizing the Laplacian strengthens these diffusivities, generating more balanced clustering (green). This effect is not desirable in evaluating the impact v: Isolated clients might show high average entropies, but selecting them for labeling would not help resolve the overall uncertainties in $\mathcal{C}$. Considering this isolated point as an outlier client C_z, strengthening its diffusivity would increase its impact $v(C_z)$ escalating the risk of selecting it. Indeed, when we use the normalized Laplacian and assign random entropy values in $[0, 1]$ to points other than C_z and a slightly higher entropy of 1.1 to C_z, the final score s was highest for C_z. This was avoided by employing the unnormalized Laplacian.

The anisotropicity γ (Eq. 6) helps exploit any additional similarity information which cannot be easily incorporated into the original metric $\widehat{w}$ of $\mathcal{M}$. Alternatively, one could define a new similarity $\widehat{w}'$ by combining $\widehat{w}$ and γ to construct an isotropic Laplacian. This is how our anisotropic Laplacian L^γ differs (slightly) from those of (Kim et al. 2015; Szlam, Maggioni, and Coifman 2008): Applied to our diffusion case, L^γ in (Kim et al. 2015; Szlam, Maggioni, and Coifman 2008) is constructed based on the equivalence between an anisotropic Laplacian on the original manifold $\mathcal{M}$ and the corresponding isotropic Laplacian on a new manifold $\mathcal{M}'$ formed by combining $\{\mathbf{p}\}$ and $\{\delta_g\}$: $\mathbf{p}' := [\mathbf{p}, \frac{\sigma_w}{\sigma_\gamma}\delta_g]^\top \in \mathcal{M}'$. This leads to a new similarity $\widehat{w}' = \exp\left(-\frac{\|\mathbf{p}'_z - \mathbf{p}'_y\|^2}{\sigma_w^2}\right)$. While both constructions properly instantiate discretizations of continuous anisotropic Laplacians on the original manifold $\mathcal{M}$, the latter does not lead to a consistent diffusion process when γ depends on g which is being diffused since this case, the resulting graph structure itself is constantly changing.[3]

5 Experiments

Experimental Setup. We evaluated the effectiveness of our algorithm on five benchmark datasets: CIFAR-10 and CIFAR-100 (Krizhevsky 2009), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), EMNIST letters (Cohen et al. 2017), and CINIC-10 (Darlow et al. 2018). For the first four datasets, the training sets were distributed to 100 clients, while for the (more extensive) CINIC-10 dataset, 300 clients were used. We considered two types of non-independent and identically distributed (non-IID) data allocation: In the first setting (*Dirichlet*), local data in each client was allocated based on the probability distributions of output classes generated by sampling from a Dirichlet distribution with $\alpha \in [0.1, 0.2]$ (Wang et al. 2020; Hsu, Qi, and Brown 2020). In the second setting (*Shard*), each local dataset was sampled from only 20% of total classes (McMahan et al. 2017).

[3]Refer to the technical appendix for the derivation of the graph anisotropic diffusion as a spatial discretization of continuous anisotropic diffusion on manifolds.

