Data Glitches Discovery using Influence-based Model Explanations

Nikolaos Myrtakis
myrtakis@csd.uoc.gr
ETIS Lab, ENSEA
Cergy, France
University of Crete
Heraklion, Greece

Ioannis Tsamardinos
tsamard.it@gmail.com
University of Crete
Heraklion, Greece

Vassilis Christophides
vassilis.christophides@ensea.fr
ETIS Lab, ENSEA
Cergy, France

Abstract

We address the problem of detecting data glitches in ML training sets, specifically mislabeled and anomalous samples. Detection of data glitches provides insights into the quality of the data sampling. Their repair may improve the reliability and the performance of the model. The proposed methodology is based on exploiting influence functions that estimate how much the loss of the model (or a given sample) is affected when a sample is removed from the training set. We introduce three novel signals for detecting, characterizing, and repairing data glitches in a training set based on sample influences. Influence-based signals form an explainable-by-design data glitch detection framework, producing intuitively explainable signals of the actual predictive model built. In contrast, specialized algorithms that are agnostic to the target ML model (e.g., anomaly detectors) replicate the work of fitting the data distribution and may detect glitches that are inconsistent with the decision boundary of the predictive model. Computational experiments on tabular and image data modalities demonstrate that the proposed signals outperform, in some cases up to a factor of 6, all existing influence-based signals, and generalize across different datasets and ML models. In addition, they often outperform specialized glitch detectors (e.g., mislabeled and anomaly detectors) and provide accurate label repairs for mislabeled samples.

This work has been accepted for publication in the research track of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) [35]

Keywords: Other

CCS Concepts

• **Information systems** → **Data cleaning**; • **Computing methodologies** → **Machine learning**.

Keywords

Training Data Debugging, Data Debugging for ML, Mislabeled Samples, Anomalies, Influence Functions, Influence Signals

ACM Reference Format:

Nikolaos Myrtakis, Ioannis Tsamardinos, and Vassilis Christophides. 2025. Data Glitches Discovery using Influence-based Model Explanations. In *Proceedings of the 31st ACM SIGKDD Conf. on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3690624.3709285

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709285

1 Introduction

Machine Learning (ML) has become an essential tool for gaining insights and making data-driven decisions in science and industry. However, the success of ML projects is heavily dependent on the quality of data used to train models. As a matter of fact, even the most performant algorithms trained on mislabeled, or underrepresented data may lead to models that make inaccurate predictions and do not generalize well in real-world settings [4, 15, 54].

In particular, we are focusing on data glitches that break the following assumptions made by traditional ML algorithms, such as: (a) the observed labels of training samples are correct [49]; (b) the feature values of training samples are consistent and occupy different regions of the input data space per class [8]. Unfortunately, these assumptions are rarely met in real-world settings jeopardizing the performance of ML models with nefarious consequences for business applications and people's lives[1]. As a matter of fact, the ratio of corrupted labels in real-world datasets is estimated to be between 8.0% and 20% [29, 48].

In this paper, we propose an *explainable-by-design data glitch detection framework* that allows analysts to identify and explain different types of data glitches during the data modeling. In this respect, we leverage recent advances in *influence-based explanation methods* [16] to propose novel interpretable signals for *detecting*, *repairing*, and *characterizing* both *singular* and *mixtures* of data glitches in *train* sets. Specifically, we compute the influence of each training sample on a *small-but-clean validation* set to identify the train glitches. We argue that *different types of glitches have distinct influence signatures on the decision boundary* of a trained model compared to clean samples. Moreover, our signals[2] can be used with any influence estimator and classification model trained with gradient descent or similar variants.

To the best of our knowledge, there is no previous study addressing how existing influence-based signals (i) can effectively detect different types of data glitches, under the same influence function, ML models, and benchmark datasets, (ii) can provide accurate label repairs for the mislabeled samples and (iii) are compared to glitch detectors w.r.t. detection performance. Motivated by these questions we propose three novel signals to accurately *detect*, *characterize*,

[1] https://www.forbes.com/sites/gilpress/2021/06/16/andrew-ng-launches-a-campaign-for-data-centric-ai/
[2] We use the term signals and influence signals interchangeably

and *repair* both types of data glitches under the same influence framework. In a nutshell, the main contributions of our work are:

- To identify the mislabelled, anomalous, and mixed types of glitches we introduce three novel signals, called CNCI, PCID, and CFRank, respectively. Conceptually, we categorize the signals into a new family of signals, that we call *counterfactual* signals, examining the change of a sample's influence if its label had been different.
- We show that the counterfactual signals provide accurate label repairs for the mislabeled samples, showing the potential of influence-based explanations to detect and repair data imperfections.
- We conduct the first comparative evaluation of the four existing and the three proposed influence-based signals w.r.t. their detection performance for both mislabeled samples and anomalies under the same influence estimator, ML models, and datasets for two tabular and image data modalities.
- For low-class noise ratio, CNCI outperforms the state-of-the-art mislabeled detection signal, *Self-Influence*, by a margin of 65% (F1) on average, as well as the mislabel detector CleanLab[39] by a margin of 75% (F1) on average. For anomalies, PCID outperforms up to a factor of 6 the existing influence signals, as well as three anomaly detectors in two out of the three datasets while being the most robust.
- Ablation studies on glitch ratios and validation set size show that CNCI and PCID are effective and robust even when only 2% of the samples are reserved as a validation set.

Our study provides the following insights: (i) mislabeled samples and anomalies have fundamentally different influence signatures during model training that lead to their detection and characterization, and (ii) anomalies proved to be easy-to-fit samples having an extremely low positive influence on each foundation model.

The remainder of the paper is organized as follows. Section 2 presents the notation used in our work and the foundations of influence functions. Section 3 describes the different types of data glitches (mislabeled and anomalous samples) studied in our work. Section 4 introduces the three influence-based signals. Section 5 details the experimental testbed. Section 6 reports the results of our experimental evaluation. Section 7 positions our work w.r.t. existing influence-based signals used to detect data glitches in training sets. Finally, Section 8 summarizes our contributions and draws directions for future work.

2 Influence Functions

Influence Functions (*IFs*) [7, 17, 24] aim to reveal the training samples that are responsible for the prediction of another sample during training or testing and hence enable *instance-based* explanations.

Here we introduce the notation that we will use throughout the paper. Let D_{tr} be a training dataset $D_{tr} = \{(x, \tilde{y})\}_{i=1}^{n}$ that comprises n pairs of feature vectors $x \in \mathbb{R}^d$ and possibly noisy labels $\tilde{y} \in \mathbb{Z}$. We denote $C \in \mathbb{Z}^k$ as the set containing the k classes, i.e., the domain of the noisy label vector $\tilde{Y}$. In our work, we use a separate clean validation set to clean the training glitches that we denote as $D_{val} = \{(x, y)\}_{i=1}^{m}$. Note that y signifies the true class of a sample, which is not assumed to be given for D_{tr}. A classification model is denoted as $f_\theta : X \rightarrow \tilde{Y}$, parameterized by $\theta \in \mathbb{R}^p$, where p is the total number of parameters. These parameters are obtained via the optimization of a loss function $\mathcal{L} : \tilde{Y} \times \tilde{Y} \rightarrow \mathbb{R}$. Formally, given a train dataset D_{tr} the optimal parameters are obtained by minimizing the empirical risk (ERM): $\hat{\theta} = \text{argmin}_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(z_i; \theta)$, where $z_i = (x_i, \tilde{y}_i) \in D_{tr}$.

Finally, we denote the influence of a sample z_i to another z_j via an IF $\mathcal{I} : \mathbb{R}^d \rightarrow \mathbb{R}$ as $\mathcal{I}(z_i \rightarrow z_j)$.

Deletion diagnostics, i.e., the influence of a sample on the model via its deletion, have been introduced using the *Leave-One-Out-Retraining* (LOOR) method. LOOR is defined as follows: $\mathcal{I}_{LOOR}(z_i \rightarrow z_j) = \mathcal{L}(z_j; \theta_{\setminus z_i}) - \mathcal{L}(z_j; \theta)$.

Intuitively, a *negative* $\mathcal{I}_{LOOR}(z_i \rightarrow z_j)$ indicates that z_j's loss has *decreased* when z_i was absent. *Positive* $\mathcal{I}_{LOOR}(z_i \rightarrow z_j)$ means that the absence of z_i led to an *increase* in loss of z_j. Other works refer to positive / negative influencers as *proponents* / *opponents* [43] or *excitatory* / *inhibitory* samples [57]. Although LOOR makes fewer assumptions than other IFs, it is infeasible for models with a moderate size of parameters even on small datasets, as LOOR requires $n + 1$ retrainings to convergence.

For this reason, *gradient-based* IFs have been introduced [1, 5, 19, 24, 26, 27, 43, 50, 57] to estimate the *leave-one-out* effect without re-training models. Gradient-based IFs could be either *static* or *dynamic* [16]. The former estimates the samples' influence using the final model, i.e., at the end of the training phase. The latter provides a more fine-grained view of influence by unrolling the gradients throughout the training process, capturing the training dynamics. We stress that the choice of IF is orthogonal to our work, as long as the influence estimator is accurate. Thus, we employ the dynamic estimator TracIn [43] rather than static estimators due to limitations studied by recent works [2, 16, 45, 46]. Finally, since the objective of our work is to use IFs, Representer Point-based influence estimators [50, 57] are not studied in this work.

TracIn [43] is a dynamic IF that estimates how the parameter updates, caused by a sample z_i, affected the loss of z_j through time, i.e., across epochs: $\sum_{t=1}^{T} \mathcal{L}(z_j, \theta_{t,i}) - \mathcal{L}(z_j, \theta_{t+1,i})$, where $\theta_{t,i}$ are the model's parameters at an epoch t after updating with a sample z_i and T is the total number of epochs. Taking the first-order Taylor approximation on $\mathcal{L}(z_j, \theta_{t+1,i})$, the TracIn-Ideal becomes $\mathcal{I}_{TracInId}(z_i \rightarrow z_j) = \sum_{t=1}^{T} \eta_t \nabla\mathcal{L}(z_j, \theta_{t,i}) \nabla\mathcal{L}(z_i, \theta_{t,i})$, where η_t is the learning rate at the iteration t. An important assumption of $\mathcal{I}_{TracInId}$ is that the model is built using stochastic gradient descent with singular batch size (B=1). As this is not a realistic assumption, a batched version of TracIn-Ideal has been proposed:

$$\mathcal{I}_{TracIn}(z_i \rightarrow z_j) = \sum_{t=1, z_i \in B_t}^{T} \frac{\eta_t}{|B_t|} \nabla\mathcal{L}(z_j, \theta_t) \nabla\mathcal{L}(z_i, \theta_t), \quad (1)$$

where B_t is the sample batch at iteration t that contains z_i. Note that in $\mathcal{I}_{TracIn}$ the parameter updates are also affected by other samples in B_t, however, the batched version approximates well the estimation of TracInId [43]. In our work, we use $\mathcal{I}_{TracIn}$ as an IF to assess the accuracy of the proposed signals to detect data glitches.

3 Data Glitches

In this section, we describe the main types of data glitches that we address in this work, namely, *mislabeled* and *anomalous* samples. Some glitches may arise due to human or script errors in the

data acquisition, transmission, and collection processes, while others are a natural product of the intrinsic nature of the domains of interest. We should stress that we focus on *non-deliberate* data glitches in *training* sets that may incur systematic bias in ML models. Deliberate glitches may include data poisoning and backdoor attacks [53]. Finally, *test set* glitches e.g. subpopulation shift and out-of-distribution samples [31] are beyond the scope of this work.

3.1 Mislabeled Samples

Recall that y refers to the true unknown class of a training sample and $\tilde{y}$ refers to the possibly noisy label, as recorded in the dataset. If $\tilde{y} \neq y$ the sample is considered *mislabelled*. Label errors occur when (i) the human annotators face imperfect/difficult patterns or when the task is subjective [33], (ii) non-expert crowdsourcing platforms, such as Amazon MTurk, are used due to time and budget constraints, often lead to unreliable labels [41], or (iii) automated classification systems are used [42].

In our work, we assume that label errors or class noise are generated from a stochastic process that is either independent or dependent w.r.t. the sample features [11, 40, 49]. In this work, we focus on error generation mechanisms that are conditionally independent of the features given the true class $P(\tilde{y}|X, y) = P(\tilde{y}|y)$, Hence, Feature-Dependent class noise is beyond the scope of this work.

Uniform Class Noise. In this type of noise, also known as *symmetric* noise, the true label of a sample $y = i$ is flipped to another label $\tilde{y} = j$ with equal probability. Given a set of C classes, $P(\tilde{y} = j|y = i) = \frac{\epsilon}{|C|-1}, \forall j \neq i \wedge i, j \in C$ and $P(\tilde{y} = i|y = i) = 1 - \epsilon, \forall i \in C$, where ϵ denotes the flip probability.

Class-Dependent Noise. In this type of noise, also known as *asymmetric* noise, the true label of a sample is more likely to be flipped to a specific class. Given a set of classes C, class-dependent noise is defined as: $\max_{c \in C \setminus \{i,j\}} P(\tilde{y} = c|y = i) < P(\tilde{y} = j|y = i) < \epsilon$. Note that in asymmetric noise $P(\tilde{y} = i|y = i) = 1 - \epsilon, \forall i \in C$.

Although the common assumption when building robust models is the uniform class noise, where most ML models are proved to be tolerant, class-dependent noise may pose a serious risk to models' performance resulting in up to $\sim$ 20% drop in test accuracy for 20% noise [40]. This is particularly important for deep neural networks that can overfit a training set even with a high ratio of corrupted labels due to a large number of parameters, resulting in poor generalization performance [58].

3.2 Anomalous Samples

An anomaly is a sample that is irregular from the remainder of the samples in the dataset. Anomalies can occur due to data entry errors, bugs in data wrangling and preprocessing software, sensor faults, etc. Unlike Out-Of-Distribution (OOD) samples [10] observed in test sets, in this work we study the influence of anomalous samples in a *training* set where their feature representation significantly deviates from the rest of the samples. According to [3], an anomaly x should satisfy $p(x) < \lambda$, where λ is a density threshold and p the probability density function, indicating that anomalies often lie in sparse, low-density areas.

In our work, we consider clustered anomalies, i.e., samples sharing common characteristics that have not been previously seen during training (aka novelties) and outlier samples that have been corrupted under a common corruption mechanism. As frequently observed in tabular data[3] novelties form low-density clusters while outlier samples are *scattered*. In this work, we examine the case of Far Clustered-Anomalies (Far-CA), as studied in [56]. The key difference with [56], is that in our work we consider the presence of Far-CA in training instead of the test set. Such samples can be for example undetected Out-of-Distribution (OOD)[4] examples slipped into the training set. The objective is to detect and inform a human analyst about their presence as they come from a different distribution. Note that these anomaly notions are well-studied for images but they are not directly applicable to tabular datasets. Although many tabular anomaly detection datasets exist, their ground truth is for unsupervised anomaly detection. In contrast, we solve a classification problem using the target ML model and aim to detect data glitches based on their impact on the model's decision boundary.

Similarly to mislabeled samples, the effect of anomalies in the context of ML has been studied mainly under the lens of model performance. Several empirical studies have shown that anomalies do not significantly affect the model performance [19, 28, 37]. However, the influence of anomalies in the formation of the model's decision boundary is left unexplored. In other words, does the reported insubstantial impact on performance also mean low influence?

4 Influence-based Signals

Despite the substantial efforts on the development of more accurate influence estimators, it remains open how one can leverage the output of an IF to form informative signals to accurately detect *training* data glitches, characterize their *type* or propose *repairs* when mixtures of glitches exist in a complex ML dataset.

The influence scores between different pairs of samples computed by an IF form a matrix that we call *Influence Matrix* (*IM*). We define an $IM^t : \mathbb{R}^{n \times m}$ as a $n \times m$ matrix that contains the sample-wise influences of the n training samples to m samples, at a specific training epoch t with model parameters denoted as θ_t. Each cell i, j of IM^t indicates the sample-wise influence $\mathcal{I}^t(x_i \rightarrow x_j)$.

In the case of TracIn IF, the final $IM = \sum_i^T IM^t$. An *influence-signal* is then defined as an aggregation A over the rows or columns of the *IM*. In our work, we calculate the *train-to-validation* influence, which captures the impact of each training on each validation sample, resulting in an $IM : \mathbb{R}^{|D_{tr}| \times |D_{val}|}$. To discover training glitches, we consider row-wise aggregations, i.e., $A(IM_{i,:})$ measuring how the training i-th sample affects another subgroup of validation samples.

The computation of train-to-validation influence reveals interesting semantics that relate to the model's generalization performance. It is important to note that the validation set must be clean to reveal training data errors [59]. This assumption is reasonable as our approach is part of the target model training, where a clean reference set is necessary to tune, evaluate, and select the final ML model. The typical size of a validation set is 10-20% of the total dataset size, which is challenging in practical applications as it requires manual inspection and substantial resources. For this reason, we empirically show in our experiments that the proposed signals are

[3] https://odds.cs.stonybrook.edu/
[4] https://github.com/Jingkang50/OpenOOD

effective even in the case that the validation set occupies 2% of the total dataset size.

4.1 Counterfactual Influence Signals

In this section, we formalize the following desiderata that lead to defining the novel signals:

- D1. *Sign*-wise influence aggregations. Distinguishing between the negative and positive influential samples when aggregating conveys information for the presence of glitches.
- D2. *Conditional* influence aggregations. The *class-conditioned* influence of z_i to a subgroup of samples may provide less noisy influence score aggregations [38].
- D3. *Magnitude* of aggregated influence. The magnitude reveals how much impact z_i has on a collection of samples. Extreme or unusual values (*negative*, *positive*, or *near zero*) may reveal glitches with different intrinsic characteristics.
- D4. *Repair Actions*. An influence signal should provide repairs, when possible e.g. in case of mislabeled samples, that respect the decision boundary of the target ML model.

As depicted in Table 2 of Section 7 none of the existing influence signals address all the previous desiderata. More importantly, existing signals do not provide repair actions, especially for mislabeled samples. This motivated the introduction of a novel family of influence signals, namely *Counterfactual Signals*. A counterfactual signal aims essentially to answer *how does the relabelling of a sample alter its influence?*. In a binary classification problem, it is trivial to relabel each sample to the opposite class and recalculate the influence. However, in multi-class settings, the problem becomes harder as we need to run the IF for each modified label vector Y'. In the following, we present the rationale behind the two counterfactual signals we propose for mislabeled and anomalous samples. Then, we introduce a combination of these two signals to detect arbitrary mixtures of both types of data glitches.

Our signals assume that the final IM is computed based on all epochs. The computational complexity of the IM is $O(Tnmd)$, where T is the number of epochs where the model checkpoints were saved, the n is the number of the training samples, m is the number of validation samples and d is the number of parameters of the layers whose gradients were used to compute the influence. Moreover, we note that $\mathcal{I}(z_i \rightarrow z_j)$ and $\mathcal{I}(z_i = (x_i, \tilde{y}_i) \rightarrow z_j)$ are equivalent. Finally, we define conditional influence as $\mathcal{I}(z_i \rightarrow z_j|c) = \mathcal{I}(z_i \rightarrow z_j)\mathbb{1}_{\{\tilde{y}_j=c\}}$. Similarly, conditional joint influence is defined as $\mathcal{I}(z_i \rightarrow D_{val}|c) = A([\mathcal{I}(z_i \rightarrow z_j|c)]_{z_j \in D_{val}})$, where A is an aggregation function applied in each $z_j \in D_{val}$ of class c.

Conditional Negative Counterfactual Influence (CNCI). The first signal proposed in our work aims to detect mislabeled samples and suggest accurate label repairs. CNCI translates various characteristics of mislabeled samples into an influence signal. The main characteristics of mislabeled samples (MS) as observed by prior works [12, 18, 49] are: (i) they often cause out-of-sample performance degradation, (ii) they lie in a dense area of a class but are labeled differently from their peers, and (iii) they exhibit high loss as the model struggles to fit them given their feature values and the assigned (observed) label. The steps of the CNCI for a single training sample are presented in Algorithm 1. To encode the first characteristic, CNCI keeps only *negative* influence scores in the IM, as those scores indicate a harmful relation between training and validation samples, fulfilling the D1. The second characteristic implies that the negative influence of MS is not symmetric towards the whole validation set but *conditional* towards the class of their peers that are labeled differently. Thus, CNCI computes the cumulative negative influence of each training sample per class, resulting in a vector of $|C|$ elements, fulfilling the D2. The most interesting element of this vector is the class affected the *most* negatively, as the gradient updates of MS point to a different direction than the samples of this class.

We call the class with the most negative conditional influence the *negative counterfactual (ncf)* class. The *ncf* class is the key of CNCI and is computed per training sample, fulfilling the desiderata D1-D3 (sign, magnitude, and conditional influence aggregations). To do the detection, CNCI relabels all training samples to their *ncf* class and performs a negative-conditional cumulative aggregation towards their observed class (i.e., initially assigned class based on the final parameter vector θ_T). This simulates how the negative impact of the training on validation samples would have changed for a subsequent hypothetical epoch using the counterfactual label. Intuitively, if the *ncf* is the true class of a mislabeled sample, its negative influence should be small resembling the behavior of a clean sample. In contrast, a clean-labeled sample is expected to have a large negative influence as the *ncf* makes it mislabeled. Therefore, clean samples will receive large negative values and mislabeled samples will receive small values, close to zero.

The computational complexity of CNCI is $O(nm|C|)$, where n is the number of training samples, m is the number of validation samples, and $|C|$ is the number of classes. Note that every component of CNCI is parallelizable as the influence aggregations can be computed independently per training sample and class.

CNCI is the first influence-based signal that hits two targets with one arrow: detection and label repairs using the *ncf* class for the detected MS. This paves the road on the potential that instance-based explanation methods have for detecting and repairing data quality issues using the model's decision boundary and explaining its decisions at the same time.

Algorithm 1: Conditional Negative Counterfactual Influence (CNCI) to detect mislabeled samples

Data: $z_i = (x_i, \tilde{y}_i) \in D_{tr}, D_{val}, IM, \mathcal{I}$
Result: $CNCI_i, rn_i$
$IM^-_{ij} \leftarrow \min(IM_{ij}, 0), \forall i, j$
$ncf_i \leftarrow \underset{c \in C}{\operatorname{argmin}}[\mathbb{1}_{\{y_{val}=c\}}]^T \cdot IM^-_{i,:}$
$CNCI_i \leftarrow \sum_{z_{val} \in D_{val}} \min(\mathcal{I}^T(z'_i = (x_i, ncf_i) \rightarrow z_{val}|\tilde{y}_i), 0)$
return $CNCI_i, ncf_i$

Positive Counterfactual Influence Difference (PCID). We propose the influence signal PCID to detect anomalous samples. To the best of our knowledge, this is the first influence signal aiming to detect anomalies. Prior works have shown [19, 28, 37] that anomalous samples do not substantially impact the performance of classifiers. We should stress that despite the insubstantial impact of anomalies on performance they may still affect the construction of the model's decision boundary locally. In Table 3 in the Appendix,

we show that the models achieve lower loss on anomalies than on clean samples on average, thus, each foundation model learns the anomalous area. PCID translates the observation to an influence signal considering that anomalies have a small positive influence on the whole validation set as they fall in sparse isolated areas and do not interact with the rest of the samples in terms of parameter updates. PCID leverages the counterfactual influence assuming that the influence of an anomalous sample will remain small before and after relabeling it to a different class, aiming to reduce the false detection of clean-but-low-influential training samples.

PCID follows the steps of Algorithm 2 for a single training sample z_i. First, only the positive scores of the IM^+ are retained, while the rest are set to zero. Using the non-negative IM^+, we compute the conditional positive influence vector $\mathbf{u}_i \in \mathbb{R}^k_{\geq 0}$ for each class C. To assess the difference before and after relabelling z_i we use $\mathbf{u}_i$ to compute the *positive counterfactual (pcf)* class if z_i; the *pcf* is the class that the sample affects the most positively and is different from its observed class. The calculation of *pcf* fulfills the desiderata D1-D3. After relabeling z_i to its *pcf* class, a counterfactual positive influence vector $\mathbf{u'}_i$ is computed, similarly to $\mathbf{u}_i$. To do the detection, the infinite norm returns the maximal difference between the two conditional positive influence vectors, weighted by the marginal positive influence of z_i to the validation set. Finally, the inverse weighted infinite norm is computed to assign higher scores to anomalies than the inlier samples.

Similarly to CNCI, PCID is highly parallelizable. It requires $O(nm|C|)$ to be computed. However, in comparison to CNCI, the *pcf* class can not be used for repairs as there is no straightforward repair for anomalies.

Algorithm 2: Positive Counterfactual Influence Difference (PCID) to Detect Anomalies

Data: $z_i = (x_i, \tilde{y}_i) \in D_{tr}, D_{val}, IM, \mathcal{I}$
Result: $PCID_i$

$IM^+_{ij} \leftarrow \max(IM_{ij}, 0)$
$\mathbf{u}_i \leftarrow [[\mathbb{1}_{\{y_{val}=c\}}]^T \cdot IM^+_{i,:}]_{c \in C}$
$pcf_i \leftarrow \underset{c \in C, c \neq \tilde{y}_i}{\operatorname{argmax}} [\mathbb{1}_{\{y_{val}=c\}}]^T \cdot IM^+_{i,:}$
$\mathbf{u}'_i \leftarrow [\sum_{z_{val} \in D_{val}} \max(\mathcal{I}^T(z'_i = (x_i, pcf_i) \rightarrow z_{val}|c), 0)]_{c \in C}$
$w_i \leftarrow \sum_{c \in C} [\mathbb{1}_{\{y_{val}=c\}}]^T \cdot IM^+_{i,:}$
$PCID_i \leftarrow (w_i \cdot ||\mathbf{u}_i - \mathbf{u}'_i||_\infty)^{-1}$
return $PCID_i$

Counterfactual Influence Rank (CFRank). So far, CNCI and PCID target only one type of data glitch. However, in real-world datasets, one may have to deal with a mixture of glitches that pose significant challenges in both their detection and characterization. In this respect, we propose *CFRank*, a rank aggregation signal that combines CNCI and PCID to detect both mislabeled and anomalous samples that may be contained in a training set. Note that CFRank targets samples that can be either mislabeled or anomalous but they do not simultaneously exhibit both types of glitches as in the case of feature-dependent class noise [11, 40]. Detecting and characterizing samples with multiple glitches is left as future work. CFRank first computes the rank of each sample for the CNCI and PCID lists in ascending order using a rank function r; we use the rank since the two signals may contain scores of different scales. Then, CFRank is defined as follows:

$$CFRank = \{max(r(CNCI_i), r(PCID_i))\}_{i=1}^{n}$$

The intuition is that both CNCI and PCID signals assign higher scores to glitches than to clean samples, thus, the max function keeps the most plausible category of the two glitches. This serves to *characterize* the detected glitches. When combining the two signals, CFRank assumes that the mislabeled samples and anomalies are not prioritized simultaneously by the same signal, i.e., CNCI and PCID are mutually exclusive. This requirement is fulfilled because a counterfactual mislabeled sample may retain a large conditional positive influence vector, which is not the case for the anomalies.

5 Experimental Setting

All experiments run on a 16-core Intel Xeon, 64 GB of main memory, and no GPU. The code is available on https://github.com/myrtakis/data-hedgehog.

Datasets. We employ three benchmark image datasets, namely MNIST, Fashion-MNIST, and CIFAR-10, which are widely used in the IF literature [5, 19, 26, 27, 43, 51]. For the tabular datasets we employ Epsilon, Forest Cover and Jannis [14] since the employed ML models perform well for these datasets. For each dataset, we draw a 10% stratified[5] subset uniformly at random to speed up influence computations as performed in [51]. Then we split each subset into 80% for training (D_{tr}) and 20% for validation (D_{val}).

Data glitch injection on training sets. For each D_{tr}, we introduce uniform class noise, by randomly flipping the label of 10% of D_{tr} [24, 38, 43] to a different class uniformly at random. For the class-dependent noise, we select a class c_i uniformly at random that contains the samples S_i and flip the labels of the 10% of S_i to another class c_j, as described in Section 3.1. To inject anomalies we follow the methodology as described in Section 3.2. Specifically, to create far-isolated clusters we first select 10% of the samples from a specific class of MNIST denoted as S_M, Fashion-MNIST denoted as S_{F-M}, and CIFAR-10 denoted as S_C. We then use the following combinations to contaminate the training sets of each dataset: $D_M = D_M \cup S_{F-M}$, $D_{F-M} = D_{F-M} \cup S_M$, and $D_C = D_C \cup S_{F-M}$. Note that we assign a random class to the anomaly subset that exists in the contaminated dataset. To inject corruptions, we used MNIST-C [34] and CIFAR-10-C [22] that include corrupted versions of MNIST and CIFAR-10. In particular, we selected 10% of the samples and corrupted them using *brightness* and *stripe* corruptions in the training sets generated by [34]. We chose these two corruption types based on their impact on the models' generalization performance [34]. Finally, for the mixed glitch case, where D_{tr} contains both mislabeled samples and anomalies, we set the glitch ratio to 20%, distributed evenly between mislabeled and anomalies.

ML Models. For the image datasets, we employ powerful and popular foundation models for computer vision tasks with diverse architectures such as ResNet-20 [20] pre-trained on CIFAR-10, ConvNeXt [32] pre-trained on ImageNet and Vision Transformer (ViT-16B) [9] pre-trained on JFT-300M. For the tabular datasets, we employed FT-Transformer, ResNet and MLP [14]. Each model was

[5] Stratification preserves the initial class distribution

then trained for a few fine-tuning steps on each contaminated training set, minimizing the cross-entropy loss and achieving a good validation accuracy. The learning rate, number of epochs, batch size, and validation accuracy for each model can be found in our code repository. Each model was fine-tuned using stochastic gradient descent with no momentum to meet TracIn's assumptions [43]. Note that each model was trained in each dataset, resulting in 9 combinations in total. Regarding the ResNet-20 - CIFAR-10 combination, the rationale is to continue the training procedure, to increase the predictive performance further while injecting data glitches into the training set. This simulates the scenario of fresh incoming samples with data quality issues that become part of the training set. The goal is to detect those glitches.

IF. As described in Section 2, we employ TracIn [43] using the Captum [6] package implemented in PyTorch. To speed up computations, the influence is estimated based only on the weights of the last layer of each model, which does not degrade influence estimation [1, 38, 43].

Performance metric. To compare the detection efficacy of the proposed influence signals and the baselines, the F1-score is employed. Intuitively, the F1-score is higher when a corresponding method precisely retrieves most mislabeled or anomalous samples. As the F1-score is a binary metric, to binarize the scores returned by the influence signals and the employed glitch detectors, we considered the true mislabeled and anomaly ratios as thresholds [36].

Influence-based Signals. We compare the detection performance of the three new with existing influence signals such as Self Influence (SI), Average Absolute Influence (AAI), GD-class, and Marginal Influence (MI) (see Section 7 for the exact definitions).

Glitch Detectors. For the mislabeled detection, we compare our signals with CleanLab [39], a *model-based* mislabeled detector with label repair capabilities[7]. Regarding the anomalies, we compare against popular *model-agnostic* detectors such as Deep-SVDD [44] DIF [55], Isolation Forest (iForest) [30], and CBLOF [21], all implemented by PyOD [60]. Given that data glitch detection is an unsupervised problem, i.e., the error labels are not known in advance, the default hyper-parameters as proposed in the respective works are employed. Note that CNCI and PCID do not require tuning, i.e., they do not have any hyper-parameters.

6 Experimental Evaluation

In this section, we report the results of the experimental comparison of the 3 proposed influence-based signals CNCI, PCID, and CFRank with existing signals and detectors. In particular, we are interested in detecting mislabeled and anomalous samples *during model training* by examining their influence footprint on the ML models. Note that each experiment was repeated five times with different random seeds. Due to space constraints, the pipeline schema along with its description is illustrated in Section A.1 in Appendix.

[6]https://captum.ai/api/index.html
[7]https://github.com/cleanlab/cleanlab

6.1 Mislabeled Samples

In this experiment, we are interested in assessing the detection performance of the influence-based signals for uniform and class-based label noise. In our experiments, we have observed a similar trend between uniform and class-based noise in terms of detection performance (see Fig. 1a and Fig. 7 in the Appendix). Therefore, we explore a more challenging case by contaminating the 10% of a *single* class instead of all classes. Now, the mislabeled data correspond to the 1% of the dataset. We call this case *singular mislabeled class*.

6.1.1 Influence-Based Signals Comparison. As illustrated in Fig. 1, CNCI outperforms the existing influence signals in terms of detection performance for both image and tabular data modalities. CNCI exhibits robust performance across the different ML models irrespective of the class noise type (uniform or class-based noise). Specifically, for the singular mislabeled class in Fig. 1d, CNCI outperforms the state-of-the-art signal for label noise, namely SI, by a margin of 65% and up to a factor of 2 the signals GD-class and AAI on average. Moreover, we show that CNCI is very effective in the high-resolution ImageNet-Dogs in Fig. 9 (Appendix). The detection performance difference is higher for the tabular datasets in both label noise types, where CNCI outperforms SI by a factor of 2.3 on average. When the models are trained from scratch, the parameter updates are expected to diverge more than when fine-tuning a foundation model, especially in the earlier stage of the training [45], confusing the existing signals into attributing greater influence to clean samples. CNCI is confused less by this phenomenon. This may attributed to the last influence step when computing the counterfactual influence using the final model parameters, i.e., when the model is at a convergence state. The execution time of CNCI is depicted in Table 1. The signal takes approximately a fraction of 9 w.r.t. the total model training time without using a GPU. Apart from the debugging objective, employing influence-based signals offers the additional advantage of instance-based explanations for every training example.

6.1.2 CNCI vs CleanLab. In Fig. 2a and 2b, we compare for tabular and image modalities the detection performance of CNCI with the model-based mislabeled detector, CleanLab [39] which also provides label repairs on the detected mislabeled samples. We present the results when only one class contains mislabeled samples, which is a more realistic noise distribution in real applications. The uniform noise detection and repair comparison is depicted in 8 in the Appendix. CNCI achieves a 50% higher F1-score for tabular and 70% for image datasets on average. CleanLab returns more false positives, i.e., clean samples identified as mislabeled. Since both CNCI and CleanLab are model-based detectors, their F1-Score is expected to decrease with model performance degradation, which is the case for ViT-16B and ConvNeXt. In Section 4 we argued that a counterfactual label will enable CNCI not only to detect but to repair a problematic label. This motivates us to compare the accuracy of the counterfactual labels that serve as label corrections, with CleanLab. In Fig. 2c and 2d, the label cleaning performance of CNCI is on par with CleanLab, achieving 85% label correction accuracy on average for both data modalities. Interestingly enough, even in the presence of 10 classes on image datasets, CNCI suggests the correct label with approximately 75% accuracy on average.

This verifies that the counterfactual labels obtained by CNCI were indeed the true labels of the mislabeled samples. Note that in the label correction experiment, we evaluated both repair mechanisms, CNCI and CleanLab, under the assumption that all the mislabeled samples were detected, i.e., focusing exclusively on how accurate label repairs they provide.

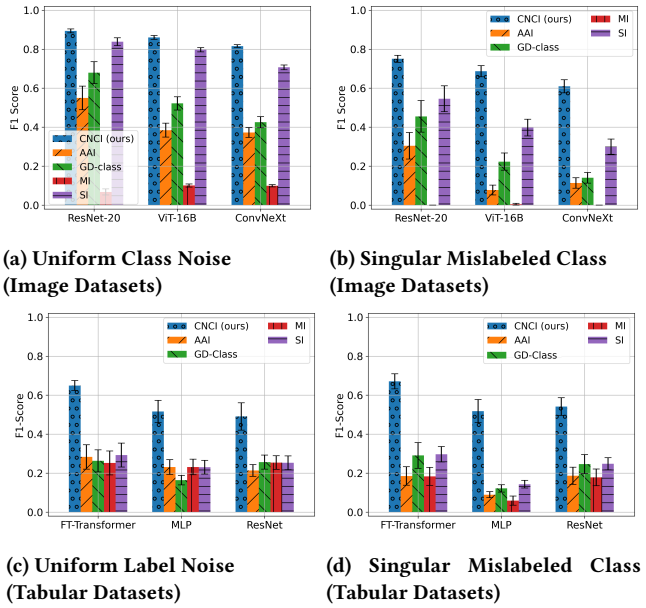
6.1.3 CNCI Robustness on Different Validation Set Characteristics and Noise Ratios. Although a common assumption is that the validation set is clean [59], this may not hold in practice. For instance, obtaining a clean validation set may be challenging; thus, it is important to assess that influence signals can still work using a either a very-small-and-clean or an imperfect validation set. Fig. 3a shows various validation set sizes w.r.t. the training set ranging from 2% up to 20%. We report the steepest performance difference from 2% to 10% validation size among all dataset and model combinations. For every dataset and model combination, the performance difference after the 10% validation size was not substantial. The experiment is repeated for 100 bootstraps for every ratio, and the average detection performance is depicted. The detection performance of CNCI when the validation set is 2% is not substantially than the performance on 10%. In a real-world scenario where the model is trained and deployed rapidly, the existence of a clean validation set may be unrealistic. For this reason, we evaluated CNCI on imperfect validation sets containing 20% class-dependent label noise. In Fig 3b, CNCI is not affected substantially by the existence of noisy labels in the validation set showing almost no performance difference.

Finally, CNCI is more robust than the rest of the signals in low label noise ratios such as 1%, as depicted in Fig. 11 in the Appendix.

Lessons Learned. The mislabeled samples negatively influence the model's decision boundary, especially the samples of their true unknown class (the *ncf* class). Computing the counterfactual influence lets CNCI outperform the existing influence signals, especially on the singular mislabeled class case, for both data modalities while being robust across the different ML models, validation set size (2%), and varying label noise ratios. Moreover, CNCI is on par and sometimes outperforms CleanLab while providing accurate label repairs. Finally, we should stress that the performance of all influence signals and model-based detectors such as CleanLab are heavily impacted by the model's performance.

6.2 Anomalous Samples

The influence signature between an anomalous and a mislabeled sample during model training is fundamentally different. In contrast to mislabelled, anomalies often lie in sparse, isolated areas. We observed that each foundation model predicts them with very high accuracy and achieves lower loss on anomalies than on clean samples (see Table 3 in the Appendix). This explains why in Fig.4a the proposed signal PCID is the only one that can detect Far-Clustered anomalies, outperforming all other signals up to a factor of 6 for all foundation models. Anomalies proved to be easy-to-fit samples with small conditional positive influence. Moreover, we show that PCID is very effective in the high-resolution ImageNet-Dogs in Fig. 9 (Appendix). Interestingly enough, as the Far-CA ratio increases, the detection quality increases for every foundation model and dataset combination, see Fig. 12 (Appendix). Finally, even with a 2% validation set size, PCID can still accurately spot the Far-CA



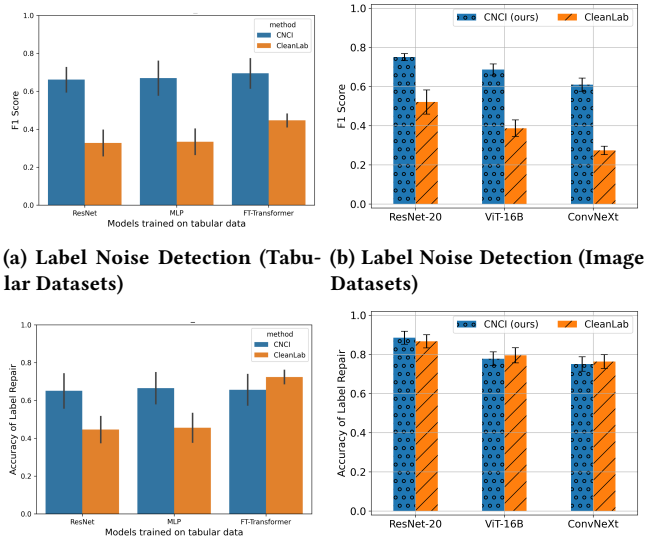
(a) Uniform Class Noise (Image Datasets)

(b) Singular Mislabeled Class (Image Datasets)

(c) Uniform Label Noise (Tabular Datasets)

(d) Singular Mislabeled Class (Tabular Datasets)

Figure 1: Detection performance comparison of CNCI with existing signals for 10% class noise. The results are averaged by ML model on the three image (a,b) and tabular (c,d) datasets. In image data, the foundation models are fine-tuned. In tabular data, the DNNs are trained from scratch.



(a) Label Noise Detection (Tabular Datasets)

(b) Label Noise Detection (Image Datasets)

(c) Label Noise Repair (Tabular Datasets)

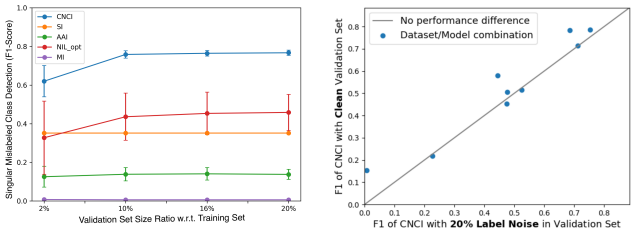
(d) Label Noise Repair (Image Datasets)

Figure 2: CNCI and CleanLab detecting (a,b) and repairing (c,d) 10% samples of Singular Mislabelled Class, averaged by ML model for tabular and image datasets.

samples. The execution time of PCID is depicted in Table 1. PCID takes approximately a fraction of 10 w.r.t. the total model training time. The execution time can be further reduced if a GPU is used.

As depicted in Fig. 4b, PCID's detection performance is on par with dedicated anomaly detectors. PCID seems to be the most robust method across different datasets, which is not the case for the employed anomaly detectors. We perform the experiments on the outlier samples injected via corruptions, depicted in Fig. **??** in the Appendix. Despite the fact that PCID outperforms the existing signals for detecting outliers while being the most robust across the different datasets, Isolation Forest significantly outperforms PCID in the MNIST dataset for the two corruption types. Note that as explained in Section 3, the anomaly detection experiments are performed only for the image datasets; to the best of our knowledge, there is no straightforward method to inject far-clustered anomalies in tabular data when solving a classification problem, instead of unsupervised anomaly detection.

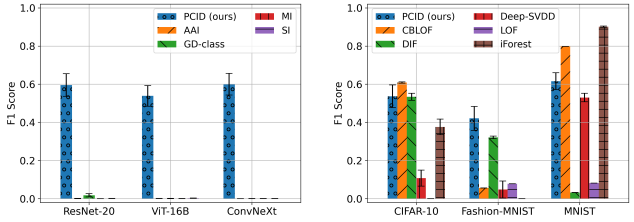
Lessons Learned. Anomalies are easy-to-fit samples for each foundation model and have a very small positive impact on all classes compared to the clean samples. This particular characteristic is not captured by existing signals that rely on influence aggregations of high magnitude to identify an anomalous sample. In contrast, PCID exhibits a high detection performance and it is more robust than anomaly detectors.



(a) Steepest detection performance difference of all model/dataset combinations for different validation set sizes.

(b) Detection performance on clean and imperfect val. set. The diagonal indicates no performance difference.

Figure 3: CNCI robustness on detecting class-dependent mislabeled samples for different validation set characteristics.



(a) PCID vs influence-based signals. All baselines achieve F1 lower than 0.1

(b) PCID vs anomaly detectors. PCID is the most robust method across the three datasets

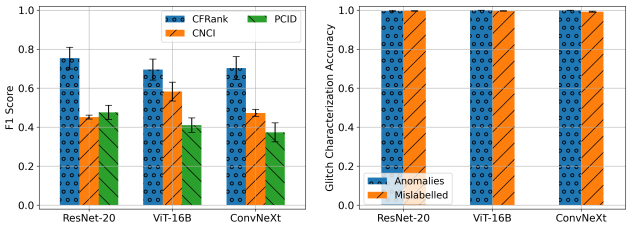
Figure 4: Detection performance comparison of 10% Far-CA. The results are averaged by the foundation model on the three image datasets.

Model	#Parameters	Model Training	CNCI	PCID
ResNet-20	270K	600	50	60
ConvNeXT	27M	2,000	300	320
ViT	85M	4,900	500	580

Table 1: Execution time for models training on 6K samples and calculating the proposed influence signals in seconds (no GPU or parallelization used). Both signals require a fraction, approximately 10%, of the total training time.

6.3 Glitch Mixtures

In the previous experiments, the training sets were contaminated with single-type glitches, either mislabeled or anomalous samples. In this experiment, we are interested in discovering arbitrary mixtures of glitches in the same training set using the CFRank signal and assessing its added value over the CNCI and PCID that target single glitches. As stated in Section 4.1, the glitch mixtures considered in our experiments concern glitched samples that are either mislabeled or anomalous, rather than samples with multiple errors such as feature-dependent class noise. In our experiments, we contaminated the three image datasets with 10% mislabeled and 10% anomalies. In Fig. 5a, we observe that CFRank achieves a 50% performance improvement in detecting both types of glitches by combining the two signals. Apart from discovering glitch mixtures, CFRank is also able to characterize their type precisely. In Fig. 5b, we observe that CFRank accomplishes this additional task with high precision by examining the individual PCID and CNCI values. Anomalous and mislabeled samples have consistently higher PCID/CNCI ranks than clean samples.



(a) Detection

(b) Characterization

Figure 5: Performance of CFRank for detecting (a) and characterizing (b) mixed glitches, i.e., training sets with both mislabeled and anomalies, averaged by foundation model.

7 Related Work

In this section, we survey prior works on IFs and their application in the detection of data glitches. IFs were first introduced for regression tasks [7, 17] and they recently served as model diagnostic tools in classification tasks [24, 25]. Apart from explaining the model's predictions, IFs provide valuable information for detecting various data glitches during training [19, 24, 27, 38, 43, 51] or testing [52]. Table 2 summarizes the main characteristics of existing influence-based signals for detecting mislabeled samples or anomalies during training and positions the new signals proposed in this work. Influence signals [52] for detecting *test* errors such as out-of-distribution, open-set recognition, adversarial attacks [6] or data poisoning [47]

Influence Signal	IF	Sign	Magnitude	Conditional	Mislabeled samples	Anomalies	Mixed	Repair
SI [24, 27, 43]	[24, 27, 43]	-	●	-	●	●	○	○
MI [5, 26]	[5, 24]	○	●	○	●	○	○	○
AAI [19]	[19]	○	●	○	○	●	○	-
GD-class [38]	[43]	○	●	●	●	○	○	○
CNCI (ours)	[43]	●	●	●	●	○	○	●
PCID (ours)	[43]	●	●	●	○	●	○	-
CFRank (ours)	[43]	●	●	●	●	●	●	●

Table 2: Influence-Based signals for detecting mislabeled samples, anomalies, or mixed data glitches in *training* sets. IF denotes the influence estimator where the signal was used, the sign, magnitude, and conditional columns concern characteristics of the influence aggregations made by a signal as described in Section 4. The last column shows if a signal can provide repairs to the detected errors. The "-" denotes that the characteristic can not be defined for that signal, e.g., SI measures the influence of a sample on itself, thus sign and conditional aggregations are undefined; for anomalies, there is no straightforward repair.

are outside the scope of this work. We should stress that our signals are orthogonal to the choice of the IF, assuming that the IF provides accurate influence estimates. In this respect, we rely on dynamic IFs [43], omitting Shappely-Values-based methods [13, 23] as well as Representer-Point-based methods [50, 57].

The seminal signal for detecting mislabeled and anomalous samples is *Self Influence (SI)* [24, 27, 43]. Considering a sample z_i and an IF $\mathcal{I}$, SI is defined as $\mathcal{I}(z_i \rightarrow z_i)$. Samples with high SI magnitude are considered "influential" and are more prone to have errors in labels [24, 43] or features [27]. Although previous studies report that SI can also detect anomalies in generative models [27], our experiments do not confirm such results for classification tasks. Indeed, the efficacy of the proposed signal PCID stems from the fact that anomalies do not have a substantial positive influence on any of the employed foundation models.

Marginal Influence (MI) [5, 26] is a joint influence signal defined as $I(z_i \rightarrow S) = \sum_{j=1}^{m} I(z_i \rightarrow z_j)$, where $z_j \in S$, capturing the marginal influence of z_i to a sample set S to detect mislabeled samples. MI assumes that these samples have large influence values, without considering sign or class information in the influence aggregation. This fundamental distinction sets CNCI apart, leading to a substantial performance advantage over MI across all ML models. *Average Absolute Influence* (AAI) [19] is a joint influence signal for detecting anomalies, defined as $I(z_i \rightarrow S) = \sum_{j=1}^{m} \frac{1}{m} |\mathcal{I}(z_i \rightarrow z_j)|$, where $z_j \in S$. This signal prioritizes anomalies with high AAI scores. Similarly to MI, it does not consider sign or class information, leading to lower performance than our PCID.

Finally, *GD-class* [38] is a joint conditional signal defined as $\mathcal{I}(z_j \rightarrow S) = min_{y \in Y} \sum_{j=1}^{m} \mathcal{I}(z_i \rightarrow z_j|c)$, where $z_j \in S$. It assumes that these samples have a very large negative influence on a specific class. The authors state that the most negative class of a mislabeled sample is its true class; however, they use GD-class only for detection and not for label correction. We find that this signal is not effective especially for a low class noise ratio, because some hard (but clean) samples. CNCI deals with these samples by performing a single influence step on the counterfactual label.

Unlike the existing joint influence-based signals, the three proposed signals are the only ones that account for both the influence sign and class information (see Table 2). This is crucial to enhance the interpretability of our signals: CNCI and PCID work with negative and positive influencers rather than raw aggregations. Moreover, existing signals target exclusively single types of glitches. In contrast, CFRank has been introduced to detect mixtures of data glitches, proposing at the same time label corrections using CNCI. Last but not least, CFRank strives to characterize the type of a detected glitch which is practically useful in datasets with unknown data imperfections.

8 Summary

In this work, we introduced an influence-based framework to detect and characterize arbitrary mixtures of data glitches during model training using distinct influence signatures. We proposed three novel influence-based signals for mislabeled and anomalous samples in training sets that generalize well across different ML models and datasets outperforming existing signals and in some cases data glitch detectors. The proposed signals proved robust across various datasets, varying validation set sizes, and imperfect validation sets. More importantly, we demonstrated that instance-based explanations can serve as accurate detectors and introduced a new family of signals called counterfactuals that provide accurate repairs for label noise. As future work, we are interested in studying the effectiveness of the proposed signals on feature-dependent class noise. Moreover, we plan to study the detection and characterization of data glitches such as out-of-distribution, adversarial examples in test sets, and data poisons. We argue that discovering distinct influence signatures under a common framework using the target ML model for different types of glitches paves the road for *Explainable Data Debugging Frameworks*.

Acknowledgments

This research work has received funding from the Horizon Europe Framework Programme under Grant agreement No 101135775 (PANDORA) and ANR under the Grant agreement 24-CE23-6509 (AIDA).

We also thank Georgios Manos for the preliminary analysis of IFs on linear models, which enhanced our decision to choose TracIn.

References

[1] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1899–1909.

[2] Samyadeep Basu, Phillip Pope, and Soheil Feizi. [n.d.]. Influence Functions in Deep Learning Are Fragile. In *ICLR 2021*.

[3] Christopher M Bishop. 1994. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing* 141, 4 (1994), 217–222.

[4] Chengliang Chai, Jiayi Wang, Yuyu Luo, Zeping Niu, and Guoliang Li. 2022. Data Management for Machine Learning: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–1.

[5] Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. 2021. Hydra: Hypergradient data relevance analysis for interpreting deep neural networks. In *AAAI*, Vol. 35. 7081–7089.

[6] Gilad Cohen, Guillermo Sapiro, and Raja Giryes. 2020. Detecting adversarial samples using influence functions and nearest neighbors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14453–14462.

[7] R Dennis Cook. 1977. Detection of influential observation in linear regression. *Technometrics* 19, 1 (1977), 15–18.

[8] Swagatam Das, Shounak Datta, and Bidyut B Chaudhuri. 2018. Handling data irregularities in classification: Foundations, trends, and future challenges. *Pattern Recognition* 81 (2018), 674–693.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[10] Sebastian Farquhar and Yarin Gal. 2022. What'Out-of-distribution'Is and Is Not. In *NeurIPS ML Safety Workshop*.

[11] Benoit Frenay and Michel Verleysen. 2014. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* 25, 5 (2014), 845–869.

[12] Thomas George, Pierre Nodet, Alexis Bondu, and Vincent Lemaire. 2024. Mislabeled examples detection viewed as probing machine learning models: concepts, survey and extensive benchmark. *arXiv preprint arXiv:2410.15772* (2024).

[13] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *ICML*. PMLR, 2242–2251.

[14] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems* 34 (2021), 18932–18943.

[15] Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. 2021. Data Quality for Machine Learning Tasks. In *KDD*. New York, NY, USA, 4040–4041.

[16] Zayd Hammoudeh and Daniel Lowd. 2022. Training data influence analysis and estimation: A survey. *arXiv preprint arXiv:2212.04612* (2022).

[17] Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association* 69, 346 (1974), 383–393.

[18] Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W Tsang, James T Kwok, and Masashi Sugiyama. 2020. A survey of label-noise representation learning: Past, present and future. *arXiv preprint arXiv:2011.04406* (2020).

[19] Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. Data cleansing for models trained with SGD. *Advances in Neural Information Processing Systems* 32 (2019).

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[21] Zengyou He, Xiaofei Xu, and Shengchun Deng. 2003. Discovering cluster-based local outliers. *Pattern recognition letters* 24, 9-10 (2003), 1641–1650.

[22] Dan Hendrycks and Thomas G Dietterich. 2018. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697* (2018).

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

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

[25] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems* 32 (2019).

[26] Shuming Kong, Yanyan Shen, and Linpeng Huang. 2021. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*.

[27] Zhifeng Kong and Kamalika Chaudhuri. 2021. Understanding instance-based interpretability of variational auto-encoders. *Advances in Neural Information Processing Systems* 34 (2021), 2400–2412.

[28] Peng Li, Xi Rao, Jennifer Blase, Yue Zhang, Xu Chu, and Ce Zhang. 2021. CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks. *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2021), 13–24.

[29] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. 2017. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862* (2017).

[30] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[31] Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624* (2021).

[32] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11976–11986.

[33] Andrea Malossini, Enrico Blanzieri, Raymond T Ng, et al. 2006. Detecting potential labeling errors in microarrays by data perturbation. *BIOINFORMATICS-OXFORD-* 22, 17 (2006), 2114.

[34] Norman Mu and Justin Gilmer. 2019. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337* (2019).

[35] Nikolaos Myrtakis, Ioannis Tsamardinos, and Vassilis Christophides. 2025. Data Glitches Discovery using Influence-based Model Explanations. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, Toronto, ON, Canada.

[36] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. 2021. Machine learning for anomaly detection: A systematic review. *Ieee Access* 9 (2021), 78658–78700.

[37] Felix Neutatz, Binger Chen, Yazan Alkhatib, Jingwen Ye, and Ziawasch Abedjan. 2022. Data Cleaning and AutoML: Would an Optimizer Choose to Clean? *Datenbank-Spektrum* 22 (2022), 121–130.

[38] Thang Nguyen-Duc, Hoang Thanh-Tung, Quan Hung Tran, Dang Huu-Tien, Hieu Nguyen, Anh T. V. Dau, and Nghi Bui. 2023. Class based Influence Functions for Error Detection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 1204–1218.

[39] Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research* 70 (2021), 1373–1411.

[40] Diane Oyen, Michal Kucer, Nicolas Hengartner, and Har Simrat Singh. 2022. Robustness to Label Noise Depends on the Shape of the Noise Distribution. *Advances in Neural Information Processing Systems* 35 (2022), 35645–35656.

[41] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. 2010. Running experiments on amazon mechanical turk. *Judgment and Decision making* 5, 5 (2010), 411–419.

[42] