Client-level Active Error Correction in Distributed Learning

Kwang In Kim
POSTECH
kimkin@postech.ac.kr

Abstract

Label errors can significantly degrade model performance, making effective mechanisms crucial. Active error correction (AEC) addresses this by prioritizing data points for human re-labeling where corrections are expected to have significant impact. We extend AEC to distributed collaborative learning, where clients hold local data and a central server allocates labeling resources. Existing AEC methods assume centralized access and do not generalize to distributed settings. To overcome this, we use neural network weight gradients from client updates as proxies for local data and apply a Gaussian process in gradient space to strategically select clients for correction. Our method identifies gradient inconsistencies and encourages diversity through a computationally efficient rank-one Cholesky update. Experiments on eight benchmark datasets demonstrate the effectiveness of our approach.

1 Introduction

Consider a scenario where a dataset contains noisy labels, and a fixed amount of resources is allocated to correct these errors within a selected subset of the data. However, the specific instances of mislabeled data are not known in advance. Random selection may waste resources on already correct labels, motivating the need for active error correction (AEC). AEC aims to optimize re-labeling efforts by prioritizing points likely to enhance learning outcomes. Common strategies for AEC include selecting those points with significant losses (Bernhardt et al. 2022) and uncertainties (Park, Jo, and Choi 2021), and learning error models (Li et al. 2022b).

In distributed collaborative learning, training data reside with individual *clients*, who retain control to preserve data confidentiality. Consider an AEC scenario in this setting, where the central coordinator aims to develop a well-trained network while operating within a constrained re-labeling budget allocated to clients. In this context, the coordinator must strategically distribute the budget to maximize overall performance improvement.

This scenario arises, for example, when a coordinator aims to develop a patient diagnosis prediction system using data from multiple hospitals and healthcare centers, each acting as an independent client. While these institutions are willing to contribute to and benefit from the system, concerns over patient privacy prevent them from sharing raw data. Moreover, some clients may have limited resources for obtaining high-quality annotations, resulting in potentially noisy labels. A key challenge in implementing AEC under these conditions is that the coordinator (i.e. the server) lacks access to local client data, restricting its ability to devise an effective selection strategy. Consequently, conventional centralized AEC algorithms are not suitable for this scenario.

In this paper, we present Gaussian Process AEC (*GPAEC*), a server-side AEC algorithm tailored for distributed learning environments. *GPAEC* uses the gradients of the network weights provided by clients as proxies for local data and applies a Gaussian process (GP) model in the gradient space to identify clients with high label error likelihoods. This model also incorporates a diversity-promoting mechanism to avoid repeatedly selecting *similar* clients. By combining GP-induced client inconsistency with the loss information provided by individual clients (which may be imprecise), our algorithm can avoid selecting clients likely to propagate erroneous information. Implemented via an efficient rank-one Cholesky update, *GPAEC* ensures computational efficiency.

To our knowledge, this work represents the first attempt to address AEC in a distributed learning context. In experiments with eight benchmark datasets, our algorithm demonstrates significantly improved error correction efficiency over random selection methods and adaptations of traditional, non-distributed AEC algorithms.

2 Related Work

Distributed Learning enables model training in scenarios where traditional centralized methods are impractical, e.g. due to data privacy, ownership, or scalability constraints. Recent research has focused on improving communication efficiency (e.g. via model quantization (Koloskova et al. 2020) and active client selection (Tang et al. 2022; Huang et al. 2024; Li et al. 2022a)), accommodating heterogeneous client data through personalized training (Tan et al. 2023), enhancing robustness to Byzantine attacks and data corruption (Zhu et al. 2023; Turan et al. 2021; Németh et al. 2022), and preserving data privacy through differential privacy and reconstruction prevention (Ovi et al. 2023; Huang et al. 2021).

Active Error Correction (AEC). AEC methods in centralized learning iteratively select a subset of samples for relabel-

ing to improve label quality under limited annotation budgets. These methods typically rely on model-derived signals such as predictive confidence or loss. For instance, Rebbapragada et al. (2012) selected samples with the highest uncertainty (entropy) values, estimated from class-conditional predictive distributions produced by the trained model. Henter et al. (2015) used the margin between the top two predictive probabilities as an uncertainty measure. Nallapati, Surdeanu, and Manning (2009) selected samples with high loss values, based on the intuition that the model learns correctly labeled examples more easily, yielding smaller losses before eventually memorizing noisy ones. Bernhardt et al. (2022) combined these strategies and suggested selecting samples that have both large loss and high entropy values.

A straightforward extension of AEC to distributed learning would select clients with high average loss, entropy, or gradient magnitude. However, such strategies become unreliable in the presence of faulty or adversarial clients that may intentionally report misleading statistics (e.g. false loss values or dataset sizes; see Sec. 5). Our experiments show that combining our proposed client *inconsistency* measure with potentially unreliable loss information substantially improves error-correction efficiency compared with naive adaptations of existing AEC approaches.

Byzantine-Robust Algorithms, and Automatic Error Correction and Detection. Byzantine-robust algorithms aim to defend against adversarial or corrupted client updates within a single training round, whereas AEC operates across multiple correction stages by identifying clients whose data should be relabeled by human annotators. Certain robustness criteria or algorithms can in principle, be adapted for client selection, e.g. by filtering clients with persistently high loss values (Li et al. 2022b), large gradient norms (Li et al. 2024), or anomalous gradients identified through robust mean estimation (Turan et al. 2021). However, our experiments show that the proposed method substantially outperforms such adaptations.

AEC also complements studies on automatic error correction and detection, which aim to identify or correct mislabeled data without human intervention. These techniques have been explored in centralized settings using, for instance, noise transition models (Li et al. 2022b), and have more recently been extended to distributed learning (Tsouvalas et al. 2024; Ji et al. 2024; Jiang et al. 2024; Xu et al. 2022). As shown in Sec. 5, our algorithm achieves substantially higher performance than direct adaptations of automatic error detection methods for client selection. The supplementary material further shows that robust distributed learning and AEC are complementary, and that their integration yields performance surpassing that of any individual component.

3 Distributed Collaborative Learning

In a standard (non-distributed) learning scenario, we are provided with a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ consisting of N input-output pairs. In this context, the goal of training a neural network is to minimize the following energy:

$$f(\boldsymbol{w};\ \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} l(\boldsymbol{w};\ (\boldsymbol{x}_i, \boldsymbol{y}_i)), \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is the weight vector of the network being optimized, and l is a loss function. This problem is typically solved through gradient descent iterations:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \nabla_{\boldsymbol{w}} f(\boldsymbol{w};\ \mathcal{D}), \tag{2}$$

where t is the iteration number and η is the learning rate.