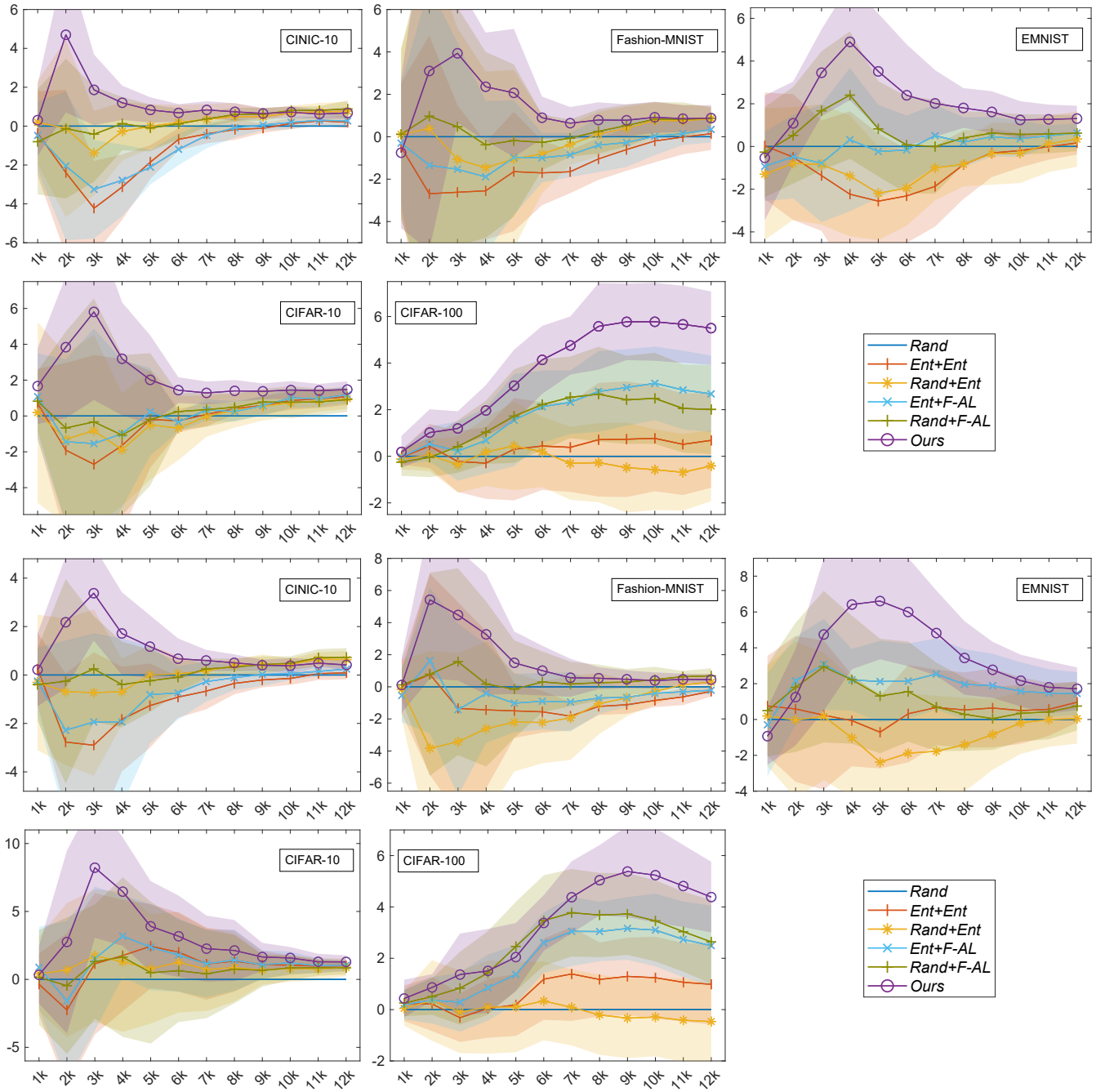

Figure 3: Mean accuracy improvement (in %; higher is better) of different distributed AL algorithms from random selection. The shaded area represents $\pm$ the standard deviations. The x-axis indicates the number of labels, starting with 1,000 randomly selected labels. The first and last two rows correspond to *Dirichlet* and *Shard* data allocations, respectively. *Rand+Ent* and *Ent+Ent* denote combinations of (Ahmed et al. 2020)'s federated AL algorithm with random and entropy-based sampling of clients, respectively. Similarly, *Rand+F-AL* and *Ent+F-AL* use (Ahn et al. 2022)'s *F-AL* algorithm. Absolute accuracy values are provided in the appendix (supplemental document).

For all datasets except CINIC-10, initially, 1,000 labels were prepared by selecting five clients and labeling 200 local data points per client. Afterward, 1,000 labels were added in each AL round by selecting five clients and 200 local points therein (i.e., V=5, W=200). For CINIC-10, we used V=10 and W=100. The number of total AL rounds R was set at 11, yielding total 12,000 labeled points in the final round. We ran experiments ten times with different random client

data allocations for each dataset and data allocation type, and averaged the results. We used the ResNet101 pre-trained on ImageNet combined with three fully connected layers of size 200 and a softmax layer as the learner network f. The learning rate η (Eq. 3) was initially set to 0.001, and it decayed by 0.1 at every 10-th epoch.

