A Computationally Efficient Method for Dataset Quality Evaluation based on the Coverage of the Dataset

Beomjun Kim MIT

Jaehwan Kim* AIMMO

Kangyeon Kim KAIST

Sunwoo Kim SNU

Heejin Ahn KAIST

Abstract

Evaluating dataset quality is an essential task, as the performance of artificial intelligence (AI) systems heavily depends on it. A traditional method for evaluating dataset quality involves training an AI model on the dataset and testing it on a separate test set. However, this approach requires significant computational time. In this paper, we propose a computationally efficient method for quantifying dataset quality. Specifically, our method measures how well the dataset covers the input probability distribution, ensuring that a high-quality dataset minimizes out-of-distribution inputs. We present a GPU-accelerated algorithm for approximately implementing the proposed method. We highlight three applications of our approach. First, it can evaluate the impact of data management practices, such as data cleaning and core set selection. We experimentally demonstrate that the quality assessment provided by our method strongly correlates with the traditional approach, achieving an $R^2 \geq 0.985$ in most cases while being 60-1200 times faster. Second, it can monitor the quality of continuously growing datasets with computation time proportional to the added data size. Finally, our method can estimate the performance of traditional methods for large datasets. our method is based on an intuitive concept similar to metric entropy and core set selection. However, the novelty of our method is to propose a systematic data set quality assessment that can be used to estimate training performance. This idea, linked with GPU-acceleration can be considerably helpful for researchers dealing with large datasets or continuously growing datasets.

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

1 Introduction

Data plays the most critical role in artificial intelligence (AI) systems. For example, machine learning (ML) models are trained using datasets, and consequently, their quality is limited by the quality of datasets used in training. The importance of data has led to significant research attention to data-centric AI, aiming to produce the best dataset for ML models. For example, data cleaning for training datasets has been proven effective [Krishnan et al., 2016] to achieve better performance of the trained model.

One of the important questions that should be developed simultaneously is how one can determine whether a dataset is good. A common approach to evaluating the quality of a dataset [Moreno-Barea et al., 2020,Guo et al., 2022,Sener and Savarese, 2018] is to train a neural network with the dataset and evaluate the network with a separate test dataset. However, it requires a large amount of time to train a neural network, often increasing superlinearly with the dataset size. This can become a burden in data management tasks like data cleaning. Moreover, additional training is required whenever datasets are updated. The repeated need to retrain the network to assess dataset quality can result in prohibitively high computation costs.

In this paper, we propose a computationally efficient algorithm for evaluating the quality of a dataset. In particular, we propose a quality function, mapping a dataset to a value in $[0, 1]$, based on how well it covers the distribution of input data. We also present a computationally efficient algorithm for approximately computing the quality function. We demonstrate three applications of our proposed method, compared to the traditional evaluation method [Moreno-Barea et al., 2020,Guo et al., 2022,Sener and Savarese, 2018]. First, it can assess the results of data management tasks, such as data cleaning. Second, it supports real-time quality monitoring for continuously updating datasets. It is particularly useful for companies that manage active data streams from their services. Finally, our method can estimate the test accuracy of an AI model trained with a large dataset by evaluating that with small random samples.

Our contributions are summarized as follows.

- We propose a quality function that quantifies how well a dataset covers the input data distribution.
- We develop a computationally efficient algorithm that approximately implements the dataset quality function. We also provide a GPU-accelerated code.
- We show that our method can be used to evaluate the results of data management, with empirical examples of data cleaning [Chu et al., 2016] and core set selection [Sener and Savarese, 2018].
- We empirically validate that our method can monitor the quality of a continuously growing dataset in real-time.
- We show that our method can estimate the test accuracy of an AI model trained with a large dataset based on that with small random samples.

Related Work

Data-centric AI Data-centric AI is a movement that highlights systematic data management [Zha et al., 2023b, Zha et al., 2023a] in AI systems. Many works have contributed to data development and maintenance. Improvements in data collection [Bogatu et al., 2020] and labeling [Zoph et al., 2020, Dekel and Shamir, 2009], synthetic data generation [Chen et al., 2020, Frid-Adar et al., 2018], and others have been proposed to obtain better datasets. Moreover, for data maintenance, researchers have dealt with outlier detection [Lai et al., 2021] and data cleaning [Chu et al., 2016, Krishnan et al., 2016, Zhang, 2016]. There have also been studies on data visualization [Borkin et al., 2013, Burch and Weiskopf, 2013], storage [Herodotou et al., 2011], and pipelines [Drori et al., 2021, Heffetz et al., 2020]. In other side, to reduce the storage and computation time, data selection [Sener and Savarese, 2018, Amagata, 2024] and condensation [Shin et al., 2023] are actively being investigated. To efficiently select the subset of the dataset, data valuation [Li and Yu, 2024, Wang et al., 2024] is also being studied. Note that data valuation measures the value of a data point in a dataset, whereas in this paper we focus on measuring the quality of a dataset.

Dataset quality measure Dataset quality measures have mostly been studied in management [Pipino et al., 2002] or deterministic computing [Batini et al., 2009]. However, these works do not apply to ML because datasets in these works mainly refer to measured values or statistics of important factors, such as sales rate and yield rate, whereas datasets for ML refer to samples of the real world. Other works [Gong et al., 2023, Budach et al., 2022] empirically explore data quality statistics, such as timeliness, completeness, and unbiasedness. Even though these works deal with datasets for ML, they focus on statistical data and do not indicate how the dataset can train or test AI models well. There are prior works [Moreno-Barea et al., 2020, Guo et al., 2022] that evaluate the quality of a dataset by training an ML model and testing its accuracy with a trustworthy dataset. They have a major drawback in that training of AI model requires a long time, even often superlinear to the dataset size. There are also prior works [Garima et al., 2020, Koh and Liang, 2017, Park et al., 2023] for measuring the value of each data point within a dataset. Although they can be adapted to evaluate dataset quality, evaluateing the effect of each individual data point and summing these values leads to high variance and requires large computation time. Comparison with them is provided in the experiment section. To the best of our knowledge, no existing work provides an evaluation method for dataset quality in terms of its ability to cover the input probability distribution and assess the risk of out-of-distribution scenarios.

Quantities of information There have been discussions to quantify the information in information theory. Shannon entropy [Shannon, 1948] quantifies the amount of information in symbolic communication. As an extension of Shannon entropy to the metric space, metric entropy [Lorentz, 1964] is an indicator of the required number of metric balls to cover the metric space or its subset. The metric entropy and its approximation [Lorentz, 1966] are also used in information theory. Since the metric entropy can be interpreted as how many points are needed to compress a given metric space under a given maximal error, this concept is well applied in data compression [Donoho et al., 1998]. Our proposed ζ-coverage concept (defined in Section 2) is similar to the concept of metric entropy. The difference is that we consider the probability measure of the space and provide an efficient algorithm to estimate the quality.

2 Problem setup

This paper proposes a method to measure the quality of a dataset. In this section, we introduce the notations to define a quality function and provide conditions it should satisfy.

Let Ω be the set of all feasible inputs (without label) to a machine learning model and L be the space of possible outputs. Here, we focus on cases where the cardinality of L, denoted by $|L|$, is finite; for example, L can be the set of possible labels for classification tasks. The probability distribution of the input data that a dataset should represent is expressed by a probability measure $P : \Sigma \to [0, 1]$ with a σ-algebra Σ of $\Omega \times L$. A dataset D is a finite submultiset of $\Omega \times L$. We assume that Ω is compact and is a pseudometric space. Let the pseudometric be $M(\omega_1, \omega_2)$ for $\omega_1, \omega_2 \in \Omega$.

To quantify the quality of a dataset, we introduce a function g, which maps subsets of the dataset to a value between 0 and 1, that is, $g : \{X \in N^{\Omega\times L} | |X| < \infty\} \to [0, 1]$, where $N^{\Omega\times L}$ is a set of submultisets of $\Omega \times L$. We call this function *a quality function* if it satisfies the following condition.

Condition 1 *When we sample dataset D following P, then $Pr\left(\lim_{|D|\to\infty} g(D) = 1\right) = 1$.*

This condition means that the quality of an ideal dataset, which can be constructed by sampling an infinite number of data following P, converges to 1.

There can be many quality functions, such as the ones that quantify completeness and label correctness. In this paper, we focus on a specific quality function that reflects how well a dataset covers the input distribution. For a quality function g, we call it *ensures coverage* if it satisfies the following condition.

Condition 2 *For any positive real number r, there exists $\bar{g} \in \mathbb{R}$ such that any dataset D with $g(D) \geq \bar{g}$ satisfies*

$$Pr_{(\omega,l)\sim P} \left(\exists(\omega_1, l) \in D, M(\omega, \omega_1) \leq r\right) = 1. \tag{1}$$

The condition indicates that when the quality function value is sufficiently high, any input data following distribution P will have a similar counterpart within the evaluated dataset D with a probability of 1. In other words, there is no risk of encountering out-of-distribution data points, meaning that all potential inputs are well-represented within D.

The goal of this paper is to design a coverage-ensuring quality function, i.e., one that satisfies Conditions 1 and 2. In the next section, we construct a function q and prove that it is a coverage-ensuring quality function.

3 Proposed Quality Function

In this section, we propose a function that satisfies Conditions 1 and 2. To formalize this, we first define some notations.

We begin by defining the label-wise probability measure $P_l : \Omega \to [0, 1]$ as the probability measure on the problem space Ω associated with each label l. Assume that P_l is well-defined on a σ-algebra that includes all open sets. Given a dataset $D \in N^{\Omega\times L}$, define $D_l \in N^{\Omega}$ as a part of D associated with label l.

We use the pseudometric $M(\omega_1, \omega_2)$ for $\omega_1, \omega_2 \in \Omega$ to define *the dataset coverage*. Note that the pseudometric $M(\omega_1, \omega_2)$ is not restricted to Euclidean norms, and domain experts can define it as containing sufficient information about important features of $\omega \in \Omega$. For example, in an experiment in Section 5, we compute $M(\omega_1, \omega_2)$ by first using EfficientNetV2 [Tan and Le, 2021] to extract features of ω_1 and ω_2 and computing the Euclidean distance of the features.

Now, to measure how well a dataset covers the input distribution, we define the concept of dataset coverage.

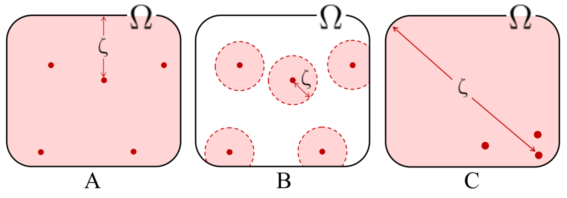

Figure 1: Conceptual example of coverage. A: Large ζ, B: small ζ, and C: a low-quality ζ-coverage dataset.

Definition 1 *For $\zeta > 0$, we call a dataset D_l ζ-coverage if and only if D_l satisfies the following:*

$$P_l\left(\cup_{\omega\in D_l} B(\omega, \zeta)\right) = 1, \tag{2}$$

where $B(\omega, \zeta)$ is an open ball with radius ζ centered at ω.

The definition of ζ-coverage is illustrated in Figure 1, assuming that P_l is a uniform probability distribution. The red dots represent data points in D_l, and the red shaded area is $\cup_{\omega\in D_l} B(\omega, \zeta)$. When ζ is large as in Figure 1-A, compared to ζ in Figure 1-B, we have large $P_l(\cup_{\omega\in D_l} B(\omega, \zeta))$ and the dataset is likely to be ζ-coverage. To reduce the size of ζ while remaining ζ-coverage, $|D_l|$ should be high and also contain data points in low probability regions. Figure 1-C is an example of a dataset that requires an extremely large ζ to be ζ-coverage because the data points are biased and we need a large radius to satisfy (2).

Based on the concept of ζ-coverage, we now introduce a candidate for a coverage-ensuring quality function.

Definition 2 *For a label-wise dataset D_l, we define the function q as*

$$q(D_l) := \frac{2}{\pi} arccot \left(\frac{\inf(\{\zeta | D_l : \zeta\text{−coverage}\})}{\mathrm{c}} \right) \quad (3)$$

if $D_l \neq \phi$ and $q(D_l) = 0$ otherwise. Here, c is a hyperparameter. In addition, for a dataset D, we define the function $q(D)$ as $q(D) := \frac{\sum_l q(D_l)}{|L|}$.

In the definition of $q(D_l)$, we use $\frac{2}{\pi}\text{arccot}\left(\frac{r}{c}\right)$ as a nonlinear mapping function, to map radius ζ in $(0, \infty)$ to a quality value in $[0, 1]$ and to map a low radius to a larger quality value. A parameter c determines the shape of the mapping.

The function $q(D)$ can often be too small, that is, $\inf(\{\zeta | D_l : \zeta\text{−coverage}\}$ is too large; for example, when the probability measure P has a long tail, $q(D)$ tries to cover the long tail and significantly undervalue the dataset. To avoid such cases, we relax the definition of ζ−coverage in (2), to require the probability $1 - \epsilon$ with some margin ϵ, instead of 1. In the following, we define ϵ-quasi-ζ-coverage and the corresponding ϵ-quasi functions.

Definition 3 *For $\zeta > 0$ and $0 < \epsilon < 1$, we call D_l ϵ-quasi-ζ-coverage if and only if D_l satisfies the following:*

$$P_l(\cup_{\omega \in D_l} B(\omega, \zeta)) \geq 1 - \epsilon. \quad (4)$$

Definition 4 *For a label-wise dataset D_l and given $0 < \epsilon < 1$, we define the ϵ-quasi function $q_\epsilon(D_l)$ as*

$$q_\epsilon(D_l) := \frac{2}{\pi} arccot \left(\frac{\inf(\{\zeta | D_l : \epsilon\text{-}quasi\text{-}\zeta\text{-}coverage\})}{c} \right) \quad (5)$$

if $D_l \neq \phi$ and $q_\epsilon(D_l) = 0$ otherwise. In addition, for a dataset D, we define the ϵ-quasi function $q_\epsilon(D)$ as $q_\epsilon(D) := \frac{\sum_{l \in L} q_\epsilon(D_l)}{|L|}$.

Note that these functions are well-defined since $0 \leq q(D_l), q(D), q_\epsilon(D_l), q_\epsilon(D) \leq 1$ by the definitions. We prove that the functions are coverage-ensuring quality functions in the following theroem. All the proofs are provided in the supplementary material.

Theorem 1 *The proposed function $q(D)$ is a coverage-ensuring quality function. That is, it satisfies Condition 1 and 2.*

Also, for any $0 < \epsilon < 1$, $q_\epsilon(D)$ satisfies Condition 1. Moreover, for any positive real number r, there exists $\bar{g} \in \mathbb{R}$ such that any dataset D with $g(D) \geq \bar{g}$ satisfies

$$Pr_{(\omega,l)\sim P}\left(\exists(\omega_1, l) \in D, M(\omega, \omega_1) \leq r\right) \geq 1 - \epsilon. \quad (6)$$

According to Theorem 1, q and q_ϵ yield the value of 1 for an infinite number of data sampled from the input distribution P. Also, if $q(D)$ yields a high value, the dataset D covers the support of P with a probability of 1, and if $q_\epsilon(D)$ yields a high value, D covers a region of probability at least $1 - \epsilon$.

4 Computation of the Proposed Dataset Quality Function

To compute the functions $q(D_l)$ and $q_\epsilon(D_l)$ for a given label-wise dataset D_l, we use a separate test dataset, which the traditional evaluation method used in [Moreno-Barea et al., 2020, Guo et al., 2022, Sener and Savarese, 2018] also assumes to have. We denote test data with label l as $T_l := \{t_{l1}, \ldots, t_{ln_l}\}$ and define R as the set of minimum distances between an element of the test dataset and an element of D_l, that is, $R = \{\min_{\omega \in D_l} M(\omega, t_{li}) | i = 1, \ldots, n_l\}$. Let the ordered elements of R be denoted as $r_{l1} \leq \cdots \leq r_{ln_l}$. With the minimum distances, we can compute the confidence interval of the quality function q_ϵ in the following.

Proposition 1 *For any natural number $a \leq b$, when $\{t_{l1}, \ldots, t_{ln_l}\}$ is the iid samples of Ω following P_l, ϵ-quasi label-wise proposed quality function $q_\epsilon(D_l)$ satisfies $\frac{2}{\pi}arccot(\frac{r_{la}}{c}) \leq q_\epsilon(D_l) \leq \frac{2}{\pi}arccot(\frac{r_{lb}}{c})$ with confidence level of at least*

$$\sum_{i=a}^{b-1} \binom{n_l}{i} \epsilon^{n_l - i}(1 - \epsilon)^i. \quad (7)$$

When n_l is sufficiently large, we can simplify the result as follows. By Central Limit Theorem [de Laplace, 1810], (7) approaches the $F(\frac{b-(1-\epsilon)n_l}{\sqrt{n_l\epsilon(1-\epsilon)}}) - F(\frac{a-(1-\epsilon)n_l}{\sqrt{n_l\epsilon(1-\epsilon)}})$ as n_l goes to infinity when F is the cumulative distribution function of the standard normal distribution. This shows $(1 - \epsilon)n_l$ becomes the mean (thus, peak) of the normal distribution. Assuming that ϵn_l is an integer for simplicity, we can estimate $q_\epsilon(D_l)$ as

$$\hat{q}_\epsilon(D_l) = \frac{2}{\pi}\text{arccot}\left(\frac{r_{l(1-\epsilon)n_l}}{c}\right). \quad (8)$$

Note that we use a finite test dataset T as the only trusted set of data from the input distribution P to evaluate the proposed quality function. However, the evaluation is not constrained by the size of the test dataset, as long as it consists of independent and identically distributed (iid) samples from the input probability distribution. This flexibility arises from the properties of random sampling.

Algorithm 1 describes the steps for evaluating the quality estimate $\hat{q}_\epsilon$ of a given dataset D. We first compute the set R of ordered elements (lines 3-6), and compute $\hat{q}_\epsilon(D_l)$ for each label l according to (8) (line 7). Algorithm 1 computes the label-wise average of $\hat{q}_\epsilon(D_l)$ and returns $\hat{q}_\epsilon(D)$ (lines 8-9).

Algorithm 1: Calculation of quality estimate

```
Input: Label set $L$, Metric $M$, Tolerance $\epsilon$,
       Dataset $D = \cup_{l \in L} D_l \times \{l\}$, Test dataset
       $T = \cup_{l \in L} T_l \times \{l\}$
Output: Quality $\hat{q}_\epsilon(D)$
1 Initialize all elements in the mindist array as a
  sufficiently large number.
2 for $l = 1 \to |L|$ do
3     for $i = 1 \to |T_l|$, $j = 1 \to |D_l|$ do
4         if $M(D_{lj}, T_{li}) < mindist[l][i]$ then
5             $mindist[l][i] \leftarrow M(D_{lj}, T_{li})$
6     Sort $mindist[l][1 \sim |T_l|]$ in ascending order.
7     $\hat{q}_\epsilon(D_l) \leftarrow \frac{2}{\pi}\text{arccot}(\frac{mindist[l][(1-\epsilon)|T_l|]}{c})$
8 $\hat{q}_\epsilon(D) \leftarrow \frac{\sum_l \hat{q}_\epsilon(D_l)}{|L|}$
9 return $\hat{q}_\epsilon(D)$
```

We further improve the computational efficiency of Algorithm 1 by improving the computation of the metric M using GPUs. In particular, we focus on cases where the metric M is an Euclidean norm in the projected space, such as the output space of a neural network. Algorithm 2 is the GPU-accelerated version of Algorithm 1 when the metric M is the Euclidean norm in the projected space.

Algorithm 2 begins by precomputing the projection of the test dataset using the projection PJ (line 1). We compute the pairwise distance between the projection of the test dataset T and the projection of the dataset D in each batch (line 4). These distances are used to update the *mindist* array for each label l (lines 5-6). Once the minimum distances are computed for all batches, they are sorted in ascending order (line 8) and used to compute $\hat{q}_\epsilon(D_l)$ (line 9). Finally, we return $\hat{q}_\epsilon(D)$, the label-wise average of $\hat{q}_\epsilon(D_l)$.

As the last result of this section, we discuss the computational complexity of Algorithms 1 and 2.

Proposition 2 *The computational complexity of Algorithm 1 is not larger than*

$$O(|D||T|)C(M) + O(|T|ln(|T|)) \tag{9}$$

and that of Algorithm 2 is not larger than

$$O(|D|+|T|)C(PJ)+O(|D||T|\max(p,|L|))+O(|T|ln(|T|)), \tag{10}$$

where $C(M)$ and $C(PJ)$ are the computational complexity of metric M and projection PJ, respectively.

Algorithm 2: GPU-accelerated version of Alg. 1

```
Input: Label set $L$, Tolerance $\epsilon$, Test dataset $T$,
       Dataset $D$ consisting of $k$ batches of size
       $|B|$, $B_1, \ldots, B_k$, Projection $PJ : \Omega \to \mathbb{R}^p$
Output: Quality $\hat{q}_\epsilon(D)$
1  Initialize $TestOut$ of size $|T| \times p \times |B|$ as $|B|$
   duplicates of $PJ(\cup_{l \in L} T_l)$.
2  for $m = 1 \to k$ do
3      $BatchOut \leftarrow |T|$ duplicates of $PJ(B_m)$
4      $Dist \leftarrow$norm of $(BatchOut - TestOut)$
       element-wise
5      for $l = 1 \to |L|$ do
6          Update $mindist[l][1 \sim |T_l|]$ with
           minimum of $Dist$ among elements with
           label $l$ in $B_m$
7  for $l = 1 \to |L|$ do
8      Sort $mindist[l][1 \sim |T_l|]$ in ascending order.
9      $\hat{q}_\epsilon(D_l) = \frac{2}{\pi}\text{arccot}(\frac{mindist[l][(1-\epsilon)|T_l|]}{c})$
10 $\hat{q}_\epsilon(D) = \frac{\sum_l \hat{q}_\epsilon(D_l)}{|L|}$
11 return $\hat{q}_\epsilon(D)$
```

In (9) and (10), the dominating terms are the metric computational complexity $C(M)$ and the projection computational complexity $C(PJ)$. In (9), $O(|D||T|)$ is multiplied to $C(M)$, which is much larger than $O(|D|+|T|)$ that is multiplied to $C(PJ)$ in (10). The main modification in Algorithm 2 is that we precompute the projection of the test dataset (line 1) and thus reduce computation time by reusing the saved projection outputs. Note that in practice, Algorithm 2 is much faster than the theoretical bound given in (10) because it can be accelerated by using GPUs.

Our method computes the quality function based on the inference results (the projection) and on the calculation of the norms and the minimum. This makes it much faster than the traditional method, which relies on training a network using the dataset D.

5 Experiments

Our method can benefit both academia and industry for various applications. To highlight its effectiveness and practicality, we present three applications: evaluating the effect of data management, real-time data monitoring, and large dataset quality evaluation.

5.1 Experiment details

We compare our proposed method with the traditional method in terms of the following metrics:

- *Test accuracy* is the quality of a dataset D eval-

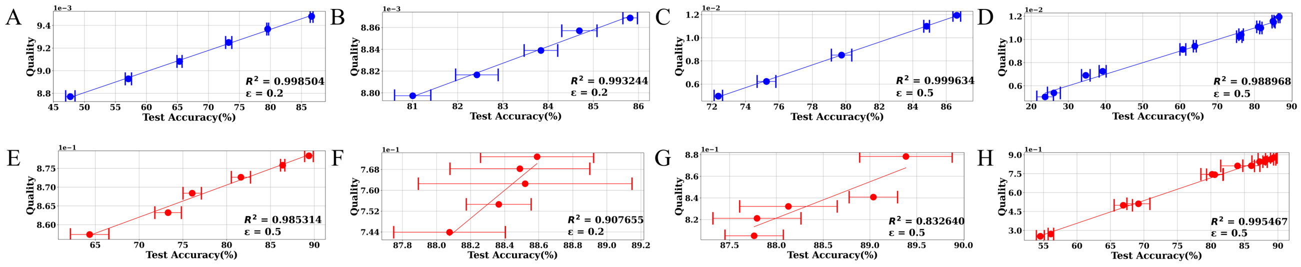

Figure 2: Relationship between the test accuracy and quality for CIFAR-10 (A, B, C, D) and IMDB (E, F, G, H).

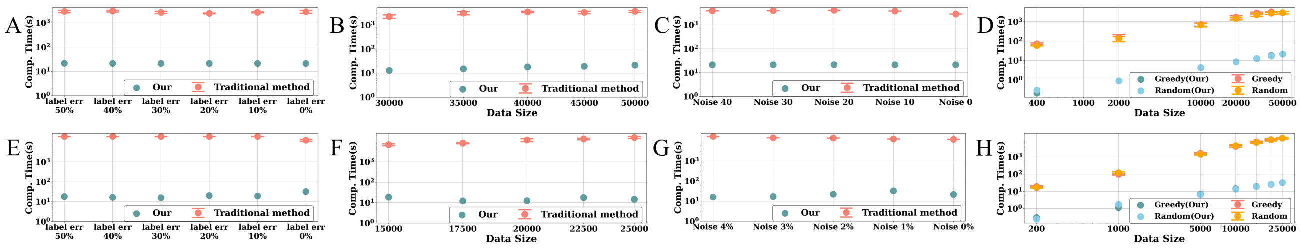

Figure 3: Computation time of the traditional and our methods on CIFAR-10 (A, B, C, D) and IMDB (E, F, G, H).

uated by the traditional method. The traditional method trains a neural network with the dataset D and evaluates, as the quality of D, the test accuracy with a separate test dataset.

- *Quality* is the quality estimate $\hat{q}_\epsilon(D)$ evaluated by our proposed method.

- *Computation time* is the time required to yield the test accuracy and quality, respectively by the traditional method and our method. We exclude initialization time, including dataset loading time, package loading time, CUDA initialization time, and test dataset training and inference time.

We validate our method with two datasets: image classification of the CIFAR-10 dataset [Krizhevsky, 2009] and sentiment classification of the IMDB dataset [Maas et al., 2011]. For both datasets, we use the (default) training dataset as the dataset D for quality evaluation and the (default) test dataset as the test dataset T in the evaluation process. For image classification, we use EfficientNetV2 [Tan and Le, 2021] and, in particular, use the code given in [Mayurji, 2021] for training. For sentiment classification, we use bidirectional LSTM [Hochreiter and Schmidhuber, 1997].

In all experiments, we use a virtual cloud computing service consisting of 8 A100 GPUs with 40GB SXM4 and 104 CPU cores provided by Lambda [Lambda,]. For the traditional method, we run training 4 times with GPU-parallelization, and report the mean and standard deviation. For the proposed method, we train a network with a test dataset and use the trained network as a projection PJ. Since the test dataset does not change for different tasks, we train the metric only once and use the saved checkpoint. Thus, the result is deterministic and the same at all times. We do not use GPU-parallelization since the proposed method is based on inference, and GPU-parallelization does not increase inference efficiency. Other experimental details are presented in the supplementary material.

5.2 Application 1: Evaluating the Effect of Data Management

We compare our method with the traditional method for data cleaning of label errors, incomplete datasets, or noise injected datasets and for core set selection.

5.2.1 Data Cleaning

Label error: We construct six datasets by replacing labels of 0%, 10%, 20%, 30%, 40%, 50% of images of each class with a random label. The random label is not necessarily an incorrect label, and the correct label can also be re-selected from the random label selection. The evaluation result and computation time of the traditional and proposed methods are presented in Figure 2-A and Figure 3-A for CIFAR-10 and in Figure 2-E and Figure 3-E for IMDB. The test accuracy varies from 47.72% to 86.63% (CIFAR-10) and from 67.41%

to 89.38% (IMDB), depending on the proportion of incorrect labels. The quality measured by our method shows a linear relation with the test accuracy with $R^2 \geq 0.985$. To obtain the test accuracy and quality, for the label error 30% dataset, the traditional method takes 2700.1 s for CIFAR-10 and 19175.5 s for IMDB, and our method takes 21.4 s for CIFAR-10 and 15.9 s for IMDB. This shows that our method achieves up to 1200 times faster computation time.

Incomplete dataset: We construct four datasets as follows: We replace 70% of image pixel rows with random numbers between 0 and 255 for 40% images for each class for CIFAR-10, and replace 10% of characters with random ASCII characters for 40% reviews for each class for IMDB. Then, we construct five datasets by removing $100\%, 75\%, 50\%, 25\%, 0\%$ of damaged images in each class. The evaluation result and computation time of the traditional and proposed method are presented in Figure 2-B and Figure 3-B for CIFAR-10, and in Figure 2-F and Figure 3-F for IMDB. For CIFAR-10, the quality measured by our method has a linear relation with the test accuracy with 171-207 times shorter computation time. For IMDB, however, the comparison is slightly distorted due to the large variance in the test accuracy.

Noise Injection: We construct five datasets by injecting a Gaussian random noise with a standard deviation of 0, 10, 20, 30, and 40 into the CIFAR-10 dataset and replacing 0%, 1%, 2%, 3%, and 4% of characters with a random ASCII character in IMDB dataset. The evaluation result and computation time of the traditional and proposed method are presented in Figure 2-C and Figure 3-C for CIFAR-10, and in Figure 2-G and Figure 3-G for IMDB. For CIFAR-10, the quality measured by our method has a linear relation with the test accuracy with 134-192 times shorter computation time. For IMDB, the comparison is slightly distorted due to the large variance in the test accuracy.

5.2.2 Core Set Selection

We use two core set selection approaches, label-wise random selection and k-center greedy selection. For the label-wise random selection, we randomly shuffle the entire dataset and take the first $0.8\%, 4\%, 20\%, 40\%, 60\%, 80\%$ samples from each label. For the k-center greedy selection, we take the first $0.8\%, 4\%, 20\%, 40\%, 60\%, 80\%$ of the sequence obtained by the k-center greedy method [Google, 2017]. The evaluation result and computation time of the traditional and proposed method are presented in Figure 2-D and Figure 3-H for CIFAR-10 and in Figure 2-D and Figure 3-H for IMDB. In Figure 3-D and 3-H, for the 0.8% dataset with k-center greedy selection, the traditional method takes 72.7 s for CIFAR-10 and 18.9 s for IMDB, whereas our method takes only 0.2s for CIFAR-10 and 0.3 s for IMDB. In total, our method requires at least 60 times less computation time compared to the traditional method. Also, the quality measured by our method exhibits a linear relation with the test accuracy.

In this application, we have demonstrated that our proposed quality evaluation method significantly reduces computation time, allowing researchers to check the quality of data in just a few seconds. This enables more frequent monitoring and faster iteration during data management.

5.2.3 Comparison with Baselines

We compare our work with 3 baselines [Garima et al., 2020, Koh and Liang, 2017, Park et al., 2023] for all experiments we conducted in this research except TRAK for IMDB case since the packing procedure included in the network is not compatible with the TRAK package. For each related work, we calculated the dataset score by summing the individual data point scores, using the entire test dataset as the target. For both CIFAR-10 and IMDB, we utilized training checkpoints from the full training datasets. In particular, we selected 10 checkpoints at equal intervals for TracIn and TRAK. We neglect details of Adam optimizer in all baseline experiments. The reported values of the test accuracy in the table are the average of four trials.

As shown in the table, the baselines from the designated related works show significantly worse values with test accuracy and significantly longer computation times compared to our proposed quality measure, across all cases and datasets except IMDB noise case (this case is significantly affected by variance of test accuracy) and datasets.

5.3 Application 2: Real-time Data Management

In practice, datasets are often continuously growing; for example, companies collect customer data while providing services. It is necessary to efficiently manage such datasets and maintain their values. In this section, we present the use of our proposed evaluation method for this purpose.

We use the datasets constructed in Section 5.2 through the label-wise random and k-center greedy core set selections. We consider two approaches of the traditional and proposed methods. The first approach is to evaluate the quality from scratch using the given datasets, and the second approach is to update the evaluation result for added data only. For the updates, the traditional method fine-tunes the trained neural network

Table 1: Comparison of R^2 (with test accuracy) and computation time between the quality and assigned baselines

Example	Result	Method	Label error	Incomplete data	Coreset selection	Input noise
CIFAR-10	R^2	Ours	**0.9985**	**0.9932**	**0.9890**	**0.9996**
		TracIn	0.01241	0.9907	0.8088	0.9355
		Influence	0.4786	0.07007	0.5999	0.8375
		TRAK	0.9677	0.9555	0.9652	0.7818
	Computation time(s)	Ours	128.6	86.72	110.5	94.63
		TracIn	982.8	662.6	874.3	814.9
		Influence	8631	5750	7743	7382
		TRAK	73676	51919	79749	63394
IMDB	R^2	Ours	**0.9853**	**0.9077**	**0.9955**	0.8326
		TracIn	0.9722	0.8174	0.7265	**0.8986**
		Influence	0.8803	0.6457	0	0.3348
	Computation time(s)	Ours	122.5	75.67	166.4	105.8
		TracIn	15582	9382	13350	11581
		Influence	13280	8433	13440	10153

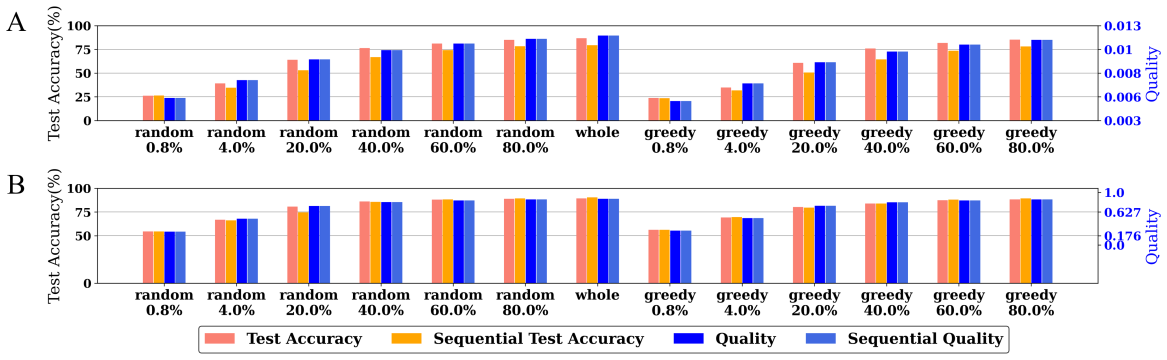

Figure 4: Evaluation of the test accuracy and quality on growing datasets of CIFAR-10 (A) and IMDB (B).

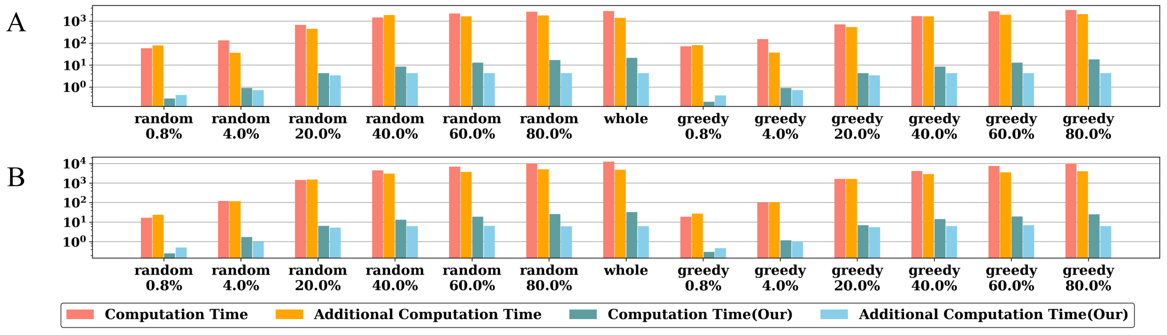

Figure 5: Computation time of the sequential evaluation on growing datasets for CIFAR-10 (A) and IMDB (B).

with the existing and added data, and our method computes distances for added data and updates the minimum distance *mindist* in Algorithm 2. These results are denoted in Figure 4 as "Sequential Test Accuracy" and "Sequential Quality," and in Figure 5 as "Additional Computation Time."

Figures 4 and 5 show the comparison between the traditional method and our method for growing datasets (subplots A for CIFAR-10 and subplots B for IMDB). Note in Figure 4 that the test accuracy and sequential test accuracy are different, whereas the quality and sequential quality measured by our method are the same. This implies that the traditional method may not be suited for sequentially evaluating the quality of a growing dataset because the test accuracy can become distorted. In Figure 5-B, for the random 0.8%, 4%, 20%, 40%, 60%, 80%, and whole datasets of IMDB, the computation times of the traditional method are 16.6 s, 121.3 s, 1448.0 s, 4520.4 s, 6925.9 s, 10000.9 s, 12400.4 s, and the additional computation times are 24.0 s, 119.0 s, 1528.1 s, 3047.5 s, 3742.6 s, 5094.1 s, and 4783.2 s, respectively. In contrast, the computation times of our method are 0.2 s, 1.7 s, 6.4 s, 13.3 s, 18.8 s, 25.9 s, and 32.5 s, and the additional computation times are 0.5 s, 1.0 s, 5.2 s, 6.2 s, 6.4 s, 6.1 s, and 6.2 s, respectively. Note that the additional computation time of our method is proportional to the size of the newly added data (added data are 0.8%, 3.2%, 16%, 20%, 20%, 20%, 20%, respectively). Also, the additional computation time of our method is 47-830 times shorter than that of the traditional method. This demonstrates that our method is highly effective for real-time quality evaluation of continuously growing datasets.

5.4 Application 3: Large Dataset Evaluation

As the final application of our proposed method, we demonstrate how our method can be used to estimate the test accuracy on a large (training) dataset. Instead of training a neural network on a large dataset, which can be extremely time-consuming, we can train the network on smaller datasets and estimate the test accuracy on the full dataset. The key idea is to utilize the linear relation between the test accuracy and the quality measured by our method.

In the experiments, we use the test accuracy and quality measured by our method for three datasets comprising 0.8%, 4%, and 20% of the entire dataset, which we construct in Section 5.2.2 through random core set selection. We perform linear regression, similar to Figure 2, between the test accuracy and quality for the three smaller datasets, obtaining $y = 9434x - 26.27$ with $R^2 = 0.9827$ for CIFAR-10 and $y = 53.25x - 40.79$ with $R^2 = 0.9990$ for IDMB. Next, we use our method to measure the quality of the entire dataset, which is possible due to its computational efficiency. Using the linear equation and the quality of the entire dataset, we can obtain the estimate of the test accuracy on the entire dataset. As shown in Figure 6, the relative error between the estimate and actual test accuracy is 0.294% for CIFAR-10 and 2.038% for IMDB. This demonstrates the effectiveness of our method in estimating the test accuracy on a large dataset.

6 Conclusion

We have defined a coverage-ensuring quality function, as the function that quantifies how well the dataset covers the input probability distribution of AI models. We proposed a candidate function, which we have proven to satisfy the coverage-ensuring conditions, and presented a computationally efficient, GPU-accelerated algorithm that approximately implements the function. To demonstrate the effectiveness and practicality of our method, we presented three applications: evaluating the impact of data management, enabling real-time data quality monitoring, and assessing large datasets. [1] The promising results presented in this paper strongly suggest that our proposed quality function will be a powerful and computationally efficient tool for a wide range of applications in data-centric AI, including data collection, processing, and quality assurance. We believe that our proposed method holds significant potential for evaluating the quality of newer and larger datasets, a task that has been challenging or even infeasible with existing approaches. We look forward to its application to the larger datasets commonly used in modern machine-learning problems.

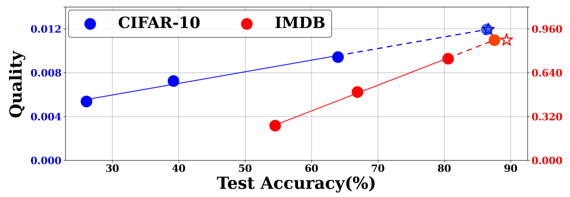

Figure 6: Estimated test accuracy (circles at the end of the dotted lines) and the actual test accuracy (star-shaped).

Our proposed method can be extended to consider some other critical aspects of dataset quality. Although the proposed dataset quality function utilizes only the coverage of the same label, we can compare coverage within the same label to coverage of different labels to obtain some information regarding accuracy. Similarly, by comparing coverage to the subset of the target dataset associated with each category, we can also consider bias. Moreover, our method can be used to provide the generalization bound (see supplementary material).

For complex, noisy, or imbalanced datasets, computing precise test accuracy poses a challenge in comparison, as these types of datasets typically introduce large variance in test accuracy. Developing a stable metric to effectively demonstrate the usefulness of our quality measure on these challenging datasets is left for future work.

[1]Since the experiments use the test dataset in quality evaluation, we also run experiments that use separate dataset for quality evaluation and report the result in supplementary material to show that our method works well without label leakage.

Acknowledgements

A major part of this research is conduct as the paid work of Beomjun Kim in AIMMO CO., Ltd. The work of Kangyeon Kim and Heejin Ahn was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP)-ITRC grant funded by the Korea government (IITP-2025-RS-2023-00259991, 100 %). We use ChatGPT for proofreading.

References

[Amagata, 2024] Amagata, D. (2024). Fair k-center clustering with outliers. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 10–18.

[Batini et al., 2009] Batini, C., Cappiello, C., Francalanci, C., and Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM Comput. Surv.*, 41(3):1–52.

[Bogatu et al., 2020] Bogatu, A., Fernandes, A. A. A., Paton, N. W., and Konstantinou, N. (2020). Dataset discovery in data lakes. In *IEEE 36th International Conference on Data Engineering (ICDE)*, pages 709–720.

[Borkin et al., 2013] Borkin, M. A., Vo, A. A., Bylinskii, Z., Isola, P., Sunkavalli, S., Oliva, A., and Pfister, H. (2013). What makes a visualization memorable? *IEEE transactions on visualization and computer graphics*, 19(12):2306–2315.

[Budach et al., 2022] Budach, L., Feuerpfeil, M., Ihde, N., Nathansen, A., Noack, N., Patzlaff, H., Naumann, F., and Harmouch, H. (2022). The effects of data quality on machine learning performance. *arXiv preprint arXiv:2207.14529*.

[Burch and Weiskopf, 2013] Burch, M. and Weiskopf, D. (2013). On the benefits and drawbacks of radial diagrams. In *Handbook of human centric visualization*, pages 429–451. Springer.

[Chen et al., 2020] Chen, J., Yang, Z., and Yang, D. (2020). Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*.

[Chu et al., 2016] Chu, X., Ilyas, I. F., Krishnan, S., and Wang, J. (2016). Data cleaning: Overview and emerging challenges. In *International Conference on Management of Data (MOD)*, SIGMOD '16, page 2201–2206, New York, NY, USA. Association for Computing Machinery.

[de Laplace, 1810] de Laplace, P. (1810). *Mémoire sur les approximations des formules qui sont fonctions de très-grands nombres, et sur leur application aux probabilités*. Baudouin.

[Dekel and Shamir, 2009] Dekel, O. and Shamir, O. (2009). Vox populi: Collecting high-quality labels from a crowd. In *Conference on Learning Theory (COLT)*.

[Donoho et al., 1998] Donoho, D. L., Vetterli, M., DeVore, R. A., and Daubechies, I. (1998). Data compression and harmonic analysis. *IEEE transactions on information theory*, 44(6):2435–2476.

[Drori et al., 2021] Drori, I., Krishnamurthy, Y., Rampin, R., Lourenco, R. d. P., Ono, J. P., Cho, K., Silva, C., and Freire, J. (2021). Alphad3m: Machine learning pipeline synthesis. *arXiv preprint arXiv:2111.02508*.

[Frid-Adar et al., 2018] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *IEEE 15th international symposium on biomedical imaging (ISBI)*, pages 289–293. IEEE.

[Garima et al., 2020] Garima, Liu, F., Kale, S., and Sundararajan, M. (2020). Estimating training data influence by tracing gradient descent. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

[Gong et al., 2023] Gong, Y., Liu, G., Xue, Y., Li, R., and Meng, L. (2023). A survey on dataset quality in machine learning. *Information and Software Technology*, 162:1199–1208.

[Google, 2017] Google (2017). active-learning. *https://github.com/google/active-learning/blob/master/sampling_methods/kcenter_greedy.py*.

[Graves and Schmidhuber, 2005] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610. IJCNN 2005.

[Guo et al., 2022] Guo, C., Zhao, B., and Bai, Y. (2022). Deepcore: A comprehensive library for coreset selection in deep learning. In Strauss, C., Cuzzocrea, A., Kotsis, G., Tjoa, A. M., and Khalil, I., editors, *Database and Expert Systems Applications*, pages 181–195, Cham. Springer International Publishing.

[Heffetz et al., 2020] Heffetz, Y., Vainshtein, R., Katz, G., and Rokach, L. (2020). Deepline: Automl tool for pipelines generation using deep reinforcement learning and hierarchical actions filtering. In *26th ACM SIGKDD international conference on knowledge discovery data mining (KDD)*, pages 2103–2113.

[Herodotou et al., 2011] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S. (2011). Starfish: A self-tuning system for big data analytics. In *Conference on Innovative Data Systems Research (CIDR)*, volume 11, pages 261–272.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, abs/1412.6980.

[Koh and Liang, 2017] Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

[Krishnan et al., 2016] Krishnan, S., Wang, J., Wu, E., Franklin, M. J., and Goldberg, K. (2016). Activeclean: Interactive data cleaning while learning convex loss models. *ArXiv preprint arXiv:1601.03797*, abs/1601.03797.

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

[Lai et al., 2021] Lai, K.-H., Zha, D., Wang, G., Xu, J., Zhao, Y., Kumar, D., Chen, Y., Zumkhawaka, P., Wan, M., Martinez, D., et al. (2021). Tods: An automated time series outlier detection system. In *AAAI conference on artificial intelligence (AAAI)*, volume 35, pages 16060–16062.

[Lambda,] Lambda. *https://lambdalabs.com/*.

[Li and Yu, 2024] Li, W. and Yu, Y. (2024). *Robust data valuation with weighted banzhaf values*. Curran Associates Inc., Red Hook, NY, USA.

[Lorentz, 1964] Lorentz, G. G. (1964). Entropy and its applications. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 1:97–103.

[Lorentz, 1966] Lorentz, G. G. (1966). Metric entropy and approximation.

[Maas et al., 2011] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

[Mayurji, 2021] Mayurji (2021). Image-classification-pytorch: Learning and building convolutional neural networks using pytorch. *https://github.com/Mayurji/Image-Classification-PyTorch*.

[Moreno-Barea et al., 2020] Moreno-Barea, F. J., Jerez, J. M., and Franco, L. (2020). Improving classification accuracy using data augmentation on small data sets. *Expert Systems with Applications*, 161:957–4174.

[Park et al., 2023] Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. (2023). Trak: attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

[Pipino et al., 2002] Pipino, L. L., Lee, Y. W., and Wang, R. Y. (2002). Data quality assessment. *Communications of the ACM*, 45(4):211–218.

[Sener and Savarese, 2018] Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations (CoLR)*.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

[Shin et al., 2023] Shin, S.-J., Bae, H., Shin, D., Joo, W., and Moon, I.-C. (2023). Loss-curvature matching for dataset selection and condensation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[Tan and Le, 2021] Tan, M. and Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In Meila, M. and Zhang, T., editors, *38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 10096–10106. PMLR.

[torchtext,] torchtext. *https://pytorch.org/text/stable/index.html*.

[Wang et al., 2024] Wang, J. T., Mittal, P., and Jia, R. (2024). Efficient data shapley for weighted nearest neighbor algorithms. *arXiv preprint arXiv:2401.11103*.

[Zha et al., 2023a] Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., and Hu, X. (2023a). *Data-centric AI: Perspectives and Challenges*, pages 945–948.

[Zha et al., 2023b] Zha, D., Bhat, Z. P., Lai, K.-H., Yang, F., Jiang, Z., Zhong, S., and Hu, X. (2023b). Data-centric artificial intelligence: A survey. *arXiv preprint arXiv:2303.10158*.

[Zhang, 2016] Zhang, Z. (2016). Missing data imputation: focusing on single imputation. *Annals of translational medicine*, 4(1).

[Zoph et al., 2020] Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. (2020). Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

7 PROOFS

7.1 Proof of Theorem 1

First, we prove Condition 1 for both $q(D)$ and $q_\epsilon(D)$. For simplicity, we prove for q_ϵ with $0 \leq \epsilon < 1$ since we can consider $q(D)$ as $\epsilon = 0$ case. For any $\delta > 0$, $\mathcal{B} := \left\{ B\left(\omega, \frac{c\cot(\frac{\pi}{2}(1-\delta))}{2}\right) | \omega \in \Omega \right\}$ is an open cover of Ω, and since we assumed Ω is compact, there exists finite subcover $\left\{ B\left(\omega_1, \frac{c\cot(\frac{\pi}{2}(1-\delta))}{2}\right), \ldots, B\left(\omega_{n_c}, \frac{c\cot(\frac{\pi}{2}(1-\delta))}{2}\right) \right\} \subset \mathcal{B}$ for some $\omega_1, \cdots, \omega_{n_c} \in \Omega$. Let $C_i := B\left(\omega_i, \frac{c\cot(\frac{\pi}{2}(1-\delta))}{2}\right) \setminus \cup_{j=1}^{i-1} B\left(\omega_j, \frac{c\cot(\frac{\pi}{2}(1-\delta))}{2}\right)$; then, $\{C_i\}$ is a partition of Ω whose elements have a diameter of at most $c\cot(\frac{\pi}{2}(1-\delta))$. Since we assume that each P_l is well-defined on the σ-algebra, which includes all open sets, each P_l is well-defined for C_i. Due to the strong law of large numbers, $Pr\left(\lim_{|D|\to\infty} \frac{|D_l \cap C_i|}{|D|} = P(C_i \times \{l\})\right) = 1$. Moreover, $Pr\left(\lim_{|D|\to\infty} \frac{|D_l|}{|D|} = P(\Omega \times \{l\})\right) = 1$. Combining them and considering that $P_l(C_i) = \frac{P(C_i \times \{l\})}{P(\Omega \times \{l\})}$, we obtain

$$Pr\left(\lim_{|D|\to\infty} \frac{|D_l \cap C_i|}{|D_l|} = P_l(C_i)\right) = 1 \tag{11}$$

provided that $D_l \neq \phi$. This means

$$P_l(C_i) > 0 \implies Pr\left(\exists N_i\, ,\, |D| > N_i \implies D_l \cap C_i \neq \phi\right) = 1 \tag{12}$$

for all $1 \leq i \leq n_c$ provided that $D_l \neq \phi$.

Meanwhile,

$$\begin{aligned} P_l\left(\bigcup_{x \in D_l} B(x, c\cot(\frac{\pi}{2}(1-\delta)))\right) &\geq P_l\left(\bigcup_{x\in D_l} \bigsqcup_{i: x \in C_i} C_i\right) && \text{(monotonicity of probability)} \\ &= P_l\left(\bigsqcup_{i: D_l \cap C_i \neq \phi} C_i\right) && \text{(set algebra)} \\ &= \sum_{i: D_l \bigcap C_i \neq \phi} P_l(C_i) && \text{(additivity of probability).} \end{aligned}$$

Combining this and equation (12), since $\sum_i P_l(C_i) = 1$ as $\{C_i\}$ is a partition of Ω, we obtain (for any $0 \leq \epsilon \leq 1$)

$$Pr\left(\exists N,\, |D| > N \implies P_l(\cup_{x \in D_l} B(x, c\cot(\frac{\pi}{2}(1-\delta)))) = 1\right) = 1. \tag{13}$$

This implies a possibly weaker argument as

$$Pr\left(\exists N,\, |D| > N \implies P_l(\cup_{x \in D_l} B(x, c\cot(\frac{\pi}{2}(1-\delta)))) \geq 1 - \epsilon\right) = 1 \tag{14}$$

for any $0 \leq \epsilon < 1$.

Equivalently,

$$Pr\left(\exists N,\, |D| > N \implies D_l : \epsilon\text{−quasi−}c\cot(\frac{\pi}{2}(1-\delta))\text{−coverage}\right) = 1. \tag{15}$$

Moreover, by the definition of $q_\epsilon(D_l)$, this implies that

$$Pr\left(\exists N,\, |D| > N \implies q_\epsilon(D_l) \geq 1 - \delta\right) = 1. \tag{16}$$

Therefore, considering that δ is an arbitrary positive real number, we obtain

$$Pr\left(\lim_{|D|\to\infty} q_\epsilon(D_l) = 1\right) = 1, \tag{17}$$

and thus,

$$Pr\left(\lim_{|D|\to\infty} q_\epsilon(D) = 1\right) = 1. \tag{18}$$

Next, we prove Condition 2 for $q(D)$. Let $\bar{g} := \frac{|L|-0.5+\frac{1}{\pi}\text{arccot}(\frac{r}{c})}{|L|}$. Then, when $q(D) > \bar{g}$, since $q(D_l) \geq 0.5 + \frac{1}{\pi}\text{arccot}(\frac{r}{c}) > \frac{2}{\pi}\text{arccot}(\frac{r}{c})$ for all l, $\inf(\{\zeta|D_l : \zeta\text{-coverage}\}) < r$ for all l. Thus,

$$\begin{aligned}
&P_l(\cup_{x\in D_l} B(x,r)) = 1 \\
&= P_l(\{\omega|\exists\omega_1 \in D_l, M(\omega,\omega_1) \leq r\}) \\
&= Pr_{\omega\sim P_l}(\exists\omega_1 \in D_l, M(\omega,\omega_1) \leq r) \\
&= Pr_{\omega\sim P_l}(\exists(\omega_1,l) \in D, M(\omega,\omega_1) \leq r).
\end{aligned}$$

Therefore,

$$\begin{aligned}
&Pr_{(\omega,l)\sim P}(\exists(\omega_1,l) \in D, M(\omega,\omega_1) \leq r) \\
&= \sum_l Pr_{(\omega,l_1)\sim P}(l_1 = l) Pr_{\omega\sim P_l}(\exists(\omega_1,l) \in D, M(\omega,\omega_1) \leq r) \\
&= \sum_l Pr_{(\omega,l_1)\sim P}(l_1 = l) = 1
\end{aligned}$$

Finally, for $q_\epsilon(D)$, with same $\bar{g}$, since $q_\epsilon(D_l) \geq 0.5 + \frac{1}{\pi}\text{arccot}(\frac{\text{r}}{\text{c}}) > \frac{2}{\pi}\text{arccot}(\frac{\text{r}}{\text{c}})$ for all l, $\inf(\{\zeta|D_l :$ ϵ-quasi-ζ-coverage$) < r$ for all l. Thus,

$$Pr_{\omega\sim P_l}(\exists(\omega_1,l) \in D, M(\omega,\omega_1) \leq r) = P_l(\cup_{x\in D_l} B(x,r)) \geq 1-\epsilon$$

Therefore,

$$\begin{aligned}
&Pr_{(\omega,l)\sim P}(\exists(\omega_1,l) \in D, M(\omega,\omega_1) \leq r) \\
&= \sum_l Pr_{(\omega,l_1)\sim P}(l_1 = l) Pr_{\omega\sim P_l}(\exists(\omega_1,l) \in D, M(\omega,\omega_1) \leq r) \\
&\geq \sum_l Pr_{(\omega,l_1)\sim P}(l_1 = l)(1-\epsilon) = 1-\epsilon.
\end{aligned}$$

7.2 Proof of Proposition 1

Consider the partition of Ω:

$$\begin{aligned}
&X := \cup_{x\in D_l} B(x, c\cot(\frac{\pi}{2}q_\epsilon(D_l))) \\
&Y := \bar{X} \setminus X \\
&Z = \Omega \setminus \bar{X}
\end{aligned}$$

when $\bar{X}$ is the closure of X.

By Definition 3 and 4, we can obtain

$$\begin{aligned}
&P_l(X) \leq 1-\epsilon, \\
&P_l(X\cup Y) = P_l(\bigcap_{r>c\cot(\frac{\pi}{2}q_\epsilon(D_l))} \cup_{x\in D_l} B(x,r)) = \min_{r>c\cot(\frac{\pi}{2}q_\epsilon(D_l))} P_l(\cup_{x\in D_l} B(x,r)) \geq 1-\epsilon
\end{aligned}$$

Note that since the quality function is defined to employ the infimum of the radius ζ that makes the dataset ζ-coverage in Definition 4, $P_l(X) \leq 1-\epsilon$. When the infimum equals to the minimum, $P_l(X) = 1-\epsilon$ holds.

Then, by the assumption that the test data T_l are the iid samples in Ω following P_l, the confidence level for

$$\frac{2}{\pi}\text{arccot}(\frac{r_{la}}{c}) \leq q_\epsilon(D_l) \leq \frac{2}{\pi}\text{arccot}(\frac{r_{lb}}{c})) \tag{19}$$

is the probability of event that $(|(X\cup Y)\cap T_l| \geq a \wedge |(Y\cup Z)\cap T_l| \geq n_l - b)$, which is the complementary event of $(|Z\cap T_l| \geq n_l - a \vee |X\cap T_l| \geq b)$.

Since $P_l(X) \leq 1-\epsilon$ and $P_l(Z) = 1 - P_l(X \cup Y) \leq \epsilon$, the confidence level is

$$1 - \sum_{i=b}^{n_l} \binom{n_l}{i} P_l(X)^i (1-P_l(X))^{n_l-i} - \sum_{i=n_l-a}^{n_l} \binom{n_l}{i} P_l(Z)^i (1-P_l(Z))^{n_l-i}$$

$$\geq 1 - \sum_{i=b}^{n_l} \binom{n_l}{i} (1-\epsilon)^i \epsilon^{n_l-i} - \sum_{i=n_l-a}^{n_l} \binom{n_l}{i} \epsilon^i (1-\epsilon)^{n_l-i} = \sum_{i=a}^{b} \binom{n_l}{i} \epsilon^{n_l-i} (1-\epsilon)^i$$

The inequality holds because $\sum_{i=b}^{n_l} \binom{n_l}{i} (1-\epsilon)^i \epsilon^{n_l-i}$ and $\sum_{i=n_l-a}^{n_l} \binom{n_l}{i} \epsilon^i (1-\epsilon^{n_l-i})$ are compliments of the cumulative function of binomial distributions and thus increasing as a function of $1-\epsilon$ and ϵ, respectively. The equality holds when $P_l(Y) = 0$, thus $P_l(X) = 1-\epsilon$ and $P_l(Z) = \epsilon$.

7.3 Proof of Proposition 2

For Algorithm 1, we run the if block (including the metric computation) $\sum_{l=1}^{|L|} |D_l||T_l|$ times and the sort of $|T_l|$ element for each l, which requires computational complexity as $\sum_{l=1}^{|L|} |D_l||T_l|C(M) = O(|D||T|)C(M)$, $\sum_{l=1}^{|L|} O(|T_l| ln(|T_l|)) = O(|T| ln(|T|))$, respectively. Therefore the total computational complexity is not larger than

$$O(|D||T|)C(M) + O(|T| ln(|T|)). \tag{20}$$

For Algorithm 2, we run the projection $|D|+|T|$ times. Expanding, reshaping, substituting and norm computation requires time complexity of at most the data size, thus $O(|D||T| max(p, |L|))$, when p is the dimension of the metric space. Finally, the sorting requires $O(|T| ln(|T|))$ as stated in the former paragraph. Therefore, the total computational complexity is not larger than

$$O(|D| + |T|)C(P) + O(|D||T| max(p, |L|)) + O(|T| ln(|T|)). \tag{21}$$

8 EXPERIMENT DETAILS

Network Architectures and Preprocessing: For CIFAR-10, we use EfficientNetV2-s [Tan and Le, 2021] as our classification network. We resize all the images to (224, 224) as the preprocessing. No other transformation is applied.

For IMDB, we use 2-layer Bidirectional LSTM [Graves and Schmidhuber, 2005] with embedding dimension= 100, hidden dimension= 256, and $dropout = 0.5$. We tokenize the comments and build vocabulary using torchtext [torchtext,] as the preprocessing. No other transformation is applied.

Training Details: In both cases, we train the model until convergence with maximum epoch 3000. For the batch size, we use 1000 for CIFAR-10 and 5000 for IMDB. When the dataset size is lower than theirs, we use the dataset size (thus, full batch) for the datasets. The learning rate for both cases is 0.001. We use the Adam optimizer [Kingma and Ba, 2014] for both cases.

Code for the Proposed Method: The code for the proposed method is attached as a separate file.

9 Validity regarding Label Leakage

To show that our method still has strong correlation with test accuracy when there is no risk of label leakage, we redesign the experiments so that separate datasets are used for quality evaluation and test accuracy evaluation.

Specifically, for the CIFAR-10 and IMDB datasets, we divide the test set into two, one for quality evaluation and the other for computing test accuracy. Then, we re-ran the proposed quality evaluation and re-evaluated test accuracy. We observed that the proposed quality measure remains strongly correlated with test accuracy, to a degree comparable to what is presented in our manuscript.

The test accuracy, quality, and the R^2 value of incomplete data case, labelerror case, and coreset selection case are presented in the following three tables.

Based on the results presented in the table, we can conclude that our proposed method is valid without label leakage.

Table 2: Test accuracy, proposed quality, and R^2 value for incomplete data case

Example	Method	R^2	0%	10%	20%	30%	40%
CIFAR-10	Test accuracy		74.43	82.56	84.16	84.81	85.71
	Proposed	**0.9973**	0.002009	0.002306	0.002388	0.002414	0.00245
IMDB	Test accuracy		88.08	88.37	88.52	88.49	88.59
	Proposed	**0.9680**	0.005219	0.2653	0.3350	0.3824	0.4079

Table 3: Test accuracy, proposed quality, and R^2 value for label error case

Example	Method	R^2	0%	10%	20%	30%	40%	50%
CIFAR-10	Test accuracy		79.48	79.35	73.86	66.12	57.60	47.87
	Proposed	**0.9698**	0.006671	0.0066	0.006519	0.006457	0.006361	0.006224
IMDB	Test accuracy		89.38	87.26	83.70	79.04	71.51	67.41
	Proposed	**0.9829**	0.8664	0.8643	0.8605	0.8562	0.8492	0.8417

Table 4: Test accuracy, proposed quality, and R^2 value for coreset selection case

Example	Method	R^2	Coreset Method	0.8%	4%	20%	40%	60%	80
CIFAR-10	Test accuracy		Labelwise Random	26.44	35.07	53.47	66.87	74.43	78.4
			k-center Greedy	23.53	31.92	50.75	64.83	73.55	78.0
	Proposed	**0.9611**	Labelwise Random	0.003752	0.004603	0.005557	0.006022	0.006264	0.0065
			k-center Greedy	0.002978	0.003949	0.005256	0.005912	0.006294	0.0065
IMDB	Test accuracy		Labelwise Random	54.53	66.91	80.57	86.03	88.08	88.9
			k-center Greedy	56.09	69.17	80.17	83.97	87.26	88.2
	Proposed	**0.9965**	Labelwise Random	0.2468	0.4748	0.7120	0.7923	0.8299	0.85
			k-center Greedy	0.2558	0.4846	0.7180	0.7959	0.8316	0.85

10 Generalization Bound

Our method can be used to provide generalization bound for non-i.i.d. sampled training data using the coverage notion.

Typically, classification outputs are derived from a simple function, such as a maximum or rounding operation, applied to a vector of continuous intermediate outputs. Let $\mathbf{w}(\omega)$ be the vector for the input ω and $f(\mathbf{w})$ be the classification output. The output is correct if $f(\mathbf{w}(\omega)) = l$ with the ground-truth label l. The accuracy $\mathbf{acc}^*(f(\mathbf{w}(\cdot)))$ can be defined as

$$\mathbf{acc}^*(f(\mathbf{w}(\cdot))) := \{E_{\omega:(\omega,l)\sim P}\mathbf{1}(f(\mathbf{w}(\omega)), l)\}_{l\in L}, \tag{22}$$

where $\mathbf{1}(a, b)$ is 1 if $a = b$ and 0 otherwise, L is the label set, P is the probability distribution of inputs.

Now, we can provide the lower bound of the actual accuracy based on the coverage radius of labelwise datasets D_l, when $\sum_l \sum_{\omega:(\omega,l)\in D} \min_{||\tilde{\mathbf{w}}-\mathbf{w}(\omega)||\leq\xi} \mathbf{1}(f(\tilde{\mathbf{w}}), l) = |D|$.

Theorem 2 *When ξ satisfies*

$$M(\omega_1, \omega_2) \leq \max_l \inf(\{\zeta|D_l : \epsilon\text{-}quasi\text{-}\zeta\text{-}coverage\}) \implies ||\boldsymbol{w}(\omega_1), \boldsymbol{w}(\omega_2)|| \leq \xi, \tag{23}$$

then, for any $f(\cdot)$ and $\boldsymbol{w}(\cdot)$,

$$\sum_l \sum_{\omega:(\omega,l)\in D} \min_{||\tilde{\boldsymbol{w}}-\boldsymbol{w}(\omega)||\leq\xi} \boldsymbol{1}(f(\tilde{\boldsymbol{w}}), l) = |D| \implies \boldsymbol{acc}^*(f(\boldsymbol{w}(\cdot))) \geq (1-\epsilon)\boldsymbol{1}. \tag{24}$$

Proof:

$$\forall l, \mathbf{acc}^*(f(\mathbf{w}(\cdot)))_l = E_{\omega:(\omega,l)\sim P}\mathbf{1}(f(\mathbf{w}(\omega)), l) = E_{\omega\sim P_l}\mathbf{1}(f(\mathbf{w}(\omega)), l) = P_l(\{\omega|\mathbf{1}(f(\mathbf{w}(\omega)) = l\}) \tag{25}$$

On the other hand, (23) implies $\cup_{\omega\in D_l} B(\mathbf{w}(\omega), \xi) \supset \mathbf{w}(\cup_{\omega\in D_l} B(\omega, \inf(\{\zeta|D_l : \epsilon\text{-quasi-}\zeta\text{-coverage}\})))$. Considering the exclusivity of the label, the condition of (24) implies $\sum_{\omega:(\omega,l)\in D} \min_{||\tilde{\mathbf{w}}-\mathbf{w}(\omega)||\leq\xi} \mathbf{1}(f(\tilde{\mathbf{w}}), l) = |D_l|$, and thus, $\forall\omega \in D_l, B(f(\tilde{\mathbf{w}}), \xi) \subset \{\mathbf{w}|f(\mathbf{w}) = l\}$. Combining these two statement, we obtain $f(\mathbf{w}(\cup_{\omega\in D_l} B(\omega, \inf(\{\zeta|D_l : \epsilon\text{-quasi-}\zeta\text{-coverage}\})))) = \{l\}$ and $\cup_{\omega\in D_l} B(\omega, \inf(\{\zeta|D_l : \epsilon\text{-quasi-}\zeta\text{-coverage}\})) \subset \{\omega|\mathbf{1}(f(\mathbf{w}(\omega)) = l\}$. Therefore, we obtain $\mathbf{acc}^*(f(\mathbf{w}(\cdot)))_l = P_l(\{\omega|\mathbf{1}(f(\mathbf{w}(\omega)) = l\}) \geq P_l(\cup_{\omega\in D_l} B(\omega, \inf(\{\zeta|D_l : \epsilon\text{-quasi-}\zeta\text{-coverage}\}))) \geq 1-\epsilon$.

This theorem tells that we can provide a guaranteed lower bound of the actual performance with a dataset whose quality is known. Given a network, we first compute a ξ such that $\mathbf{acc}^{\xi}(f(\mathbf{w}(\cdot)); D) = 1$. Then, we determine the ϵ that satisfies the condition (23) and therefore obtain the lower bound of the actual accuracy $1-\epsilon$.

This approach for providing a lower bound is useful when the training dataset cannot be guaranteed to consist of iid samples, which is often the case in practice. The scheme is practical since we can achieve $\sum_l \sum_{\omega:(\omega,l)\in D} \min_{||\tilde{\mathbf{w}}-\mathbf{w}(\omega)||\leq\xi} \mathbf{1}(f(\tilde{\mathbf{w}}), l) = |D|$ in training by setting the number of network parameters to be sufficiently larger than the number of training samples.