ENHANCING TRAINING ROBUSTNESS THROUGH INFLUENCE MEASURE

Anonymous authors
Paper under double-blind review

ABSTRACT

In the field of machine learning, the pursuit of robust and accurate models is ongoing. A key aspect of achieving robustness lies in identifying which data points in the training set should be excluded and which high-quality, potentially unlabeled data points outside the training set should be incorporated to improve the model's performance on unseen data. To accomplish this, an effective metric is needed to evaluate the contribution of each data point toward enhancing overall model performance. This paper proposes the use of an influence measure as a metric to assess the impact of training data on test set performance. Additionally, we introduce a data selection method to optimize the training set as well as a dynamic active learning algorithm driven by the influence measure. The effectiveness of these methods is demonstrated through extensive simulations and real-world datasets.

1 INTRODUCTION

To build robust machine learning models, the significance of individual training data points cannot be overstated. Each data point in the training set contributes uniquely to the model's learning process, shaping its performance, generalization, and resilience to various challenges (Blum & Langley, 1997). Simply evaluating the model's performance on the provided data is insufficient; understanding the influence of individual training examples and making informed decisions about their inclusion or exclusion is critical for developing robust and reliable models.

Recent advancements in machine learning have highlighted the importance of strategic data selection and management during training. Techniques such as active learning (Settles, 2009) promote the iterative selection of the most uncertain unlabeled data points for labeling and inclusion in the training set. However, estimating uncertainty in deep neural networks (DNNs) is challenging due to their tendency to exhibit overconfidence (Ren et al., 2021). To address this, methods like deep Bayesian approaches (Gal et al., 2017b), query-by-committee (Gorriz et al., 2017), Variational Auto-Encoders (Sinha et al., 2019), adversarial learning (Ducoffe & Precioso, 2018; Mayer & Timofte, 2020), graph convolutional networks (Caramalau et al., 2021), and noise stability (Li et al., 2024) have been proposed to improve uncertainty estimates. However, these approaches assume a well-trained model and fail to consider how model parameters might evolve when the training data is modified.

To bridge this gap, recent studies have focused on quantifying the impact of individual training examples on model behavior, with the main challenge being the identification of an appropriate evaluation metric. The Shapley value has emerged as a promising solution, inspiring a number of Shapley-value-based approaches (Ghorbani & Zou, 2019; Jia et al., 2019b;a; Ghorbani et al., 2020; Kwon & Zou, 2022; Wang & Jia, 2023). However, these methods often require multiple model retrainings and evaluations, making them computationally expensive.

Techniques such as influence functions (Koh & Liang, 2017; Pruthi et al., 2020; Yeh et al., 2018; Chen et al., 2021) offer insights into the effect of individual data points on model predictions, helping to identify and mitigate harmful or overly influential examples. For instance, Chhabra et al. (2024) apply influence functions to measure the impact of training data on a specific model, improving performance by pruning detrimental data points. Their method is highly efficient as it avoids the need for model retraining. However, these algorithms may still fail in certain cases, as demonstrated in the following simple example.

Binary Classification via Logistic Regression. We consider a logistic regression model $p(x) = (1 + e^{-x\beta})^{-1}$, where $\beta \in \mathbb{R}$ represents the coefficients. For a training set $\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^n$ and a validation set $\mathcal{V}$, the influence function in Chhabra et al. (2024) for example is defined as

$$\mathcal{I}(-x_i) = \sum_{(x,y)\in\mathcal{V}} \partial_\beta L(y, x; \hat{\beta})[\sum_{i=1}^n \partial_\beta^2 L(y_i, x_i; \hat{\beta})]^{-1} \partial_\beta L(y_i, x_i; \hat{\beta}),$$

where $\partial_\beta = \partial/\partial\beta$, $x \in \mathbb{R}$, $y \in \{0, 1\}$ is a binary classification label, L represents the cross-entropy loss, and $\hat{\beta} = \arg\min_{\beta\in\mathbb{R}} n^{-1} \sum_{i=1}^n L(y_i, x_i; \beta)$. According to Chhabra et al. (2024), samples x_i with negative influence function values negatively impact the model's performance on the validation set and should be removed. For all training samples x_i, the term $\sum_{(x,y)\in\mathcal{V}} \partial_\beta L(y, x; \hat{\beta})[\sum_{i=1}^n \partial_\beta^2 L(y_i, x_i; \hat{\beta})]^{-1}$ remains constant, with only $\partial_\beta L(y_i, x_i; \hat{\beta})$ varying. In the simple logistic regression scenario, we have $\partial_\beta L(y_i, x_i; \hat{\beta}) = [(1 + e^{-x_i\hat{\beta}})^{-1} - y_i]x_i$. Assuming $\sum_{(x,y)\in\mathcal{V}} \partial_\beta L(y, x; \hat{\beta})[\sum_{i=1}^n \partial_\beta^2 L(y_i, x_i; \hat{\beta})]^{-1} > 0$ and $x_i > 0$, the training sample influence is negative if and only if $y_i = 1$. However, it is clearly incorrect to solely remove data points from one class, as outliers in the other class may also negatively influence the model's performance. For higher-dimensional x_i, as shown in Section 5 using both simulated and real-world data, our method consistently outperforms the approach proposed by Chhabra et al. (2024) and others.

Given the limitations of existing methods, we draw inspiration from local influence measures developed in the statistical community (Zhu et al., 2007; 2011; Shu & Zhu, 2019; Sui et al., 2023) to propose a novel metric in this article. Our approach differs from previous ones that typically evaluate the impact of perturbations on data samples or model parameters with a fixed model assumption. Instead, our measure directly assesses the effect of minor perturbations to training samples on the model's performance on a test set, allowing the model to adapt to these changes. This method provides a more accurate assessment of training robustness, with broad applications in various downstream tasks. To implement this, we have developed a new perturbation manifold and expanded the local influence framework. Using this innovative approach, we introduce two key metrics: one for data trimming, aimed at identifying and removing training set anomalies that compromise model stability on the test set, and another for active learning, which focuses on selecting the most impactful unlabeled data to enhance training robustness. Moreover, acknowledging the challenges of slow computation and high memory usage inherent in calculating exact local influence measures, we propose two approximation methods. Our experiments on real-world datasets demonstrate that these approximations achieve performance comparable to exact calculations while significantly reducing computational overhead.

We summarize our contributions as follows,

(i) Unlike existing local influence measures, we propose a new metric that evaluates the impact of perturbations to training samples on the model's performance on test data.

(ii) The proposed metric is applicable to both data trimming and active learning. For data trimming, it evaluates the effects of minor perturbations to each sample, offering deeper insights into how individual samples impact training robustness. In the context of active learning, the method captures the relationship between training samples, unlabeled data, and parameter updates.

(iii) We propose two approximation methods to alleviate the high computational cost of calculating local influence measures. These algorithms significantly reduce computational overhead while maintaining better performance than other methods, as demonstrated in our experiments.

2 A New Influence Measure

In this section, we begin by presenting essential background information, including the Perturbation Manifold, before formally defining the proposed metric.

Perturbation Manifold. Our definition of the perturbation manifold closely follows that of Shu & Zhu (2019). Given an input sample $\boldsymbol{z} = (\boldsymbol{x}, y)$ in the training set $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^n$ and a machine learning model with an estimated parameter vector $\hat{\boldsymbol{\theta}}$, which is trained on $\mathcal{Z}$, the prediction probability for class $c \in \{1, \ldots, K\}$ is denoted as $P(c|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$. Let $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_p)^\top$ be a perturbation vector

that varies within an open subset $\Omega \subset \mathbb{R}^p$. The perturbation $\boldsymbol{\omega}$ is applied to $\boldsymbol{x}$, thereby affecting the learning of the parameter vector $\hat{\boldsymbol{\theta}}$. We denote the parameter vector obtained by the model after perturbing the training sample $\boldsymbol{x}$ with $\boldsymbol{\omega}$ as $\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})$ with $\hat{\boldsymbol{\theta}}(\boldsymbol{x}) = \hat{\boldsymbol{\theta}}$. We define $P(c|\boldsymbol{x}+\boldsymbol{\omega}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}))$ as the prediction probability under the perturbation $\boldsymbol{\omega}$ such that $\sum_{c=1}^{K} P(c|\boldsymbol{x}+\boldsymbol{\omega}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})) = 1$. It is assumed that there exists a $\boldsymbol{\omega}_0 \in \Omega$ such that $P(c|\boldsymbol{x}+\boldsymbol{\omega}_0, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0)) = P(c|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$. Additionally, we assume that $\{P(c|\boldsymbol{x}+\boldsymbol{\omega}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}))\}_{c=1}^{K}$ is positive and sufficiently smooth for all $\boldsymbol{\omega} \in \Omega$.