For comparison, we conducted experiments with baseline random client selection (*Rand*). As we are unaware of any existing AL methods directly applicable to our setting, we compared with *adaptations* of two existing distributed AL algorithms. As in our approach, they use global models trained through distributed learning to estimate point selection scores within each client. (Ahn et al. 2022)'s algorithm (*F-AL*) applies a hybrid selection strategy. (Ahmed et al. 2020)'s algorithm uses uncertainty sampling and we use entropy as the point uncertainty measure (*Ent*). This corresponds to a modification of our algorithm obtained by removing the impact term v in the final score s (Eq. 12). As both methods do not consider client selection, we combined them with random (*Rand*) and average entropy-based (*Ent*) client selection strategies. Additional comparisons, including an adaptation of (Ash et al. 2020)'s BADGE algorithm, are provided in the appendix (supplemental document).

Results. Figure 3 and Tab. 1 show the results. When clients are randomly selected, applying AL within each selected client (*Rand+Ent* and *Rand+F-AL*) improved accuracy over random selection, except for CINIC-10 (*Shard*) and EMNIST (*Dirichlet* and *Shard*). Uncertainty sampling for client selection yielded varied outcomes: *Ent+Ent* and *Ent+F-AL* performed well in four cases (CIFAR-10, and CIFAR100; *Dirichlet* and *Shard*) but underperformed in three others (CINIC-10; *Dirichlet* and *Shard*, and FashionMNIST; *Dirichlet*). In other cases, *Ent+Ent* and *Ent+F-AL* showed a tendency to improve accuracy only in later AL rounds, supporting our conjecture that independently assessing clients' uncertainties can inadvertently select clients with limited contribution to the overall performance (see Fig. 1).

This insight also applies when selecting labels within individual clients as indicated by the overall superior results of the hybrid *F-AL* approach over Ahmed et al.'s pure uncertainty-based algorithm. Embodying this insight into a selection mechanism at both client and local point levels, our algorithm demonstrated marked improvements over random selection and all combinations of *Ent* and *F-AL*. Ours was consistently best except for the final AL rounds of CINIC-10 (*Dirichlet* and *Shard*) and FashionMNIST (*Shard*) where *Rand+Ent* and *Rand+F-AL* respectively ranked first. Ours was statistically significantly better than *Rand* on 93.6% of total cases (for all datasets and # labels), and it was not significantly worse on any cases (Table 1). Uncertainty-based client sampling (*Ent+Ent* and *Ent+F-AL*) was better than *Rand* on 28.2% of total cases, but it was also worse on 17.3%.

The ablation study in the appendix shows that estimating and exploiting the global impact of selected entities at both the client and local point levels jointly contribute to improving the overall active learning performance.

Hyperparameters. The hyperparameters of our algorithm include the number of diffusion steps H and its step size δ,

Dataset	Method				
	Ent+Ent	*Rand+Ent*	*Ent+F-AL*	*Rand+F-AL*	*Ours*
CINIC-10	1 / -5	4 / 0	2 / -5	6 / 0	11 / 0
F-MNIST	0 / -4	3 / 0	0/ -2	3 / 0	9 / 0
EMNIST	0 / -3	0 / -3	2 / 0	2 / 0	10 / 0
CIFAR-10	5 / 0	3 / 0	4 / 0	4 / 0	11 / 0
CIFAR-100	0 / 0	0 / 0	10 / 0	9 / 0	11 / 0
CINIC-10	0 / -7	5 / 0	1 / 0	5 / 0	11 / 0
F-MNIST	0 / -8	0 / -3	0 / -4	2 / 0	9 / 0
EMNIST	1 / 0	0 / 0	11 / 0	0 / 0	10 / 0
CIFAR-10	5 / 0	3 / 0	6 / 0	3 / 0	10 / 0
CIFAR-100	5 / 0	0 / 0	9 / 0	9 / 0	11 / 0