In distributed learning with P clients, each client p has its own local dataset $\mathcal{D}_p$ of size N_p. In this case, the learning energy and the corresponding weight update steps can be respectively presented as

$$f(\boldsymbol{w};\ \{\mathcal{D}_p\}_{p=1}^{P}) = \sum_{p=1}^{P} \frac{N_p}{N} f(\boldsymbol{w};\ \mathcal{D}_p)$$

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \sum_{p=1}^{P} \frac{N_p}{N} \nabla_{\boldsymbol{w}} f(\boldsymbol{w}(t);\ \mathcal{D}_p). \tag{3}$$

In this model, the local data $\mathcal{D}_p$ can be kept confidential within the respective clients. Instead, each client p contributes to the learning process by transmitting the gradient $\nabla_{\boldsymbol{w}} f(\boldsymbol{w};\ \mathcal{D}_p)$, encapsulating $\mathcal{D}_p$, to a central server that coordinates the learning process. While our primary focus is on network training based on Eq. (3), our algorithm can be applied to different types of learning algorithms. For instance, individual clients may execute multiple weight update steps using their local data before transmitting the weight updates (instead of the gradients). Additionally, during each training step t, a subset of clients might actively participate in weight updates, implementing stochastic gradient descent.

4 Distributed Active Error Correction

4.1 Problem Formulation

In standard centralized learning settings with noisy labels, active error correction (AEC) aims to identify and re-label a subset of the training data $\mathcal{D}$ to improve overall model performance. Typically, this process proceeds over multiple stages: At each stage s, a model is trained on the current dataset $\mathcal{D}(s)$, and samples are prioritized for correction based on criteria such as high training loss or predictive uncertainty (e.g., entropy), under the assumption that mislabeled instances tend to incur larger losses or ambiguous predictions (Park, Jo, and Choi 2021; Huang et al. 2019; Bernhardt et al. 2022). Once a batch of samples is selected, they are sent for human annotation, and the corrected labels are incorporated to form the updated dataset $\mathcal{D}(s+1)$. The model is retrained in the following stage, and this process is repeated until the re-labeling budget is exhausted.

Extending this paradigm to distributed collaborative learning presents significant challenges. Each of the P clients maintains a private local dataset, and the server no longer has access to individual samples or labels. As a result, the focus of correction must shift from individual instances to entire clients. The server must allocate a limited re-labeling budget by selecting clients likely to contain noisy labels, relying solely on indirect signals from client behavior. Once selected, relabeling is performed locally by each client without exposing raw data to the server.

This task introduces several complications. Some clients may be uninformative due to small or low-quality data, while

others may act adversarially by submitting fabricated gradients or losses (Karimireddy, Guo, and Jordan 2022) to exploit correction resources. In more disruptive cases, compromised clients may manipulate updates to disrupt the AEC process or degrade training. Even benign clients may be unable or unwilling to share reliable feedback due to privacy concerns or limited resources. Heuristics such as average client loss $\{f(\boldsymbol{w}; \mathcal{D}_p)\}_{p=1}^P$ are therefore often unreliable and impractical. Effectively addressing label noise thus requires a robust selection strategy that can identify clients most likely to benefit from relabeling, while avoiding unnecessary engagement with already clean or *unreliable* participants.

4.2 Client Inconsistency

Our strategy focuses on identifying clients whose data exhibit significant discrepancies compared to others. We hypothesize that such *inconsistencies* may indicate a higher likelihood of label errors in their local data. Therefore, by correcting such data, we anticipate a more substantial improvement in the learning process. Since the server cannot directly access individual client data, we use the gradients of clients $\{\nabla_{\boldsymbol{w}} f(\boldsymbol{w};\ \mathcal{D}_p)\}_{p=1}^P$, as a proxy for their data. For simplicity, we denote $\nabla_{\boldsymbol{w}} f(\boldsymbol{w};\ \mathcal{D}_p)$ as ∇_p.

A naive measure of client inconsistency is the deviation of a gradient from representative values:

$$u(p) = \|\nabla_p - \boldsymbol{r}\|, \tag{4}$$

where $\boldsymbol{r}$ can represent either the mean or (geometric) median of $\{\nabla_p\}_{p=1}^P$. This approach can be seen as an adaptation of Byzantine-robust learning algorithms to the AEC context, where gradient vectors significantly deviating from the mean or median are considered Byzantine (Turan et al. 2021).[1] However, in typical distributed learning environments, the underlying distributions $\{P_p\}_{p=1}^P$ of client data can vary significantly, making them non-identically and independently distributed (non-i.i.d.). In such cases, relying on a single statistic like the mean or median is unlikely to sufficiently capture the diversity among client gradients. As shown in Table 2, the performance of the AEC algorithm that uses u is only marginally better than random selection.

We define the inconsistency $v(p; \mathcal{H}_p)$ of a client gradient ∇_p, given a set of weight gradients $\mathcal{H}_p = \{\nabla_i\}$, as follows:

$$v(p;\ \mathcal{H}_p) = 1 - \boldsymbol{k}_p^\top (\boldsymbol{K}_p + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_p, \tag{5}$$

$$\boldsymbol{k}_{p_{[i]}} = k(\nabla_p, \nabla_i),\ \text{for}\ \nabla_i \in \mathcal{H}_p,$$

$$\boldsymbol{K}_{p_{[ij]}} = k(\nabla_i, \nabla_j),\ \text{for}\ \nabla_i, \nabla_j \in \mathcal{H}_p,$$

$$k(\nabla_i, \nabla_j) = \exp\left(-a\|\nabla_i - \nabla_j\|^2\right), \tag{6}$$

where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{k}_{[i]}$ represents the i-th element of vector $\boldsymbol{k}$, $\boldsymbol{K}_{[ij]}$ is the i, j-th element of matrix $\boldsymbol{K}$, and σ^2 and a are positive constants.

This measure is derived from a Gaussian process (GP) model applied to the space of network weight gradients $\mathcal{W}$. A GP model is characterized by a mean function $m : \mathcal{W} \mapsto \mathbb{R}$ and covariance function $k : \mathcal{W} \times \mathcal{W} \mapsto \mathbb{R}$ (Rasmussen and Williams 2006). Suppose there is a hypothetical task where the objective is to estimate an underlying target function $q : \mathcal{W} \mapsto \mathbb{R}$, based on potentially noisy *observations* $\mathcal{Q}_p = \{(\nabla_i, q(\nabla_i) + \epsilon)\}$, where ϵ is a random variable representing the error level. We use the standard Gaussian noise model for ϵ, with a mean of zero and variance σ^2. Given the sample observations $\mathcal{Q}_p$ and the noise ϵ, the prediction $g(\nabla_p)$ of the target q on a test sample ∇_p is obtained as a probability distribution $P(g(\nabla_p))$ whose mean $\mathbb{E}[g(\nabla_p)]$ and variance $\mathbb{V}[g(\nabla_p)]$ are respectively presented as:

$$\mathbb{E}[g(\nabla_p)] = \boldsymbol{a}_p^\top \boldsymbol{q}, \quad \mathbb{V}[g(\nabla_p)] = k(\nabla_p, \nabla_p) - \boldsymbol{a}_p^\top \boldsymbol{k}_p,$$
$$\boldsymbol{a}_p = (\boldsymbol{K}_p + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_p^\top, \tag{7}$$

where $\boldsymbol{q}_{[i]} = q(\nabla_i) + \epsilon$. Finally, $v(p; \mathcal{H}_p)$ is obtained by applying the Gaussian kernel $k(\cdot, \cdot)$ (Eq. (6)) and considering only the predictive variances $\mathbb{V}[g(\nabla_p)]$.