Following the development in (Zhu et al., 2007; 2011), we define $\mathcal{M} = \{P(c|\boldsymbol{x}+\boldsymbol{\omega}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})) : \boldsymbol{\omega} \in \Omega\}$ as a perturbation manifold. The tangent space of $\mathcal{M}$ at $\boldsymbol{\omega}$ is denoted by $T_{\boldsymbol{\omega}}$, which is spanned by $\{\partial l(\boldsymbol{\omega}|c, \boldsymbol{x}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}))/\partial\omega_i\}_{i=1}^{p}$, where $l(\boldsymbol{\omega}|c, \boldsymbol{x}, \hat{\boldsymbol{\theta}}(\boldsymbol{x})) = \log P(c|\boldsymbol{x}+\boldsymbol{\omega}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}))$. Let $\boldsymbol{G}_{\boldsymbol{z}}(\boldsymbol{\omega}) = \sum_{c=1}^{K} \partial_{\boldsymbol{\omega}}^{\top} l(\boldsymbol{\omega}|c, \boldsymbol{x}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}))\partial_{\boldsymbol{\omega}} l(\boldsymbol{\omega}|c, \boldsymbol{x}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}))P(c|\boldsymbol{x}+\boldsymbol{\omega}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}))$ with $\partial_{\boldsymbol{\omega}} = (\partial/\partial\omega_1, \ldots, \partial/\partial\omega_p)$. If $\boldsymbol{G}_{\boldsymbol{z}}(\boldsymbol{\omega})$ is positive definite, then $\mathcal{M}$ is a Riemannian manifold (Shu & Zhu, 2019) with $\boldsymbol{G}_{\boldsymbol{z}}(\boldsymbol{\omega})$ serving as the Riemannian metric tensor (Amari, 2012; Amari & Nagaoka, 2000).

Although $\boldsymbol{G}_{\boldsymbol{z}}(\boldsymbol{\omega})$ is often not positive definite in classification problems, we can still reduce the dimensionality of the perturbations and reconstruct a Riemannian manifold (Shu & Zhu, 2019).

The Influence Measure. Let $L(y', \boldsymbol{x}'; \boldsymbol{\theta})$ denote the loss function of the model with parameter $\boldsymbol{\theta}$ on $\boldsymbol{z}' = (\boldsymbol{x}', y') \notin \mathcal{Z}$, we can get the expression for the (first-order) influence measure:

$$\mathrm{FI}(\boldsymbol{z}', \boldsymbol{z}) = \partial_{\boldsymbol{\omega}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0))\boldsymbol{G}_{\boldsymbol{z}}^{\dagger}(\boldsymbol{\omega}_0)\partial_{\boldsymbol{\omega}}^{\top} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0)), \tag{2.1}$$

where $\boldsymbol{G}_{\boldsymbol{z}}^{\dagger}(\boldsymbol{\omega}_0)$ is the pseudoinverse of $\boldsymbol{G}_{\boldsymbol{z}}(\boldsymbol{\omega}_0)$.

We consider a linear perturbation approach where applying perturbation $\boldsymbol{\omega}$ to the training sample $\boldsymbol{z}$ transforms $(\boldsymbol{x}, y)$ into $(\boldsymbol{x}+\boldsymbol{\omega}, y)$. Consequently, the parameter vector updates to $\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}) := n^{-1} \arg\min_{\boldsymbol{\theta}\in\Theta}\{\sum_{i=1}^{n} L(y_i, \boldsymbol{x}_i; \boldsymbol{\theta}) + L(y, \boldsymbol{x}+\boldsymbol{\omega}; \boldsymbol{\theta}) - L(y, \boldsymbol{x}; \boldsymbol{\theta})\}$, with $\boldsymbol{\omega}_0 = \mathbf{0}$. Using the chain rule, we can derive the expression for $\partial_{\boldsymbol{\omega}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0))$:

$$\partial_{\boldsymbol{\omega}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0)) = \partial_{\boldsymbol{\theta}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))\partial_{\boldsymbol{\omega}} \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}. \tag{2.2}$$

Here, $\partial_{\boldsymbol{\omega}} \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0} \approx n^{-1}\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1}\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$, whose derivation is detailed in Appendix A), where $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}} := n^{-1}\sum_{i=1}^{n} \partial_{\boldsymbol{\theta}}^2 L(y_i, \boldsymbol{x}_i; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$. The term $\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$ represents the gradient of the loss function L first taken with respect to the model parameters $\boldsymbol{\theta}$ and then with respect to $\boldsymbol{x}$, evaluated at the perturbed training sample $(\boldsymbol{x}, y)$.

For the computation of $\boldsymbol{G}_{\boldsymbol{z}}(\boldsymbol{\omega}_0)$, $P(c|\boldsymbol{x}+\boldsymbol{\omega}_0, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0))$ can be directly obtained using the learned model parameter vector $\hat{\boldsymbol{\theta}}$ and the unperturbed sample point $\boldsymbol{x}$. The calculation of $\partial_{\boldsymbol{\omega}} l(\boldsymbol{\omega}_0|c, \boldsymbol{x}, \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$ requires the application of the chain rule:

$$\begin{aligned} \partial_{\boldsymbol{\omega}} l(\boldsymbol{\omega}_0|c, \boldsymbol{x}, \hat{\boldsymbol{\theta}}(\boldsymbol{x})) =& \partial_{\boldsymbol{\omega}} \log P(c|\boldsymbol{x}+\boldsymbol{\omega}_0, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0)) \\ =& \partial_{\boldsymbol{\theta}} \log P(c|\boldsymbol{x}+\boldsymbol{\omega}_0, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0)) \cdot \partial_{\boldsymbol{\omega}} \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})|_{\omega=\boldsymbol{\omega}_0} \\ &+ \partial_{\boldsymbol{x}} \log P(c|\boldsymbol{x}+\boldsymbol{\omega}_0, \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}_0)). \end{aligned} \tag{2.3}$$

All differentiation operations can be easily computed using backpropagation (Goodfellow et al., 2016) in deep learning libraries such as TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2017). This entire process is efficient and does not require retraining the model parameters.

Theorem 1. If $\boldsymbol{\varphi}$ represents a diffeomorphism of $\boldsymbol{\omega}$, then $\mathrm{FI}(\boldsymbol{z}', \boldsymbol{z})$ is invariant under any reparameterization associated with $\boldsymbol{\varphi}$.

Compared to widely used measures in Euclidean spaces, such as the Jacobian norm (Novak et al., 2018) and Cook's local influence measure (Cook, 1986), Theorem 1 demonstrates that $\mathrm{FI}(\boldsymbol{z}', \boldsymbol{z})$ remains invariant under any diffeomorphic transformation (e.g., scaling) of the perturbation vector $\boldsymbol{\omega}$. The proof of Theorem 1 can be found in Shu & Zhu (2019).

The significance of Theorem 1 is especially pronounced when there are scale differences among the dimensions of $\boldsymbol{x}$. For instance, if certain dimensions have significantly larger values than others, the contribution of perturbations to those dimensions may appear exaggerated. However, our $\mathrm{FI}(\boldsymbol{z}', \boldsymbol{z})$ mitigates this scaling issue by employing the metric tensor of the perturbation manifold instead of that of the standard Euclidean space.

3 FI FOR DATA TRIMMING.

The primary goal of data trimming is to eliminate training samples that may compromise the model's performance on datasets beyond the training set. Since our proposed influence measure (FI) quantifies the impact of each training sample on the model's performance on test sets, it serves as a natural tool for data trimming.

From a model robustness perspective, if a small perturbation in a training sample leads to a significant effect on the model's performance on the test set, that sample should be excluded, which aligns with the principle of our proposed metric. Given the inherent challenge of ensuring that all samples in the training set are entirely accurate, a sample with excessive influence could severely degrade the model's overall performance if it contains any contamination. Therefore, to enhance robustness, such samples should be removed from the training set.