Table 1: Results of significance tests. The positive and negative numbers indicate cases that are statistically significantly better and worse, respectively, than *Rand* (based on Student's t-test with $\alpha = 0.95$), across varying numbers of labels. The first and last four results correspond to *Dirichlet* and *Shard* data allocations, respectively.

the number of nearest neighbors K (Eq. 13), and the scale parameters σ_w^2 (Eq. 9) and σ_γ^2 (Eq. 10) of the Laplacian. σ_w^2 was decided as the mean of the squared distances q following the standard practice in graph Laplacian applications (Hein and Maier 2007). σ_γ^2 was similarly determined as the mean squared distance of δ_g, but it was scaled by a factor $\frac{1}{\rho}$. ρ balances the contributions of $\widehat{w}$ and γ, and it was set to 0.2. When $\rho = 0$ (anisotropicity is disabled as $\gamma = 1$; Eq. 10), the average accuracy decreased by 0.27% for CIFAR-10 (see appendix) while increasing it to values higher than 1 also rapidly degraded the performance. K was fixed at a small value of 7. The influence of K, H, and δ are complementary: Increasing their values tends to strengthen the effect of diffusion. δ was fixed at a small value of 0.1 for stable time-discretization (Griffiths and Higham 2010), while H was determined at 5. The parameters for local data batch selection were determined similarly. Optimizing hyperparameters is a difficult problem in AL due to the limited number of labeled points. As these parameters were not optimized, tuning them depending on the problem can improve performance. Our appendix (supplementary document) includes an analysis of the effect of varying hyperparameters and a detailed computational complexity analysis.

6 Conclusions

Existing distributed active learning (AL) algorithms focus on labeling points within individual clients, without leveraging the relationships between them. Our algorithm overcomes this limitation by embedding clients into a surrogate geometry $\mathcal{M}$ of the pseudo-parameters, capturing client relationships using the natural metric within $\mathcal{M}$. By implementing anisotropic diffusion on $\mathcal{M}$, our algorithm assesses the *global impacts* of labeling points in selected clients. This provides a unique capability for selecting impactful and diverse entities at both the client and local point levels. Our algorithm demonstrated significant gains in labeling efficiency over adaptations of state-of-the-art distributed AL approaches.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant (No. 2021R1A2C2012195) and the Institute of Information & communications Technology Planning & Evaluation (IITP) grants (No.2019-0-01906: Artificial Intelligence Graduate School Program (POSTECH); RS-2022-II220290: Visual Intelligence for Space-Time Understanding and Generation Based on Multi-layered Visual Common Sense), funded by the Korean government (MSIT).

References

Ahmed, L.; Ahmad, K.; Said, N.; Qolomany, B.; Qadir, J.; and Al-Fuqaha, A. 2020. Active learning based federated learning for waste and natural disaster image classification. *IEEE Access*, 8: 208518–208531.

Ahn, J.-H.; Kim, K.; Koh, J.; and Li, Q. 2022. Federated active learning (F-AL): an efficient annotation strategy for federated learning. In *arXiv:2202.00195*.

Ash, J. T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; and Agarwal, A. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*.

Caramalau, R.; Bhattarai, B.; and Kim, T.-K. 2021. Sequential graph convolutional network for active learning. In *CVPR*, 9583–9592.

Chen, X.; and Wujek, B. 2020. AutoDAL: distributed active learning with automatic hyperparameter selection. In *AAAI*.

Cohen, G.; Afshar, S.; Tapson, J.; and van Schaik, A. 2017. EMNIST: an extension of MNIST to handwritten letters. In *arXiv:1702.05373*.

Darlow, L. N.; Crowley, E. J.; Antoniou, A.; and Storkey, A. J. 2018. CINIC-10 is not ImageNet or CIFAR-10. In *arXiv:1810.03505*.