We evaluated how well v captures client data inconsistency by comparing $\{v(p; \mathcal{H}_p)\}_{p=1}^P$ with client-wise label error probabilities $\{P_{\text{e}}(p)\}_{p=1}^P$ (on *TImgNet*). The resulting mean squared correlation was 0.86, indicating a significant correlation (see the supplementary document).

Discussion. The GP predictive variance $\mathbb{V}[g(\nabla_p)]$ quantifies the uncertainty associated with the mean prediction $\mathbb{E}[g(\nabla_p)]$ generated by the GP model when it is *trained* on $\mathcal{Q}_p$. Such predictive variances have been applied in various domains, including sensor placement (Krause, Singh, and Guestrin 2008), active learning (Kandasamy, Schneider, and Póczos 2015), and data point rejection (Seo et al. 2000).

To derive the predictive distribution $P(g(\cdot))$, we introduced a hypothetical target function q. However, defining the inconsistency measure v does not require the existence of this function, as the predictive variance $\mathbb{V}[g(\cdot)]$ remains independent of it.[2] This independence results from the Gaussian noise model ϵ, while under different noise models, the presence of the target function q may influence $\mathbb{V}[g(\cdot)]$.

Therefore, the variance $\mathbb{V}[g(\nabla_p)]$ can be interpreted as an inverse measure of the information conveyed by $\mathcal{H}_p$ about ∇_p. Specifically, it reflects the *collective similarity* of ∇_p to the elements in $\mathcal{H}_p$: $\boldsymbol{k}_p$ in Eq. (5) encodes raw similarities, which may be distorted when multiple clients have label errors and do not capture inter-client dependencies. To address this, we calibrate $\boldsymbol{k}_p$ by multiplying it with $(\boldsymbol{K}_p + \sigma^2 \boldsymbol{I})^{-1}$, incorporating relationships among gradients in $\mathcal{H}_p$.

Furthermore, the inner product of $\boldsymbol{k}_p$ with $\boldsymbol{a}_p$ in Eq. (5) introduces a regularization effect, allowing $1 - v(p; \mathcal{H}_p)$ to be interpreted as a reconstruction of the kernel evaluations $\boldsymbol{k}$ using a *linear smoother*, with $\{\boldsymbol{k}_{p_{[i]}}\}_{i \neq p}$ as the target values (Sollich and Williams 2004):

$$1 - v(p; \mathcal{H}_p) = \sum_{i=1}^{|\mathcal{H}_p|} s_i(\nabla_p) \boldsymbol{k}_{p_{[i]}}. \tag{8}$$

[1] For algorithms aimed at achieving Byzantine robustness, the primary objective is to *filter out* adversarial clients, unlike our problem, where the goal is to *quantify* the extent of label error.

[2] For further discussion on the target independence effect, see (Krause, Singh, and Guestrin 2008) in the context of sensor placement. Alternatively, one could explicitly define the target q, e.g., via a GP filtering model (Deisenroth, Huber, and Hanebeck 2009), which would implicitly establish a gradient denoising task.

The *weight* functions $\{s_i\}_{i=1}^{|\mathcal{H}_p|}$ are presented as combinations of smoothing kernels:[3]

$$s_i(\nabla_p) = \sum_{j=1}^{|\mathcal{H}_p|} b_{ij} k(\nabla_p, \nabla_j) \quad \text{with} \quad b_{ij} = (\boldsymbol{K}_p + \sigma^2 \boldsymbol{I})^{-1}_{[i,j]}.$$

4.3 Error Correction Process

Initially, at the active error correction (AEC) stage $s = 0$, each client p possesses a local dataset $\mathcal{D}_p(0)$, which may contain noisy labels. Subsequently, at each stage s, a network is trained on $\{\mathcal{D}_p(s)\}_{p=1}^P$ by iterating over multiple weight update steps as in Eq. (3). During training, clients provide their corresponding weight updates $\{\nabla_p\}_{p=1}^P$ and associated average losses $\{f_p := f(\boldsymbol{w};\ \mathcal{D}_p)\}_{p=1}^P$. Loss values from unreliable clients may not be accurate. After the training phase, a subset $\mathcal{S}(s)$ of P clients is selected and provided with resources to correct the labels within their respective datasets, generating the updated set $\{\mathcal{D}_p(s+1)\}_{p=1}^P$.

At each stage s, our algorithm, Gaussian process-based AEC (*GPAEC*), incrementally constructs $\mathcal{S}(s)$, starting from an empty set $\mathcal{S}(s) = \emptyset$. During each step c within stage s, the algorithm expands $\mathcal{S}(s)$ by incorporating a client, employing the following score:

$$v(p) = v(p;\ \mathcal{H}_p(s)) + v(p;\ \mathcal{S}(s)), \tag{9}$$

where $\mathcal{H}_p(s)$ is defined as $\{\nabla_1, \ldots, \nabla_{p-1}, \nabla_{p+1}, \ldots, \nabla_P\}$. The kernel parameter a (Eq. (6)) is automatically determined as the reciprocal of the average squared distance between $\{\nabla_p\}_{p=1}^P$ and their mean, following (Hein and Maier 2007; Kitayama and Yamazaki 2011). The constant σ^2 (Eq. (5)) is fixed to a small positive value of 10^{-8}. This ensures that $(\boldsymbol{K}_p + \sigma^2 \boldsymbol{I})^{-1}$ remains unconditionally well-defined. While fine-tuning a and σ^2 for specific problems and datasets could improve performance, optimizing hyperparameters in distributed learning is challenging due to the limited data available on the server.

The first term in Eq. (9) quantifies the inconsistency of client p with respect to the entire client set, while the second term captures inconsistency with respect to already corrected clients. Including the second term promotes diversity within $\mathcal{S}(s)$: Since clients generally have non-i.i.d. data, the relative similarities between a client and others may vary. Therefore, it is beneficial to select clients that are not only inconsistent overall but also distinct from those already included in $\mathcal{S}(s)$. Ideally, if only one client is selected at each stage ($|\mathcal{S}(s)| = 1$), such diversification would be unnecessary. However, this would require frequent network retraining, resulting in prohibitively high computational costs.