Following the setup of Chhabra et al. (2024), we introduce a training set $\mathcal{Z}$, a validation set $\mathcal{V}$, and a base model $\mathcal{F}$. In this context, we assess the impact of each training sample on the model's performance by computing the FI for each training tuple $z_i \in \mathcal{Z}$ with respect to $\mathcal{V}$. To achieve this, we extend Equation 2.1 to encompass the entire validation set. This involves replacing the loss function for an individual validation sample with the mean loss function across the entire validation set, as follows:

$$\mathrm{FI}^{util}(\boldsymbol{z}) = \partial_{\boldsymbol{\omega}} L(\mathcal{V}; \hat{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\omega}_0)) \boldsymbol{G}_{\boldsymbol{z}}^{\dagger}(\boldsymbol{\omega}_0) \partial_{\boldsymbol{\omega}}^{\top} L(\mathcal{V}; \hat{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\omega}_0)),$$

where $\partial_{\boldsymbol{\omega}} L(\mathcal{V}; \hat{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\omega}_0)) := \frac{1}{|\mathcal{V}|} \sum_{(\boldsymbol{x}', y') \in \mathcal{V}} \partial_{\boldsymbol{\omega}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\omega}_0))$.

The algorithm for computing FI^{util} is outlined in Algorithm 1. After computing these values, we can sort all data points in the training set in descending order based on their FI^{util} values and remove the top b points to enhance the model's performance on the test set. For the detailed data trimming algorithm, please refer to Algorithm 3.

Algorithm 1 Calculation of FI^{util}

Input: Training set $\mathcal{Z}$, Validation set $\mathcal{V}$, Base model $\mathcal{F}$
Output: Influence measure vector $\mathrm{FI}^{util} \in \mathbb{R}^{|\mathcal{Z}| \times 1}$

procedure FI^{util}-CALCULATION($\mathcal{Z}$, $\mathcal{V}$, $\mathcal{F}$)
  Train $\mathcal{F}$ with $\mathcal{Z}$, and obtain the parameter vector $\hat{\boldsymbol{\theta}}$
  Generate an empty vector FI^{util} of size $|\mathcal{Z}| \times 1$
  Calculate $\frac{1}{n} \sum_{(\boldsymbol{x}', y') \in \mathcal{V}} \partial_{\boldsymbol{\theta}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}})$ and $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}$
  for every $\boldsymbol{z}_i$ in $\mathcal{Z}$ **do**
    Calculate $\boldsymbol{G}_{\boldsymbol{z}_i}(\boldsymbol{\omega}_0)$ and $\partial_{\boldsymbol{x}} \partial_{\boldsymbol{\theta}} L(y_i, \boldsymbol{x}_i; \hat{\boldsymbol{\theta}})$
    $\partial_{\boldsymbol{\omega}} L(\mathcal{V}; \hat{\boldsymbol{\theta}}) \leftarrow \frac{1}{n} \sum_{(\boldsymbol{x}', y') \in \mathcal{V}} \partial_{\boldsymbol{\theta}} L(y', \boldsymbol{x}'; \hat{\boldsymbol{\theta}}) \boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \partial_{\boldsymbol{x}} \partial_{\boldsymbol{\theta}} L(y_i, \boldsymbol{x}_i; \hat{\boldsymbol{\theta}})$
    $\mathrm{FI}^{util}[i] \leftarrow \partial_{\boldsymbol{\omega}} L(\mathcal{V}; \hat{\boldsymbol{\theta}}) \boldsymbol{G}_{\boldsymbol{z}_i}^{\dagger}(\boldsymbol{\omega}_0) \partial_{\boldsymbol{\omega}}^{\top} L(\mathcal{V}; \hat{\boldsymbol{\theta}})$
  end for
  return FI^{util}
end procedure

4 FI FOR ACTIVE LEARNING.

In active learning, the primary objective is to identify the most uncertain or informative samples from an unlabeled pool for annotation, typically in sequential batches. After each round of annotation, the newly labeled data are combined with the existing labeled set to retrain the model and improve its performance. Li et al. (2024) argue that if a small perturbation to the model parameters leads to significant changes in the prediction for a given sample, this indicates high uncertainty for that sample under the current model, suggesting that it should be labeled and added to the training set. Our approach is more fundamental: since the model parameters are derived from the training data, we directly assess how perturbations to the training samples influence the model's predictions for the unlabeled samples. If slight perturbations to most training samples substantially alter the model's

prediction for a given sample, it likely contains missing information from the training set and should therefore be included.

We now introduce a method for active learning using the proposed FI. For an unlabeled sample $\boldsymbol{x}_{unlabel}$, we first assign it a predicted label and treat it as a validation sample. Next, we calculate the influence measure of $\boldsymbol{x}_{unlabel}$ with respect to each point in the training set. The overall influence measure $\mathrm{FI}^{active}(\boldsymbol{x}_{unlabel})$ is derived by aggregating these individual measures either by averaging or using specific quantiles of these FI values. The $\mathrm{FI}^{active}(\boldsymbol{x}_{unlabel})$ is defined as follows:

$$\mathrm{FI}^{active}(\boldsymbol{x}_{unlabel}) = g(\{\mathrm{FI}(\boldsymbol{z}_{unlabel}, \boldsymbol{z}_i)\}_{i=1}^{n}), \tag{4.1}$$

where g represents the aggregation function, $\boldsymbol{z}_{unlabel} = (\boldsymbol{x}_{unlabel}, y_{pred})$, and y_{pred} is the predicted label assigned by the current model for $\boldsymbol{x}_{unlabel}$.

During each round of active learning, we begin by computing FI^{active}. We then sort all the samples in the unlabeled pool in descending order based on their FI^{active} values. The top-ranked samples are labeled and added to the training set. In the subsequent round, the model is retrained, FI^{active} is recalculated, and the process is repeated.

The active learning process described above has two main drawbacks. First, computing FI^{active} requires substantial storage for second-order derivatives, particularly in models with numerous parameters or high-dimensional data. To address this issue, we propose the KFSVD approximation method, which combines K-FAC and Truncated SVD to effectively reduce storage requirements. Second, the necessity to recalculate FI^{active} for all unlabeled data in each round significantly decreases computational efficiency. To mitigate this, we implement a subsampling approximation that enhances overall performance. The details of these two approximation methods are as follows.

Algorithm 2 Calculation of FI^{active}

Input: Labeled pool of training data $\mathcal{L}$, Unlabeled pool of training data $\mathcal{U}$, Base model $\mathcal{F}$, Aggregation function g, Truncated SVD parameter k
Output: Influence measure vector $\mathrm{FI}^{active} \in \mathbb{R}^{|\mathcal{U}|\times 1}$

procedure FI^{active}-CALCULATION($\mathcal{L}, \mathcal{U}, \mathcal{F}, g, k$)
  Train $\mathcal{F}$ with $\mathcal{L}$, and obtain the parameter vector $\hat{\boldsymbol{\theta}}$
  Generate an empty vector FI^{active} of size $|\mathcal{U}| \times 1$
  Calculate $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}$ with K-FAC approximation
  for every $x_i^{\mathcal{U}}$ in $\mathcal{U}$ **do**
    Obtain an estimated label $\hat{y}_i$ with $\mathcal{F}$
    Calculate $\partial_{\boldsymbol{\theta}} L(\hat{y}_i, \boldsymbol{x}_i^{\mathcal{U}}; \hat{\boldsymbol{\theta}})$
    $\mathcal{J} \leftarrow \emptyset$
    for every $z_k^{\mathcal{L}}$ in $\mathcal{L}$ **do**
      Calculate $\boldsymbol{G}_{\boldsymbol{z}_k^{\mathcal{L}}}(\boldsymbol{\omega}_0)$
      Calculate $\Lambda_{k\times k}$, $U_{m\times k}$ and $V_{d\times k}$ with power iteration
      $\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}} L(y_k^{\mathcal{L}}, \boldsymbol{x}_k^{\mathcal{L}}; \hat{\boldsymbol{\theta}}) \leftarrow U_{m\times k}\Lambda_{k\times k}V_{d\times k}^{\top}$
      $\partial_{\boldsymbol{\omega}} L(\hat{y}_i, \boldsymbol{x}_i^{\mathcal{U}}; \hat{\boldsymbol{\theta}}) \leftarrow \partial_{\boldsymbol{\theta}} L(\hat{y}_i, \boldsymbol{x}_i^{\mathcal{U}}; \hat{\boldsymbol{\theta}}) \boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1} \partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}} L(y_k^{\mathcal{L}}, \boldsymbol{x}_k^{\mathcal{L}}; \hat{\boldsymbol{\theta}})$
      $\mathcal{J} \leftarrow \mathcal{J} \bigcup \{\partial_{\boldsymbol{\omega}} L(\hat{y}_i, \boldsymbol{x}_i^{\mathcal{U}}; \hat{\boldsymbol{\theta}}) \boldsymbol{G}^{\dagger}_{\boldsymbol{z}_k^{\mathcal{L}}}(\boldsymbol{\omega}_0) \partial_{\boldsymbol{\omega}}^{\top} L(\hat{y}_i, \boldsymbol{x}_i^{\mathcal{U}}; \hat{\boldsymbol{\theta}})\}$
    end for
    $\mathrm{FI}^{active}[i] \leftarrow g(\mathcal{J})$
  end for
  return FI^{active}
end procedure