Goetz, J.; Malik, K.; Bui, D.; Moon, S.; Liu, H.; and Kumar, A. 2019. Active federated learning. In *arXiv:1909.12641*.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *JMLR*, 13: 723–773.

Griffiths, D. F.; and Higham, D. J. 2010. *Numerical Methods for Ordinary Differential Equations*. Springer.

Guo, Y. 2010. Active instance sampling via matrix partition. In *NIPS*, 802–810.

Hein, M.; and Maier, M. 2007. Manifold denoising. In *NIPS*, 561–568.

Henter, D.; Stahlt, A.; Ebbecke, M.; and Gillmann, M. 2015. Classifier self-assessment: active learning and active noise correction for document classification. In *ICDAR*, 276–280.

Houlsby, N.; Huszár, F.; Ghahramani, Z.; and Lengyel, M. 2011. Bayesian active learning for classification and preference learning. In *arXiv:1112.5745*.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2020. Federated visual classification with real-world data distribution. In *ECCV*.

Hu, Z.; Yang, Z.; Hu, X.; and Nevatia, R. 2021. SimPLE: similar pseudo label exploitation for semi-supervised classification. In *CVPR*, 15099–15108.

Huang, S.-J.; Zong, C.-C.; Ning, K.-P.; and Ye, H.-B. 2021. Asynchronous active learning with distributed label querying. In *IJCAI*.

Kim, K.; Park, D.; Kim, K. I.; and Chun, S. Y. 2021. Task-aware variational adversarial active learning. In *CVPR*, 8166–8175.

Kim, K. I.; Tompkin, J.; Pfister, H.; and Theobalt, C. 2015. Context-guided diffusion for label propagation on graphs. In *ICCV*, 2776–2784.

Kim, S.; Bae, S.; Song, H.; and Yun, S.-Y. 2023. Re-thinking federated active learning based on inter-class diversity. In *CVPR*, 3944–3953.

Kim, S.; Bae, S.; Yun, S.-Y.; and Song, H. 2022. LG-FAL: federated active learning strategy using local and global models. In *ICML Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 127–139.

Kirsch, A.; van Amersfoort, J.; and Gal, Y. 2019. BatchBALD: efficient and diverse batch acquisition for deep Bayesian active learning. In *NeurIPS*.

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

Lee, D.-H. 2013. Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on challenges in representation learning*.

Lee, J. M. 1997. *Riemannian Manifolds: An Introduction to Curvature*. New York: Springer.

Liu, Z.; Ding, H.; Zhong, H.; Li, W.; Dai, J.; and He, C. 2021. Influence selection for active learning. In *ICCV*, 9274–9283.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

Nguyen, H. T.; and Smeulder, A. 2004. Active learning using pre-clustering. In *ICML*, 623–630.

Qian, J.; Gochhayat, S. P.; and Hansen, L. K. 2019. Distributed active learning strategies on edge computing. In *Cyber Security and Cloud Computing*.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B. B.; Chen, X.; and Wang, X. 2022. A survey of deep active learning. *ACM Computing Surveys*, 54(9): 180:1–180:40.

Sener, O.; and Savarese, S. 2018. Active learning for convolutional neural networks: a core-set approach. In *ICLR*.

Sinha, S.; Ebrahimi, S.; and Darrell, T. 2019. Variational adversarial active learning. In *ICCV*, 5972–5981.

Szlam, A. D.; Maggioni, M.; and Coifman, R. R. 2008. Regularization on graphs with function-adapted diffusion processes. *JMLR*, 9: 1711–1739.

Tong, S.; and Koller, D. 2001. Support vector machine active learning with applications to text classification. *JMLR*, 2: 45–66.

van Bommel, J. 2021. *Active Learning during Federated Learning for Object Detection*. Bachelor's thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Netherlands.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *NeurIPS*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. In *arXiv:1708.07747*.

Yoo, D.; and Kweon, I. S. 2019. Learning loss for active Learning. In *CVPR*, 93–102.

Yun, J.; Kim, B.; and Kim, J. 2020. Weight decay scheduling and knowledge distillation for active learning. In *ECCV*, 431–447.