To evaluate the score v at each stage s, our algorithm performs a random initialization of $\boldsymbol{w}$ and collects the corresponding weight gradients $\{\nabla_p\}_{p=1}^P$ for $R = 10$ iterations to construct $\mathcal{H}_p(s)$: The server distributes random seed values for $\boldsymbol{w}$ to clients. The resulting kernel evaluations $\{\boldsymbol{k}_p\}$ and $\{\boldsymbol{K}_p\}$ are then averaged to yield the final scores v.

[3]For a rigorous analysis of this smoothing effect based on the *equivalent kernels* of GPs, refer to (Sollich and Williams 2004).

Our initial approach to constructing $\mathcal{S}(s)$ was to incrementally select clients with the highest v values. However, we observed that this approach inadvertently included a substantial number of unreliable clients in $\mathcal{S}(s)$. While these clients might use the allocated resources to improve their own labels, there is a risk that they could deliberately provide gradients based on incorrect labels. Therefore, including unreliable clients would result in the ineffective use of the error correction resource.

Our algorithm addresses this by combining $\{v(p)\}_{p=1}^P$ with the losses $\{f_p\}_{p=1}^P$ shared by the clients, which are calculated during stage $s-1$. First, we regularize $\{f_p\}_{p=1}^P$ using $\boldsymbol{a}_p$ (Eq. (7)): $\overline{f}_p = \boldsymbol{f}_p^\top \boldsymbol{a}_p$, where $\boldsymbol{f}_p = [f_1, \ldots, f_{p-1}, f_{p+1}, \ldots, f_P]^\top$. This process essentially reconstructs f_p based on the kernel smoothing of the other values in $\{f_p\}_{p=1}^P$ (equivalent to replacing $\boldsymbol{k}_p$ with $\boldsymbol{f}_p$ in $1 - v(p; \mathcal{H}_p)$; Eq. (8)). We hypothesize that clients showing a significant discrepancy between their original losses $\{f_p\}_{p=1}^P$ and regularized losses $\{\overline{f}_p\}_{p=1}^P$ are likely to be unreliable. Such clients may deliberately communicate random or artificially high loss values in an attempt to be selected for labeling. To combine these deviations with the inconsistencies in $\{v(p)\}_{p=1}^P$, accounting for potential differences in their scaling patterns, we use individual rankings:

$$v'(p) = \text{Rank}(\{v(p)\}, p) + \text{Rank}(\{-|f_p - \overline{f}_p|\}, p), \tag{10}$$

where $\text{Rank}(Z, p)$ calculates the rank of client p based on the decreasing order of values in Z. At stage s, the $L = |\mathcal{S}(s)|$ clients with the highest v'-values are selected for correction.

4.4 Efficient Implementation

Naively evaluating the client scores $\{v'(p)\}_{p=1}^P$ incurs a computational cost of $O(MP^2 + P^4)$ and a memory cost of $O(MP + P^2)$, where M is the dimensionality of the weight vector $\boldsymbol{w}$, and P is the number of clients. Our original algorithm explicitly stores the weight gradients $\{\nabla_p\}_{p=1}^P$ to calculate the kernel matrices $\{\boldsymbol{K}_p\}_{p=1}^P$, requiring $O(MP + P^2)$ memory and $O(MP^2)$ computation. Additionally, computing $\{v(p, \mathcal{H}_p)\}_{p=1}^P$ entails solving a linear system of size $P-1$ for each p (Eq. (5)), resulting in a total computational cost of $O(P^4)$. These costs can be prohibitively high, even in environments with a moderately large number of clients.

Our final algorithm avoids storing and calculating the pairwise distances for the entire gradient vectors by randomly sampling elements from these gradients. In each of the 10 gradient collection steps ($R = 10$), we generate a random index subset $\mathcal{I}$ from $\{1, \ldots, M\}$. The corresponding kernels $\{\boldsymbol{k}_p\}$ and $\{\boldsymbol{K}_p\}$ are then computed based on the sampled gradients $\{\overline{\nabla}_p\}_{p=1}^P$, where $\overline{\nabla}_p = [\nabla_{p[\mathcal{I}(1)]}, \ldots, \nabla_{p[\mathcal{I}(\overline{M})]}]^\top$. The size $\overline{M}$ of the subset $\mathcal{I}$ is fixed at 2,000 for all experiments. This also improves communication costs.

Additionally, the client inconsistency $\{v(p, \mathcal{H}_p)\}_{p=1}^P$ can be efficiently calculated through a rank-one update of the Cholesky factorization. Suppose that $\boldsymbol{K} \in \mathbb{R}^{P \times P}$ is the matrix of all pairwise kernel evaluations: $\boldsymbol{K}_{[ij]} = k(\nabla_i, \nabla_j)$ for $i, j = 1, \ldots, P$ and $\boldsymbol{L}$ is the lower triangular Cholesky factor of $\boldsymbol{K} + \sigma^2 \boldsymbol{I}$. Then, $v(p, \mathcal{H}_p)$ can be determined as

Algorithm 1: Main algorithm: P clients with datasets $\{\mathcal{D}_p\}_{p=1}^P$ share model gradients $\{\nabla_p\}_{p=1}^P$ and loss values $\{f_p\}_{p=1}^P$ upon request. Parameters: number of stages S, selected clients per stage L, and weight generation steps R.

$\mathcal{S} = \emptyset$.
for $s \in 1, \ldots, S$ **do**
  for $r \in 1, \ldots, R$ **do**
    Generate and distribute a random seed for $\boldsymbol{w}$; collect gradients $\mathcal{H} = \{\nabla_p\}_{p=1}^P$.
    Compute the kernel $\boldsymbol{K}$ from $\mathcal{H}$ and its Cholesky factor $\boldsymbol{L}$.
    for $p \in 1, \ldots, P$ **do**
      Construct $\boldsymbol{K}_p$, $\boldsymbol{k}_p$, and $\boldsymbol{L}_p$.
        {Submatrices derived from $\boldsymbol{K}$ and $\boldsymbol{L}$}
      Calculate $v(p; \mathcal{H}_p)$ using $\boldsymbol{L}_p$ and $\boldsymbol{k}_p$.
    end for
    Accumulate scores $\{v(p; \mathcal{H}_p)\}_{p=1}^P$ and update $\boldsymbol{K}$.
  end for
  Train the main network and compute regularized losses $\{\overline{f}_p\}_{p=1}^P$.
    {Use $\{\boldsymbol{L}_p\}_{p=1}^P$ as in $\{v(p; \mathcal{H}_p)\}$ calculation}
  $\mathcal{S}(s) = \emptyset$.
  for $c \in 1, \ldots, L$ **do**
    Compute $\{v'(p)\}_{p=1}^P$ from $\{v(p; \mathcal{S}(s))\}_{p=1}^P$.
      {Similar to $\{v(p; \mathcal{H}_p)\}_{p=1}^P$ calculations}
    $\mathcal{S}(s) \leftarrow \mathcal{S}(s) \cup \operatorname{arg\,min}_{p=1,\ldots,P} v'(p)$.
  end for
  Allocate error correction budget to clients in $\mathcal{S}(s)$.
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}(s)$.
end for

$\|(\boldsymbol{L}_p)^{-1}\boldsymbol{k}_p\|^2$, where $\boldsymbol{L}_p$ is the Cholesky factor of a submatrix $\boldsymbol{K}_p$ of $\boldsymbol{K}$, obtained by removing the p-th row and column from $\boldsymbol{K}$. Given $\boldsymbol{L}_p$, $\|(\boldsymbol{L}_p)^{-1}\boldsymbol{k}_p\|^2$ can be evaluated using backward substitution in $O(P^2)$ time. Furthermore, $\boldsymbol{L}_p$ can be constructed from $\boldsymbol{L}$ using a Cholesky update, as follows: $\boldsymbol{L}$ is structured as

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_{[1:p-1,1:p-1]} & 0 & 0 \\ \boldsymbol{L}_{[p,1:p-1]} & \boldsymbol{L}_{[p,p]} & 0 \\ \boldsymbol{L}_{[p+1:P,1:p-1]} & \boldsymbol{L}_{[p+1:P,p]} & \boldsymbol{L}_{[p+1:P,p+1:P]} \end{bmatrix}, \quad (11)$$

where $\boldsymbol{L}_{[l:m,n:o]}$ represents the submatrix of $\boldsymbol{L}$ obtained by extracting rows from l to m and columns from n to o. With this structure, $\boldsymbol{L}_p$ can be expressed as

$$\boldsymbol{L}_p = \begin{bmatrix} \boldsymbol{L}_{[1:p-1,1:p-1]} & 0 \\ \boldsymbol{L}_{[p+1:P,1:p-1]} & \overline{\boldsymbol{L}}_{[p+1:P,p+1:P]} \end{bmatrix}, \quad (12)$$

where $\overline{\boldsymbol{L}}_{[p+1:P,p+1:P]}$ is the rank-one Cholesky update of $\boldsymbol{L}_{[p+1:P,p+1:P]}$ achieved by adding $\boldsymbol{L}_{[p+1:P,p]}^\top \boldsymbol{L}_{[p+1:P,p]}$, which takes $O(P^2)$-time (Seeger 2004). The final time and memory complexities of our algorithm are $O(\overline{M}P^2 + P^3)$ and $O(\overline{M}P + P^2)$, respectively, with $\overline{M}$ fixed at 2,000. Given gradients $\{\nabla_p\}_{p=1}^P$, computing $\{v'(p)\}$ for a single AEC stage requires approximately 25 seconds (on *CIFAR100*; see Sec. 5). This cost is negligible compared to model training, which typically takes several hours per stage. A detailed description is provided in Algorithm 1.

The final algorithm requires $O(P^2)$ memory and $O(P^3)$ computation per AEC stage. In large-scale settings ($P > 10^5$), this complexity can be mitigated by constructing a separate GP model for each client p using a smaller subset $\mathcal{H}_p$ ($|\mathcal{H}_p| \ll P$), e.g. selected via random data partitioning. Large-scale experiments demonstrating the effectiveness of this strategy are presented in the supplementary document.

5 Experiments

We prepared distributed learning environments with $P = 1,000$ clients using Kuzushiji49 (Clanuwat et al. 2018), CIFAR10 and CIFAR100 (Krizhevsky 2009), Tiny ImageNet (TImgNet) (Le and Yang 2015), Clothing 1M (C1M) (Xiao et al. 2015), and EMNIST (Cohen et al. 2017) datasets.

To simulate heterogeneous client environments, we adopted the data partitioning method from (McMahan et al. 2017), which constructs non-i.i.d. settings by dividing the dataset into class-wise shards. In the original 10-class setup, each client is assigned two class shards. We extended this method to datasets with more than 10 classes by assigning each client $\lceil 0.2 \times C \rceil$ randomly selected class shards, where C is the total number of classes.

In the initial phase of the AEC process, all 1,000 clients exhibited label errors introduced by stochastically modifying the ground-truth labels for selected data points. These modifications, with probabilities ranging from 0.1 to 1, simulated varying degrees of label error. To model the label change, a stochastic matrix $P_{\mathrm{n}} \in \mathbb{R}^{C \times C}$ was generated for each dataset following (Li et al. 2022b; van Rooyen and Williamson 2018; Han et al. 2018). The original class assignments of the selected data points were then transitioned to new classes based on P_{n}. The supplemental document presents results from experiments with datasets containing human labeling errors (Wei et al. 2022).

Our experimental setup included 200 unreliable clients for each dataset. When prompted, these clients submitted artificially inflated loss values to increase their likelihood of being selected for error correction. To simulate this, we first collected the loss values from rational clients, and then randomly sampled the loss values of unreliable clients from the third quartile of the rational loss distribution. In each AEC stage, $L = 40$ clients were selected. Upon selection, rational clients corrected all of their assigned labels, while unreliable clients retained their initial noisy labels, resulting in wasted labeling resources. We used a network architecture combining a pre-trained ResNet50 model on ImageNet with additional fully-connected layers. Each experiment was repeated 10 times per dataset, and statistical significance was evaluated using a t-test with a significance level of 0.05.

5.1 Main Results

To our knowledge, no existing algorithms are directly applicable to active error correction (AEC) in distributed learning environments. Most AEC methods are designed for centralized settings and rely on access to individual data points,

Error correction stage		1	3	5	7	9	11
Data	Algorithm	Results					
Kuzushiji49	*Rand*	50.04 (0.72)	51.37 (0.66)	52.46 (0.55)	53.24 (0.52)	54.04 (0.43)	54.88 (0.25)
	Loss	49.50 (0.67)	49.44 (0.60)	50.87 (0.63)	52.84 (0.53)	54.05 (0.55)	55.27 (0.57)
	Ent	49.80 (0.64)	50.53 (0.70)	51.93 (0.57)	52.99 (0.57)	54.01 (0.56)	54.96 (0.55)
	Uniform	50.02 (0.62)	51.31 (0.51)	52.31 (0.49)	53.36 (0.35)	54.23 (0.24)	54.94 (0.25)
	GPAEC	**50.69 (0.38)**	**52.58 (0.43)**	**54.03 (0.38)**	**54.96 (0.49)**	**55.72 (0.27)**	**56.32 (0.35)**
CIFAR10	*Rand*	82.05 (0.25)	82.90 (0.27)	83.43 (0.23)	83.91 (0.29)	84.44 (0.28)	84.74 (0.26)
	Loss	81.53 (0.32)	81.54 (0.46)	82.81 (0.24)	83.72 (0.35)	84.75 (0.22)	85.18 (0.23)
	Ent	81.51 (0.26)	81.92 (0.35)	82.51 (0.23)	83.27 (0.28)	84.17 (0.27)	84.49 (0.25)
	Uniform	82.02 (0.32)	82.75 (0.37)	83.56 (0.28)	84.10 (0.35)	84.44 (0.19)	85.02 (0.29)
	GPAEC	**82.29 (0.24)**	**83.56 (0.24)**	**84.39 (0.18)**	**84.83 (0.26)**	**85.17 (0.41)**	**85.64 (0.17)**
CIFAR100	*Rand*	54.35 (0.55)	55.44 (0.36)	56.75 (0.40)	57.82 (0.29)	58.67 (0.35)	59.34 (0.40)
	Loss	54.04 (0.42)	54.18 (0.65)	55.57 (0.53)	57.14 (0.40)	58.87 (0.51)	59.95 (0.38)
	Ent	53.80 (0.60)	54.58 (0.60)	55.53 (0.47)	56.93 (0.51)	57.95 (0.61)	59.19 (0.23)
	Uniform	54.23 (0.52)	55.79 (0.32)	56.70 (0.50)	57.84 (0.41)	58.70 (0.40)	59.46 (0.39)
	GPAEC	**54.60 (0.35)**	**56.31 (0.57)**	**57.88 (0.42)**	**58.76 (0.35)**	**59.72 (0.40)**	**60.51 (0.35)**
TImgNet	*Rand*	44.40 (0.51)	46.99 (0.35)	49.33 (0.41)	50.54 (0.41)	52.12 (0.36)	53.47 (0.51)
	Loss	44.30 (0.43)	46.16 (0.59)	48.98 (0.82)	51.76 (0.35)	53.36 (0.39)	54.17 (0.32)
	Ent	43.90 (0.57)	45.78 (0.65)	48.30 (0.52)	50.81 (0.40)	52.28 (0.52)	53.57 (0.23)
	Uniform	44.44 (0.71)	47.10 (0.46)	49.17 (0.46)	50.90 (0.49)	52.21 (0.46)	53.33 (0.39)
	GPAEC	**45.84 (0.30)**	**48.98 (0.42)**	**50.94 (0.46)**	**52.55 (0.31)**	**53.94 (0.37)**	**54.53 (0.43)**
C1M	*Rand*	49.08 (0.99)	50.23 (1.07)	51.54 (0.76)	52.43 (0.53)	53.17 (0.59)	53.99 (0.56)
	Loss	48.70 (0.70)	49.15 (1.02)	50.12 (0.92)	52.11 (0.58)	53.45 (0.61)	54.33 (0.63)
	Ent	48.68 (0.72)	49.61 (0.79)	49.98 (0.58)	51.12 (0.50)	51.65 (0.70)	52.58 (0.55)
	Uniform	49.33 (0.96)	50.39 (0.48)	51.72 (0.60)	52.37 (0.52)	52.83 (0.28)	53.93 (0.55)
	GPAEC	**51.19 (0.54)**	**52.62 (0.25)**	**53.89 (0.73)**	**54.81 (0.44)**	**55.28 (0.55)**	**55.85 (0.47)**
EMNIST	*Rand*	78.28 (0.39)	78.99 (0.35)	79.73 (0.31)	80.32 (0.29)	80.60 (0.27)	81.15 (0.17)
	Loss	78.08 (0.22)	78.62 (0.32)	79.35 (0.33)	79.93 (0.29)	80.84 (0.23)	81.51 (0.16)
	Ent	78.02 (0.25)	78.59 (0.41)	79.05 (0.31)	79.67 (0.20)	80.35 (0.21)	80.97 (0.23)
	Uniform	78.16 (0.47)	78.95 (0.27)	79.66 (0.25)	80.31 (0.33)	80.82 (0.15)	81.34 (0.21)
	GPAEC	**79.21 (0.26)**	**80.26 (0.21)**	**80.96 (0.14)**	**81.37 (0.13)**	**81.88 (0.24)**	**82.42 (0.17)**

Table 1: Mean accuracy (with standard deviation; in %) of active error correction algorithms. The best results are highlighted in **bold**, and the second-best are underlined. Statistical significance, in comparison to the baseline *Rand*, is indicated by blue (significant improvement) and red (significant deterioration), based on a t-test at the 0.05 significance level. Our method consistently achieves significant gains. Comprehensive results are provided in the supplementary document.

making them incompatible with the privacy constraints of distributed learning. As adapting such methods would require significant architectural changes and violate key assumptions of our problem setting, we instead focus on practical baselines that align with these constraints.

Specifically, we compare *GPAEC* with client-level adaptations of standard instance-level selection strategies from centralized AEC, modified to operate without accessing raw data. These include random selection (*Rand*), selecting L clients with the highest average loss values (*Loss*), and selection based on prediction entropy (*Ent*). Such strategies are common in centralized learning, where high-loss samples are often mislabeled and benefit from re-labeling (Huang et al. 2019), and high-entropy predictions indicate ambiguous or noisy labels (Rebbapragada et al. 2012). We also include a baseline (*Uniform*) that evenly distributes the labeling budget across all P clients without selection. While simple, this strategy can become impractical when labeling sessions extend over days or weeks and involve overhead such as recruiting annotators. This is especially relevant for privacy-sensitive data, where annotators cannot access raw data online.

During the correction stages, all algorithms exhibited a natural tendency toward increasing accuracy (Table 1). However, affected by the deceptive (high) loss values generated by unreliable clients, *Loss* and *Ent* displayed the lowest error correction efficiency in the early stages. *Loss* tended to show improved efficiency in later stages, when most unreliable clients had already been selected in the initial stages, allowing it to focus more effectively on clients with genuinely high loss values. *Ent* continued to show the lowest levels of accuracy throughout. *Rand* achieved slightly higher accuracy in the early stages but was limited by its tendency to frequently select clients with low loss values. *Uniform* showed similar levels of error correction efficiency as *Rand*.

In contrast, by combining model inconsistency with client loss values, our algorithm *GPAEC* effectively identified clients who were both rational and highly noisy. In the first stage, our algorithm selected 88.9% rational clients on average, compared to 78.6% for *Rand* and 42.2% for *Loss*.[4]

[4]With 20% of clients being unreliable in our experiments, repeated trials will show an 80% selection rate for rational clients using the *Rand* approach.

Error correction stage		1	3	5	7	9	11
Data	Algorithm	Results					
(a) *Kuzushiji49*	*Rand*	41.56 (0.94)	42.42 (0.84)	43.95 (0.55)	44.72 (0.71)	45.54 (0.53)	46.43 (0.91)
	Loss	41.11 (0.87)	40.90 (0.39)	42.51 (0.93)	43.94 (0.68)	46.32 (0.89)	46.73 (0.66)
	Ent	41.13 (1.06)	41.90 (0.91)	43.11 (0.61)	44.21 (0.88)	45.33 (0.65)	46.81 (0.82)
	Uniform	41.49 (1.09)	42.71 (0.88)	43.56 (0.92)	45.38 (1.11)	45.96 (0.61)	46.64 (0.53)
	GPAEC	**41.74 (0.75)**	**43.91 (0.73)**	**45.34 (0.58)**	**46.47 (0.71)**	**47.14 (0.46)**	**47.81 (0.57)**
(b) *CIFAR10*	*MeanDist*	81.65 (0.26)	82.07 (0.37)	82.64 (0.31)	83.25 (0.29)	83.74 (0.24)	84.47 (0.29)
	MedDist	81.98 (0.19)	82.60 (0.46)	83.38 (0.46)	83.94 (0.32)	84.38 (0.27)	85.08 (0.32)
	GPAEC	**82.29 (0.24)**	**83.56 (0.24)**	**84.39 (0.18)**	**84.83 (0.26)**	**85.17 (0.41)**	**85.64 (0.17)**
(c) *CIFAR100*	*Rand*	54.92 (0.44)	56.63 (0.42)	58.20 (0.53)	59.38 (0.35)	60.42 (0.17)	61.35 (0.41)
	Loss	54.22 (0.66)	55.52 (0.68)	58.28 (0.59)	60.13 (0.21)	61.45 (0.30)	61.58 (0.42)
	Ent	54.11 (0.25)	55.85 (0.45)	57.62 (0.45)	59.28 (0.46)	60.35 (0.40)	61.36 (0.30)
	Uniform	54.83 (0.33)	56.76 (0.32)	58.40 (0.38)	59.27 (0.20)	60.34 (0.43)	61.06 (0.40)
	GPAEC	**55.17 (0.45)**	**57.95 (0.49)**	**59.38 (0.37)**	**60.60 (0.52)**	**61.56 (0.47)**	**61.81 (0.37)**
(d) *TImgNet*	*Rand*	44.17 (0.39)	45.94 (0.54)	47.29 (0.69)	48.39 (0.58)	49.57 (0.31)	50.83 (0.47)
	Loss	43.65 (0.64)	44.45 (0.51)	45.70 (0.53)	47.70 (0.53)	50.02 (0.53)	51.71 (0.37)
	Ent	43.54 (0.61)	44.17 (0.42)	45.63 (0.35)	47.55 (0.49)	48.82 (0.34)	50.69 (0.15)
	Uniform	43.70 (0.53)	46.01 (0.37)	47.40 (0.32)	48.45 (0.38)	49.50 (0.44)	50.77 (0.46)
	GPAEC	**45.14 (0.69)**	**47.12 (0.62)**	**49.15 (0.59)**	**50.52 (0.40)**	**51.49 (0.58)**	**52.64 (0.18)**
(e) *C1M*	*Rand*	49.54 (0.88)	50.78 (0.67)	52.06 (0.72)	53.04 (0.53)	54.02 (0.40)	54.77 (0.54)
	Loss	49.24 (0.98)	51.15 (0.72)	52.38 (0.45)	54.13 (0.60)	55.08 (0.59)	55.58 (0.53)
	Ent	49.73 (1.04)	51.22 (0.70)	52.17 (0.66)	52.83 (0.68)	53.63 (0.51)	54.40 (0.37)
	Uniform	49.53 (0.87)	50.74 (0.39)	52.28 (0.48)	53.13 (0.34)	53.73 (0.34)	54.79 (0.74)
	GPAEC	**51.34 (0.50)**	**53.02 (0.28)**	**54.17 (0.61)**	**55.17 (0.44)**	**55.72 (0.57)**	**56.33 (0.50)**

Table 2: Ablation results. (a) small convolutional networks with three convolutional layers of sizes 32, 64, and 64 were used; (b) mean (*MeanDist*) and median (*MedDist*) were used for $\boldsymbol{r}$ in Eq. (4) to define the inconsistency measure u; (c) and (d) in each stage, 60 and 20 clients were selected for error correction, respectively; and (e) unreliable clients transferred disproportionately high loss values, potentially manipulating the selection probability.

5.2 Ablation Study

Table 2 presents additional experiments examining different aspects of our algorithm. More comprehensive ablation studies are available in the supplementary document.

(a) Alternative network architecture. We used a network comprising three convolution layers with 32, 64, and 64 filters, each using 5×5 kernels, followed by max-pooling layers. The network was completed with two fully-connected layers.

(b) Different inconsistency measures. We explored variations of our algorithm by measuring client gradient distances from both the mean (*MeanDist*) and median (*MedDist*) gradients as inconsistency measures, with mean and median used for $\boldsymbol{r}$ in Eq. (4).

(c,d) Varying number of selected clients. We evaluated the impact of varying the number of selected clients L per stage, with $L = 60$ in (c) and $L = 20$ in (d).

(e) Different form of unreliable clients. In this scenario, upon receiving a labeling correction budget, unreliable clients cleaned their labels and provided an improved model update. However, they deliberately inflated their average loss values to enhance their chances of selection. While the overall performance of *Loss* showed significant improvement, particularly in the middle to later stages, it continued to face distractions from unreliable clients in the early stages, leading to inferior results compared to *Rand* in the initial stages.

In summary, the outcomes of all algorithms exhibited substantial variability across different conditions. Nevertheless, the consistent superiority of our algorithm remained evident, demonstrating its robustness and adaptability across diverse testing environments.

6 Conclusions

This work presents, to the best of our knowledge, the first approach to active error correction (AEC) in distributed collaborative learning, where the coordinator does not have access to client-local data. We propose a client selection algorithm based on gradient inconsistency and diversity, instantiated through a Gaussian process model and efficiently implemented using a rank one update. The method significantly improves error correction efficiency and outperforms naive adaptations of centralized AEC.

While our strategy is intuitive and empirically effective, a formal theoretical connection between gradient inconsistency, diversity, and optimal label correction remains an open question, similar to most existing (centralized) AEC frameworks. Future work could explore such foundations.

Another limitation concerns benchmarking. Existing AEC methods rely on centralized access and are fundamentally incompatible with our distributed setting. In the absence of prior methods for distributed AEC, we compare against simple client level strategies that do not require raw data. Future research may investigate privacy preserving adaptations of centralized methods to enable more meaningful comparisons within distributed environments.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant (No. RS-2024-00337559) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants (No. RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH); No. RS-2022-II220290, Visual Intelligence for Space-Time Understanding and Generation), all funded by the Korean government (MSIT).

References

Bernhardt, M.; Castro, D. C.; Tanno, R.; Schwaighofer, A.; Tezcan, K. C.; Monteiro, M.; Bannur, S.; Lungren, M.; Nori, A.; Glocker, B.; Alvarez-Valle, J.; and Oktay, O. 2022. Active label cleaning for improved dataset quality under resource constraints. *Nature Communications*, 13.

Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; and Ha, D. 2018. Deep learning for classical Japanese literature. In *arXiv:1812.01718*.

Cohen, G.; Afshar, S.; Tapson, J.; and van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. In *IJCNN*, 2921–2926.

Deisenroth, M. P.; Huber, M. F.; and Hanebeck, U. D. 2009. Analytic moment-based Gaussian process filtering. In *ICML*, 225–232.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I. W.; and Sugiyama, M. 2018. Co-teaching: robust training of deep neural networks with extremely noisy labels. In *NeurIPS*.

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

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.

Huang, H.; Shi, W.; Feng, Y.; Niu, C.; Cheng, G.; Huang, J.; and Liu, Z. 2024. Active client selection for clustered federated learning. *IEEE TNNLS*.

Huang, J.; Qu, L.; Jia, R.; and Zhao, B. 2019. O2U-Net: A simple noisy label detection approach for deep neural networks. In *ICCV*, 3326–3334.

Huang, Y.; Gupta, S.; Song, Z.; Li, K.; and Arora, S. 2021. Evaluating gradient inversion attacks and defenses in federated learning. In *NeurIPS*.

Ji, X.; Zhu, Z.; Xi, W.; Gadyatskaya, O.; Song, Z.; Cai, Y.; and Liu, Y. 2024. FedFixer: Mitigating heterogeneous label noise in federated learning. In *AAAI*.

Jiang, X.; Sun, S.; Li, J.; Xue, J.; Li, R.; Wu, Z.; Xu, G.; Wang, Y.; and Liu, M. 2024. Tackling noisy clients in federated learning with end-to-end label correction. In *CIKM*.

Kandasamy, K.; Schneider, J.; and Póczos, B. 2015. Bayesian active learning for posterior estimation. In *IJCAI*.

Karimireddy, S. P.; Guo, W.; and Jordan, M. 2022. Mechanisms that incentivize data sharing in federated learning. In *NeurIPS Workshop on Federated Learning*.

Kitayama, S.; and Yamazaki, K. 2011. Simple estimate of the width in Gaussian kernel with adaptive scaling technique. *Applied Soft Computing*, 11(8).

Koloskova, A.; Lin, T.; Stich, S. U.; and Jaggi, M. 2020. Decentralized deep learning with arbitrary communication compression. In *ICLR*.

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. *JMLR*, 9.

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

Le, Y.; and Yang, X. 2015. Tiny ImageNet visual recognition challenge. Technical report, Stanford University.

Li, H.; Funk, M.; Gürel, N. M.; and Saeed, A. 2024. Collaboratively learning federated models from noisy decentralized data. In *BIGDATA*.

Li, P.; Zhao, Y.; Chen, L.; Cheng, K.; Xie, C.; Wang, X.; and Hu, Q. 2022a. Uncertainty measured active client selection for federated learning in smart grid. In *Proc. International Conference on Smart Internet of Things*, 148–153.

Li, S.-Y.; Shi, Y.; Huang, S.-J.; and Chen, S. 2022b. Improving deep label noise learning with dual active label correction. *Machine Learning*.

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*.

Nallapati, R.; Surdeanu, M.; and Manning, C. 2009. CorrActive learning: Learning from noisy data through human interaction. In *IJCAI Workshop on Intelligence and Interaction*.

Németh, G. D.; Ángel Lozano, M.; Quadrianto, N.; and Oliver, N. 2022. A snapshot of the frontiers of client selection in federated learning. *TMLR*.

Ovi, P. R.; Dey, E.; Roy, N.; and Gangopadhyay, A. 2023. Mixed quantization enabled federated learning to tackle gradient inversion attacks. In *CVPR Workshops*.

Park, S.; Jo, D. U.; and Choi, J. Y. 2021. Over-fit: Noisy-label detection based on the overfitted model property. In *arXiv:2106.07217*.

Rasmussen, C. E.; and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Rebbapragada, U.; Brodley, C. E.; Sulla-Menashe, D.; and Friedl, M. A. 2012. Active label correction. In *ICDM*, 1080–1085.

Seeger, M. 2004. Low rank updates for the Cholesky decomposition. Technical report, University of California, Berkeley.

Seo, S.; Wallat, M.; Graepel, T.; and Obermayer, K. 2000. Gaussian process regression: active data selection and test point rejection. In *IJCNN*.

Sollich, P.; and Williams, C. K. I. 2004. Using the equivalent kernel to understand Gaussian process regression. In *NIPS*.

Tan, A. Z.; Yu, H.; Cui, L.; and Yang, Q. 2023. Towards personalized federated learning. *IEEE TNNLS*, 34(12): 9587–9603.

Tang, M.; Ning, X.; Wang, Y.; Sun, J.; Wang, Y.; Li, H.; and Chen, Y. 2022. FedCor: Correlation-based active client selection strategy for heterogeneous federated learning. In *CVPR*.

Tsouvalas, V.; Saeed, A.; Ozcelebi, T.; and Meratnia, N. 2024. Labeling chaos to learning harmony: federated learning with noisy labels. *ACM TIST*, 15(2).

Turan, B.; Uribe, C. A.; Wai, H.-T.; and Alizadeh, M. 2021. Robust distributed optimization with randomly corrupted gradients. In *arXiv:2106.14956*.

van Rooyen, B.; and Williamson, R. C. 2018. A theory of learning with corrupted labels. *JMLR*, 18.

Wei, J.; Zhu, Z.; Cheng, H.; Liu, T.; Niu, G.; and Liu, Y. 2022. Learning with noisy labels revisited: a study using real-world human annotations. In *ICLR*.

Xiao, T.; Xia, T.; Yang, Y.; Huang, C.; and Wang, X. 2015. Learning from massive noisy labeled data for image classification. In *CVPR*.

Xu, J.; Chen, Z.; Quek, T. Q. S.; and Chong, K. F. E. 2022. FedCorr: Multi-Stage Federated Learning for Label Noise Correction. In *CVPR*.

Zhu, B.; Wang, L.; Pang, Q.; Wang, S.; Jiao, J.; Song, D.; and Jordan, M. I. 2023. Byzantine-robust federated learning with optimal statistical rates. In *AISTATS*.