KFSVD approximation, which combines the Kronecker-factored (K-FAC) approximation (Martens & Grosse, 2020; Nickl et al., 2024) with Truncated Singular Value Decomposition (Truncated-SVD) approximation (Golub & Reinsch, 1971) to mitigate memory consumption associated with the Hessian matrix $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}$ and the second-order partial derivatives $\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$. The K-FAC algorithm is a widely recognized technique for approximating the Hessian matrix, which not only accelerates computations but also significantly reduces storage requirements. Meanwhile, the Truncated-SVD

approximation employs power iteration and related techniques to compute the top-k eigenvalues and their corresponding eigenvectors of $\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}}L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$, thereby providing an effective approximation of these second-order partial derivatives. Assuming the dimensionality of the sample covariates is d and the number of model parameters is m, the Truncated-SVD approximation enables the decomposition of the second-order partial derivatives as $\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}}L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})) = U_{m\times k}\Lambda_{k\times k}V_{d\times k}^{\top}$, where $\Lambda_{k\times k}$ is a diagonal matrix containing the top-k eigenvalues of $\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}}L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$ and $U_{m\times k}$ and $V_{d\times k}$ are comprised of k orthogonal vectors. Consequently, the Truncated-SVD approximation reduces the storage requirement for the second-order derivatives from $m \times d$ to $(m + d + 1) \times k$. Algorithm 2 provides a detailed procedure for calculating FI^{active} using the KFSVD approximation.

Subsampling approximation, which utilizes subsampling and random forest techniques to enhance computational efficiency. Specifically, we extract a small subset of samples (e.g., 20%) from the unlabeled pool and accurately compute their FI^{active}. These computed values are then used to sort the samples. The features of each sample, along with their corresponding ranks, serve as covariates and target variables to create a new dataset. Subsequently, we train a regression model using random forests on this dataset and leverage the trained model to predict the ranks of other samples. Selection in each round is based on these predicted ranks. Since the computation of FI^{active} is the most time-consuming part of the workflow, the overall acceleration is directly correlated with the proportion of samples for which we choose to compute FI^{active} accurately. For instance, selecting 20% of the samples can reduce the total processing time to $\frac{1}{5}$ of the original duration.

In practice, we integrate both approximation methods to develop a comprehensive FI^{active}-based active learning algorithm. This algorithm effectively addresses the storage and computational efficiency challenges inherent in calculating FI^{active}. The detailed procedure is outlined in Algorithm 4.

5 Experimental Results

In this section, we present experimental results that illustrate how our newly proposed metrics, FI^{util} and FI^{active}, contribute to enhancing model robustness. We compare our algorithms with state-of-the-art strategies in both data trimming and active learning scenarios, thereby validating the effectiveness of our approach on both simulated and real-world datasets.

5.1 Data Trimming

In this subsection, we conduct simulations using both linear and nonlinear models to demonstrate how our algorithm enhances data trimming efficiency and evaluate the effectiveness of the proposed FI on real-world datasets. The latest data trimming method, *Influence Value* (IV), introduced by Chhabra et al. (2024), serves as the primary baseline for comparison in these experiments.

Validation on 2D Linear Model. Logistic regression is utilized for this binary classification task. We begin by generating several datasets by sampling from two isotropic 2D Gaussian distributions. Each dataset comprises 150 training samples, 100 validation samples, and 600 test samples. The experimental settings for this scenario are consistent with those of the study by Chhabra et al. (2024). To account for the randomness inherent in the sampling process, we analyze our method on datasets generated under the same distribution but with different random seeds. As shown in Table 1, our method consistently enhances model performance compared to theirs in most cases, particularly when trimming 5, 10, and 20 samples, with thirty different random seeds employed each time. Moreover, it is evident from Figure 1.C that IV tends to trim samples from a specific class under certain conditions. In this context, Figure 1.D clearly demonstrates that in some scenarios IV fails, while our method continues to perform effectively.

Table 1: Comparison of two methods on linear model. Number of cases where FI outperforms IV across 30 random seeds, along with performance improvements. *Acc_FI*: the mean accuracy by FI, and *Acc_IV*: by IV.

# of deleted points	# of better case	Acc_FI(%)	Acc_IV(%)
5	23	96.22±0.65	95.77±0.76
10	28	96.20±0.65	94.84±1.07
20	30	96.23±0.64	93.36±2.21

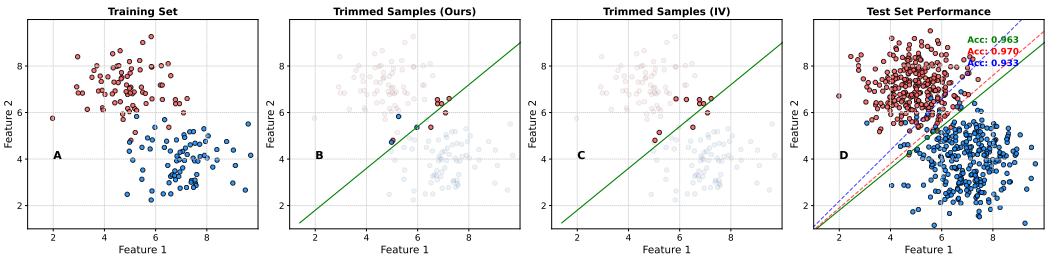

Figure 1: Performance under Linear Model. Different colored points represent different classes. **A** shows the training set. **B** and **C** respectively denote the samples to be trimmed by FI and IV. **D** denotes test set. **Green** line: boundary without trimming; **Red** line: boundary after FI trimming; and **Blue** line: boundary after IV trimming.

Validation on 2D Nonlinear Model. After demonstrating the effectiveness of our method in linear scenarios, we now extend our examination to nonlinear cases. To achieve this, we construct a binary classification dataset that is non-linearly separable, with each class represented by a crescent-shaped region. A more intuitive understanding can be obtained from Figure 9. We employ a neural network that includes an input layer, two hidden layers with ReLU activation functions, and an output layer with a sigmoid activation function. Similar to the linear case, we conduct repeated experiments in the nonlinear scenario. Each dataset comprises 500 training samples, 250 validation samples, and 250 test samples. As shown in Table 2, our method achieves a higher average accuracy and outperforms the other methods in most cases across the 20 repetitions. Figure 9 illustrates instances where utilizing IV to remove training points can lead to a deterioration in model performance under certain conditions.

Table 2: Comparison of two methods on nonlinear model. Number of cases where FI outperform IV across 20 random seeds, along with performance improvements. *Acc_FI*: the mean accuracy by FI, and *Acc_IV*: by IV.

# of deleted points	# of better case	Acc_FI(%)	Acc_IV(%)
5	17	89.90±2.16	87.50±2.46
10	17	90.24±2.05	88.02±2.62
20	15	90.32±1.67	87.80±2.54

Validation on Real-World Datasets. We evaluate the effectiveness of our proposed methods using four real-world datasets: two tabular datasets, *Adult* (Kohavi et al., 1996) and *Bank* (Moro et al., 2014); a visual dataset, *CelebA* (Liu et al., 2018); and a textual dataset, *Jigsaw Toxicity* (Noever, 2018). Additional information regarding the datasets and experimental details can be found in Appendices C and D.1. Both FI^{util} and IV are evaluated on the validation set, with Logistic Regression serving as the base model. The results on the test sets of these datasets are presented in Figure 2.

As illustrated in Figure 2, our findings indicate that under a limited budget b, our FI^{util} data trimming method consistently outperforms two baseline models, IV and Random Trimming. Among the four datasets examined, Random Trimming demonstrates no improvement in model performance. Although IV exhibits a notable enhancement on *Adult* and *Bank*, it tends to remove important data points on *CelebA*, resulting in decreased performance, and shows no improvement on *Jigsaw Toxicity* compared to Random Trimming. This suggests that IV may fail in certain scenarios. In contrast, our method consistently achieves the maximum improvements across all datasets, particularly on the *Bank* dataset, where accuracy increases by more than 10%. Interestingly, our method performs better on real-world datasets than on simulated ones. This is largely due to the simplicity of the simulated datasets, which are two-dimensional with clear boundaries effectively separating the classes and contain no erroneous data points. In such cases, a sufficiently large dataset allows the model to easily find the optimal boundary, minimizing the advantages of data trimming. In contrast, real-world datasets are typically high-dimensional and more complex, often containing errors. Here, the benefits of removing potentially erroneous high-influence points become more evident. To support this, we introduce noise into the real-world datasets and conduct further experiments; details of the noise method are in Appendix D.1, and results are shown in Figure 8. The findings clearly demonstrate that

both FI and IV outperform Random Trimming, with our method retaining significant advantages over IV, further confirming its robustness.

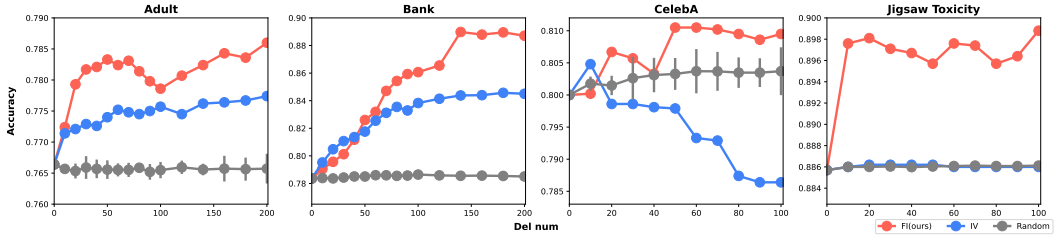

Figure 2: Accuracy curves of three data trimming methods on test sets of *Adult*, *Bank*, *CelebA* and *Jigsaw Toxicity*.

5.2 Active Learning

In this section, we conduct numerical experiments on three simple tabular datasets and three complex image datasets to demonstrate the effectiveness of the proposed FI metric in active learning. We compare our method with two state-of-the-art active learning baselines including IV (Chhabra et al., 2024) and BALD (Gal et al., 2017a; Kirsch et al., 2019; 2023). Random Selection is also included as a baseline. All reported results are averaged over three runs to ensure a reliable evaluation.

Note that, IV, as proposed by Chhabra et al. (2024), originally calculates influence values only once during the initial selection, which can hinder overall performance since these values are not updated with subsequent labelings. To ensure fairness in our comparisons, we modified IV to recalculate influence values in each round of selection, and this updated version is used in our experiments.

For our method, the aggregation function $g(\{\mathrm{FI}(\boldsymbol{z}_{unlabel}, \boldsymbol{z}_i)\}_{i=1}^n)$ in Equation 4.1 denotes the operation of calculating the mean of the data points from the set $\{\mathrm{FI}(\boldsymbol{z}_{unlabel}, \boldsymbol{z}_i)\}_{i=1}^n$ that fall between the 10th and 90th percentiles, sorted in descending order. In other words, we compute the mean of the middle 80% of the data after sorting. This approach effectively excludes extreme values, thereby enhancing the robustness of our final result.

Tabular Datasets. For the tabular datasets, we employed a Logistic Regression model. The total number of annotation rounds for each experiment, along with the settings for the unlabeled pool size and acquisition size, are detailed in Table 4. As shown in Figure 3, our method outperforms the other approaches on these simple tabular datasets. Random Selection performs reasonably well initially, but as data volume increases, improvements in model performance diminish. BALD shows comparable performance to Random Selection on tabular datasets. IV initially underperforms compared to Random Selection, but as more data is added, its performance improves and eventually becomes comparable to BALD. Our method starts similarly to BALD, but its focus on more challenging samples allows it to quickly address missing information once the model achieves a certain accuracy. As a result, in the latter stages of each graph, our method distinctly diverges from the others.

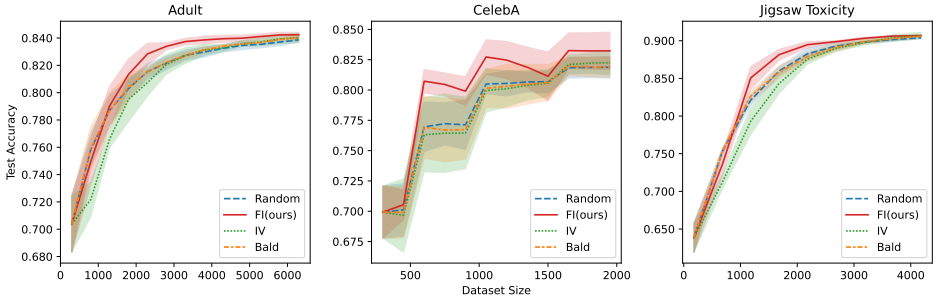

Figure 3: Classification performance due to the different active learning methods on *Adult*, *CelebA*, and *Jigsaw Toxicity*.

Image Classification. The data in the tabular datasets have been processed, resulting in high test accuracy with logistic regression. In contrast, for the image datasets—*MNIST* (LeCun et al., 1998), *EMNIST* (Cohen et al., 2017), and *CIFAR-10* (Krizhevsky & Hinton, 2009)—the input consists of raw images, and we employ a Convolutional Neural Network (CNN) as the classifier. To simplify the computation of FI^{active}, we utilize the outputs from the last layer of the neural network and focus on the parameters of that layer. However, during each training iteration, all parameters of the neural network are updated, not just those of the final layer. The settings for the unlabeled pool size and acquisition size for each image dataset are provided in Table 5. Experimental results indicate that our method maintains comparable time consumption to other approaches, even with complex image data (see Table 7 for details). Our primary focus in active learning is enhancing model accuracy rather than speed. We developed CNNs tailored for *MNIST* and *EMNIST*, with specifications in Table 6. For *CIFAR-10*, we adopted a model architecture from Trockman & Kolter (2023). As shown in Figure 4, both BALD and IV methods are suited to different scenarios, but our method consistently outperforms others across all datasets, particularly in more complex situations. The relatively straightforward *MNIST* dataset does not fully demonstrate our method's advantages, so we created more challenging unbalanced and redundant *MNIST* datasets. Comparisons on these datasets reveal our method as the most effective, with advantages even more pronounced than on the original *MNIST*, as illustrated in Figure 10. Additionally, Figure 5 visually represents the selection preferences of FI^{active}, highlighting its tendency to identify points that are more challenging for the current model to distinguish.

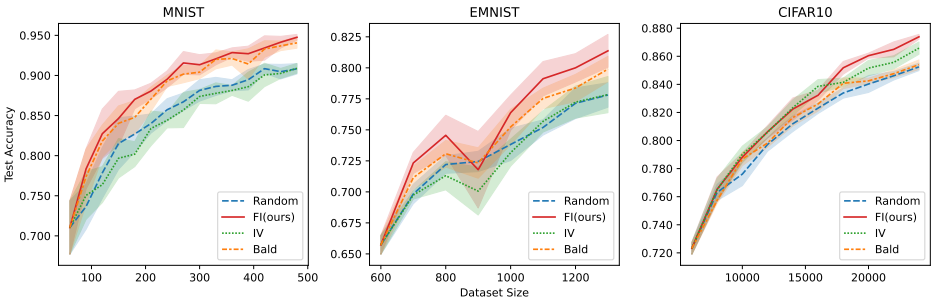

Figure 4: Classification performance due to the different active learning methods on *MNIST*, *EMNIST*, and *CIFAR-10*.

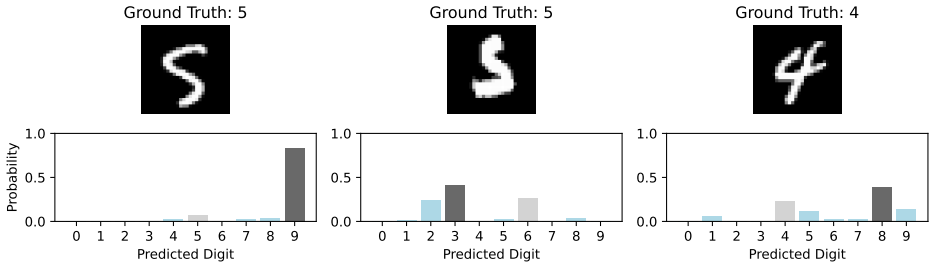

Figure 5: Images with the highest FI^{active} selected in the first round of active learning for the *MNIST* problem, along with their corresponding prediction probability distributions.

6 Conclusion

In this paper, we introduce a novel local influence metric that evaluates the impact of perturbations to training samples on model performance concerning test samples. This metric is applicable in both data trimming and active learning, offering valuable insights into the contributions of individual samples and their relationships with unlabeled data.

Furthermore, we propose two approximation methods to mitigate the computational costs associated with calculating local influence measures. Our experimental results demonstrate that these algorithms effectively reduce costs while outperforming other state-of-the-art methods.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

Shun-ichi Amari. *Differential-geometrical methods in statistics*, volume 28. Springer Science & Business Media, 2012.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000.

Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.

Razvan Caramalau, Binod Bhattarai, and Tae-Kyun Kim. Sequential graph convolutional network for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9583–9592, 2021.

Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. Hydra: Hypergradient data relevance analysis for interpreting deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7081–7089, May 2021. doi: 10.1609/aaai.v35i8.16871. URL https://ojs.aaai.org/index.php/AAAI/article/view/16871.

Anshuman Chhabra, Peizhao Li, Prasant Mohapatra, and Hongfu Liu. "what data benefits my classifier?" enhancing model performance and interpretability through influence-based data selection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HE9eUQlAvo.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

R Dennis Cook. Assessment of local influence. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 48(2):133–155, 1986.

Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1183–1192. PMLR, 2017a. URL http://proceedings.mlr.press/v70/gal17a.html.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pp. 1183–1192. PMLR, 2017b.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Amirata Ghorbani, Michael Kim, and James Zou. A distributional framework for data valuation. In *International Conference on Machine Learning*, pp. 3535–3544. PMLR, 2020.

Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Handbook for Automatic Computation: Volume II: Linear Algebra*, pp. 134–151. Springer, 1971.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Marc Gorriz, Axel Carlier, Emmanuel Faure, and Xavier Giro-i Nieto. Cost-effective active learning for melanoma segmentation. *arXiv preprint arXiv:1711.09168*, 2017.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019a.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1167–1176. PMLR, 2019b.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 7024–7035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/95323660ed2124450caaac2c46b5ed90-Abstract.html.

Andreas Kirsch, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frédéric Branchaud-Charron, and Yarin Gal. Stochastic batch acquisition: A simple baseline for deep active learning. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=vcHwQyNBjW.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pp. 202–207, 1996.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8780–8802. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/kwon22a.html.

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

Xingjian Li, Pengkun Yang, Yangcheng Gu, Xueying Zhan, Tianyang Wang, Min Xu, and Chengzhong Xu. Deep active learning with noise stability, 2024.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2020. URL https://arxiv.org/abs/1503.05671.

Christoph Mayer and Radu Timofte. Adversarial sampling for active learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3071–3079, 2020.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Peter Nickl, Lu Xu, Dharmesh Tailor, Thomas Möllenhoff, and Mohammad Emtiyaz Khan. The memory perturbation equation: Understanding model's sensitivity to data, 2024. URL https://arxiv.org/abs/2310.19273.

David Noever. Machine learning suites for online toxicity detection. *arXiv preprint arXiv:1810.01869*, 2018.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. URL https://api.semanticscholar.org/CorpusID:40027675.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/e6385d39ec9394f2f3a354d9d2b88eec-Paper.pdf.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Burr Settles. Active learning literature survey. 2009.

Hai Shu and Hongtu Zhu. Sensitivity analysis of deep neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33014943. URL https://doi.org/10.1609/aaai.v33i01.33014943.

Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5972–5981, 2019.

Yang Sui, Yukun Huang, Hongtu Zhu, and Fan Zhou. Adversarial learning of distributional reinforcement learning. In *International Conference on Machine Learning*, pp. 32783–32796. PMLR, 2023.

Asher Trockman and J. Zico Kolter. Patches are all you need? *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=rAnB7JSMXL.

Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 6388–6421. PMLR, 2023.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Chih-Kuan Yeh, Joon Sik Kim, Ian E. H. Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks, 2018.

Hongtu Zhu, Joseph G. Ibrahim, Sikyum Lee, and Heping Zhang. Perturbation selection and influence measures in local influence analysis. *The Annals of Statistics*, 35(6):2565–2588, 2007. ISSN 00905364. URL http://www.jstor.org/stable/25464601.

Hongtu Zhu, Joseph G Ibrahim, and Niansheng Tang. Bayesian influence analysis: a geometric approach. *Biometrika*, 98(2):307–323, 2011.

A Derivation of Expression for $\partial_{\boldsymbol{\omega}}\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}$

Recall that $\hat{\boldsymbol{\theta}}(\boldsymbol{x})$ minimizes the empirical risk: $R(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^{n} L(y_i, \boldsymbol{x}_i; \boldsymbol{\theta})$. We assume that R is twice-differentiable and strongly convex in $\boldsymbol{\theta}$, i.e., $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}} := \partial^2_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}) = \frac{1}{n}\sum_{i=1}^{n} \partial^2_{\boldsymbol{\theta}} L(y_i, \boldsymbol{x}_i; \hat{\boldsymbol{\theta}})$ exists and is positive definite. This ensures the availability of $\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1}$, which will be utilized in the subsequent derivation. The perturbed parameter vector $\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})$ can be written as

$$\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\theta}\in\Theta}\{R(\boldsymbol{\theta}) + \frac{1}{n}L(y, \boldsymbol{x}+\boldsymbol{\omega}; \boldsymbol{\theta}) - \frac{1}{n}L(y, \boldsymbol{x}; \boldsymbol{\theta})\}. \tag{A.1}$$

Define the parameter change $\delta(\boldsymbol{\omega}) = \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}) - \hat{\boldsymbol{\theta}}(\boldsymbol{x})$, and note that, since $\hat{\boldsymbol{\theta}}(\boldsymbol{x})$ does not depend on $\boldsymbol{\omega}$, the quantity we aim to calculate can be expressed in terms of it: $\partial_{\boldsymbol{\omega}}\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}) = \partial_{\boldsymbol{\omega}}\delta(\boldsymbol{\omega})$.

According to the definition of $\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})$, we know

$$\mathbf{0} = \partial_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})) + \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})) - \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})). \tag{A.2}$$

Next, since $\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega}) \to \hat{\boldsymbol{\theta}}(\boldsymbol{x})$ as $\boldsymbol{\omega} \to \mathbf{0}$, we perform a Taylor expansion of the right-hand side:

$$\begin{aligned}
\mathbf{0} &= \partial_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})) + \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})) - \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})), &\text{(A.3)}\\
&\approx [\partial_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x})) + \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})) - \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))]\\
&\quad + [\partial^2_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x})) + \frac{1}{n}\partial^2_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})) - \frac{1}{n}\partial^2_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))]\delta(\boldsymbol{\omega}). &\text{(A.4)}
\end{aligned}$$

Solving for $\delta(\boldsymbol{\omega})$, we get

$$\begin{aligned}
\delta(\boldsymbol{\omega}) \approx &- [\partial^2_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x})) + \frac{1}{n}\partial^2_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})) - \frac{1}{n}\partial^2_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))]^{-1}\\
&\cdot [\partial_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x})) + \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})) - \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))].
\end{aligned} \tag{A.5}$$

Since $\hat{\boldsymbol{\theta}}(\boldsymbol{x})$ minimizes R, we have $\partial_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}) = 0$. We further assume that $\partial^2_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))$ is continuous on $\boldsymbol{x}$, then we have

$$\delta(\boldsymbol{\omega}) \approx -\frac{1}{n}[\partial^2_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}(\boldsymbol{x}))]^{-1}[\frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}+\boldsymbol{\omega}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})) - \frac{1}{n}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x}))]. \tag{A.6}$$

After differentiation, the final expression can be obtained,

$$\partial_{\boldsymbol{\omega}}\hat{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\omega})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0} = \partial_{\boldsymbol{\omega}}\delta(\boldsymbol{\omega})\big|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0} \approx -\frac{1}{n}\boldsymbol{H}_{\hat{\boldsymbol{\theta}}}^{-1}\partial_{\boldsymbol{x}}\partial_{\boldsymbol{\theta}} L(y, \boldsymbol{x}; \hat{\boldsymbol{\theta}}(\boldsymbol{x})). \tag{A.7}$$

B Algorithms

Algorithm 3 Data Trimming using FI^{util}

Input: Training set $\mathcal{Z}$, Validation set $\mathcal{V}$, Base model $\mathcal{F}$, Budget b
Output: Trimmed Dataset $\mathcal{Z}'$

1: **procedure** DataTrimming($\mathcal{Z}$, $\mathcal{V}$, $\mathcal{F}$, b)
2: Call the algorithm FI^{util}-Calculation($\mathcal{Z}$, $\mathcal{V}$, $\mathcal{F}$) to obtain $\text{FI}^{util} \in \mathbb{R}^{|\mathcal{Z}|\times 1}$
3: Select b samples $\boldsymbol{z}_i \in \mathcal{Z}$ as $\{Z_b\}$, whose $\text{FI}^{util}[i]$ rank in the top b
4: $\mathcal{Z}' \leftarrow \mathcal{Z} \setminus \{Z_b\}$
5: **return** $\mathcal{Z}'$
6: **end procedure**

Algorithm 4 Active Learning using FI^{active}

Input: Labeled pool of training data $\mathcal{L}$, Unlabeled pool of training data $\mathcal{U}$, Base model $\mathcal{F}$, Number of samples for annotation (per round) N, Aggregation function g, Truncated SVD parameter k, Subsampling rate $p \in [0, 1]$
Output: Updated labeled pool $\mathcal{L}$

procedure ACTIVELEARNING($\mathcal{L}, \mathcal{U}, \mathcal{F}, N, g, k, p$)
  for $j \leftarrow 1$ to NUM_rounds **do**
    Subsample $\mathcal{U}_p \subseteq \mathcal{U}$ s.t. $x \in \mathcal{U}_p$ w.p. p for all $x \in \mathcal{U}$
    Call the algorithm FI^{active}-CALCULATION($\mathcal{L}, \mathcal{U}_p, \mathcal{F}, g, k$) to obtain $\text{FI}^{active} \in \mathbb{R}^{|\mathcal{U}_p| \times 1}$
    for $x_{i'} \in \mathcal{U}_p$ **do**
      $r_{i'} \leftarrow$ is the rank of $\text{FI}^{active}[i']$ sorted in ascending order
    end for
    Train a random forest h with $\{(x_{i'}, r_{i'})\}_{i'=1}^{\mathcal{U}_p}$
    for $x_i \in \mathcal{U}$ **do**
      $\hat{r}_i \leftarrow h(x_i)$
    end for
    Select N samples x_i as $\{X_N\}$, whose $\hat{r}_i$ ranks in the top N
    Take $\{X_N\}$ out of $\mathcal{U}$, and query their labels $\{Y_N\}$
    Update $\mathcal{L} \leftarrow \mathcal{L} \bigcup \{X_N, Y_N\}$
  end for
  return $\mathcal{L}$
end procedure

C DATA SOURCES

Adult. The Adult dataset consists of 48,842 instances and 14 features, including categorical and integer types. Extracted from the 1994 Census database by Barry Becker, it focuses on predicting whether an individual's annual income exceeds $50,000. Records were filtered based on criteria such as age, gross income, and work hours, making this dataset a valuable resource for classification tasks in social science and *Access Link*: *Adult Database*

Bank. The Bank dataset pertains to direct marketing campaigns conducted by a Portuguese banking institution, focusing on phone call outreach. The primary objective of this dataset is to classify whether a client will subscribe to a term deposit, indicated by the binary variable (yes/no). This dataset is derived from multiple marketing campaigns, which often required several contacts with the same client to ascertain their interest in the product. *Access Link*: *Bank Database*

CelebA. The CelebA dataset is a large-scale facial attribute dataset containing over 200,000 celebrity images, each annotated with 40 attribute labels. This dataset serves as a valuable resource for tasks such as facial recognition, attribute prediction, and generative modeling. *Access Link*: *CelebA Database*

Jigsaw Toxicity. The Jigsaw dataset of Wikipedia consists of comments from online platforms that have been labeled for toxicity. It contains a large number of comments, with 28 features of syntax, sentiment, emotion and outlier word dictionaries. The dataset is commonly used for training and evaluating machine learning models aimed at detecting harmful or inappropriate content in user-generated text. *Access Link*: *Jigsaw Toxicity Database*

MNIST. The MNIST database is a large collection of handwritten digits that is widely used for training and testing in the field of machine learning. This dataset contains 70,000 images of handwritten digits (0-9), each of which is a 28×28 pixel grayscale image. *Access Link*: *MNIST Database*

EMNIST. The Extended MNIST database consists of handwritten character digits sourced from the NIST Special Database 19, formatted as 28×28 pixel images to align with the structure of the MNIST dataset. There are six different splits provided in this dataset, and we use the EMNIST Letters with 145,600 characters. *Access Link*: *EMNIST Database*

CIFAR10. The CIFAR10 database, developed by the Canadian Institute for Advanced Research, is a widely utilized collection of images for training machine learning and computer vision algorithms. It

consists of 60,000 color images, each measuring 32×32 pixels, categorized into 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each class contains 6,000 images. *Access Link*: *CIFAR10 Database*

D EXPERIMENT DETAILS

D.1 EXPERIMENT DETAILS IN 5.1

Data Construction for Real-World Datasets. In this experiment, the preprocessing methods for all real-word datasets are consistent with those used in Chhabra et al. (2024). For *CelebA*, we use the extracted features provided by the authors Liu et al. (2015), and for *Jigsaw Toxicity*, we obtain text embeddings using the MiniLM transformer model (Wang et al., 2020). Further details are provided below.

- **Adult.** This dataset contains 37,692 samples, with 30,162 for training and 7,530 for testing. There are 102 features, and the target is to predict if income exceeds $50k (yes) or not (no).
- **Bank.** This dataset consists of 30,490 samples, divided into 18,292 training samples and 12,198 test samples. There are 50 features, and the target is to predict if the client will subscribe a term deposit (yes/no).
- **CelebA.** This dataset includes 104,163 samples, with 62,497 for training and 41,666 for testing. There are 39 features, and the aim is to predict whether a person is smiling (yes) or not (no).
- **Jigsaw Toxicity.** This dataset consists of 30,000 samples, split into 18,000 training samples and 12,000 test samples. There are 385 features, and the target is to determine if a tweet is toxic (yes) or not (no).

Given that the initial test accuracy of the model on the original datasets generally exceeds 90%, the impact of data trimming is minimal. Accordingly, we randomly sample 5,000 instances from the original training set to serve as the new training set, and 4,200 instances from the test set to serve as the new test set.

Data Construction for Noisy Real-World Datasets. Consider the additional experiments on data trimming presented in Figure 8. For *Adult*, *Bank*, and *Jigsaw Toxicity*, our construction method follows the same approach as in the Real-World Datasets experiment, with the additional step of randomly sampling 500 instances from the training sets and adding white noise with a variance of 0.1. For *CelebA*, since the variables are binary (-1, 1), we introduced noise by randomly selecting six feature columns and flipping their values (transforming -1 to 1 and 1 to -1). This approach generates a noisy dataset for supplementary experiments.

Random Trimming. We randomly remove b (budget) data points from the training sets under five random seeds, and evaluate the average performance on the test sets in each iteration.

Experimental Procedure and Parameter Settings. First, a model is trained on the initial dataset. Based on this trained model, we then implement three data trimming strategies, removing b data points. Finally, we retrain the model to evaluate the effectiveness of the different trimming strategies. The experimental procedure is illustrated in Figure 6, highlighting the key steps involved in our study. Additionally, the parameter settings are detailed in Table 3.

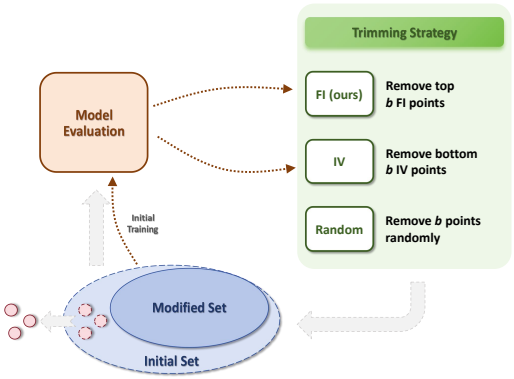

Figure 6: Flowchart of data trimming.

Table 3: Parameter Settings of Data Trimming.

	Adult	Bank	CelebA	Jigsaw Toxicity
optimizer	SGD	SGD	Adam	Adam
learning_rate	1e-2	1e-2	1e-4	1e-2
weight_decay	1e-2	1e-2	1e-6	1e-2
	Adult +noise	**Bank +noise**	**CelebA +noise**	**Jigsaw Toxicity +noise**
optimizer	SGD	SGD	SGD	Adam
learning_rate	1e-2	1e-2	1e-2	1e-1
weight_decay	1e-2	1e-4	1e-6	1e-4

D.2 Experiment Details in 5.2

Data Construction for Unbalanced and Redundant MNIST. For **Unbalanced MNIST**, we set the total sample size to 27,500. Categories 0 to 4 are each allocated an equal sample size, representing 1/55 of the total sample. Similarly, categories 5 to 9 are assigned equal sample sizes, with each constituting 10/55 of the total sample. This allocation strategy ensures a deliberate imbalance among the classes. Samples are systematically drawn based on these ratios from the original dataset to create this new unbalanced dataset. For **Redundant MNIST**, the task is delineated to classify solely the digits 1 and 7, presented in equal proportions. If the acquisition function selects an input from any class other than 1 or 7, the labeling function designates a "neither" category. This setup leads to a three-way classification scheme during training, categorized as 1 vs. 7 vs. neither. This design allows us to explore the effectiveness of the learning model in dealing with class imbalance and partial class information, critical aspects in real-world applications where similar conditions are often encountered.

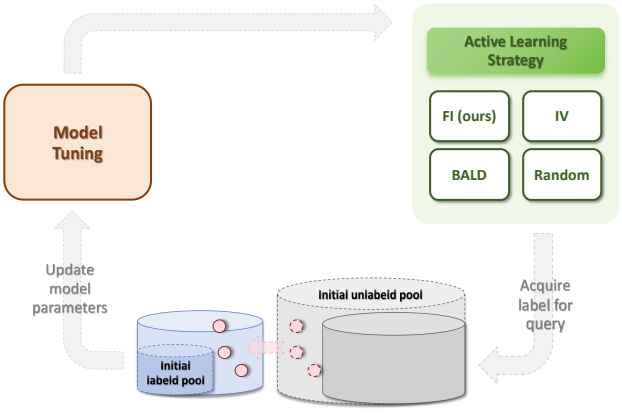

Figure 7: Flowchart of active learning.

Active Learning Experiment. Tables 4 and 5 detail the parameter settings for active learning with tabular and image data, respectively. Table 6 describes the neural network architectures employed for *MNIST* and *EMNIST*.

Table 4: Active Learning Experiment Configuration for Tabular Datasets

Attribute	Adult	CelebA	Jigsaw Toxicity
Number of Classes	2	2	2
Rounds	12	11	8
Initial Pool	300	300	180
Unlabeled Pool Size	5000	3000	5000
Acquisition Size	500	150	500

Table 5: Active Learning Experiment Configuration for Image Datasets

Attribute	MNIST	EMNIST	CIFAR10
Number of Classes	10	37	10
Rounds	14	10	9
Initial Pool	60	600	6000
Unlabeled Pool Size	420	2000	10000
Acquisition Size	30	100	2000

Table 6: Architecture of MNIST CNN

Layer Type	Activation	Output Dimensions (incl. Padding)
Conv2d	ReLU	(32, 14, 14), P=1
Conv2d	ReLU	(64, 7, 7), P=1
Dropout	-	-
Linear	ReLU	256
Linear	-	num_classes

E ADDITIONAL EXPERIMENTS

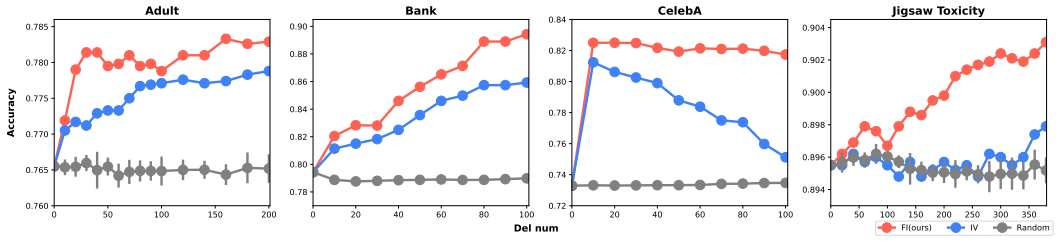

Figure 8: Accuracy curves of three data trimming methods on test sets of *Adult*, *Bank*, *CelebA* and *Jigsaw Toxicity*.

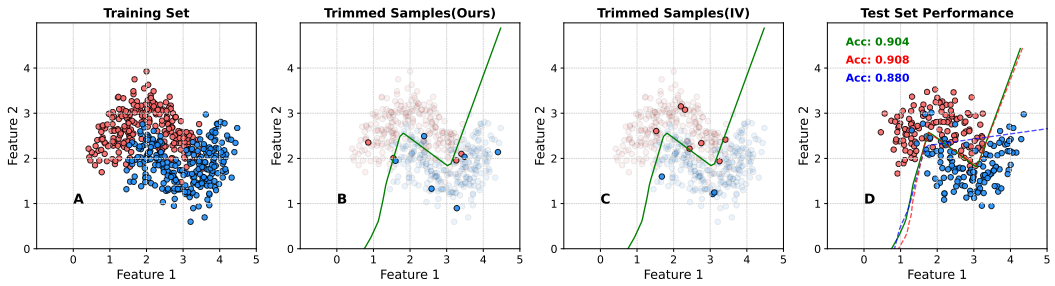

Figure 9: Performance under Nonlinear Model. Different colored points represent different classes. **A** shows the training set. **B** and **C** respectively denote the samples to be trimmed by FI method and IV method. **D** denotes test set. **Green** line: boundary without trimming. **Red** line: boundary after FI trimming. **Blue** line: boundary after IV trimming.

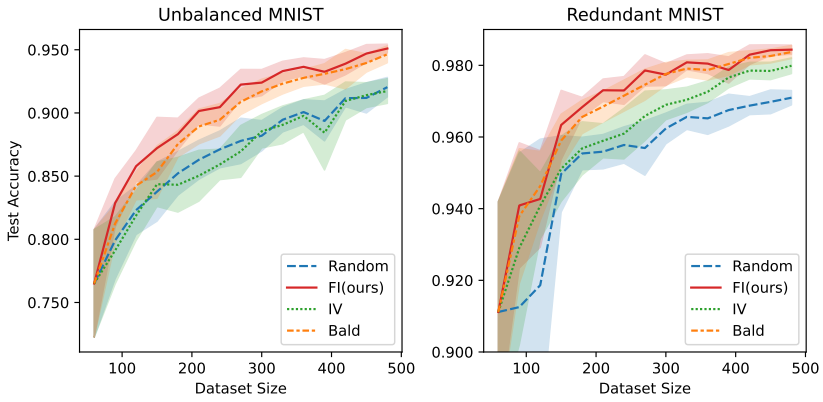

Figure 10: Classification performance due to the different active learning methods on *Unbalanced MNIST*, and *Redundant MNIST*.

Table 7: Time (seconds) for active learning algorithms over toy case on *MNIST*, *EMNIST*, and *CIFAR-10*.

Methods	MNIST	EMNIST	CIFAR-10
FI(ours)	19	45	372
IV	13	15	204
BALD	9	15	214

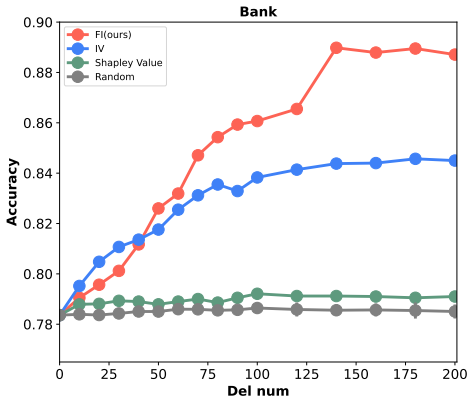

Figure 11: Performance comparison of different methods on *Bank*, including the newly added Shapley method.

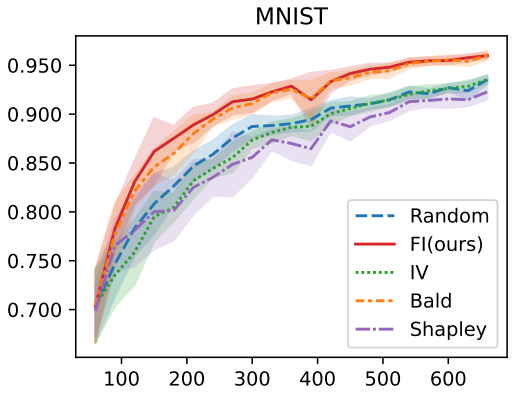

Figure 12: Performance comparison of different methods on *MNIST*, including the newly added Shapley method.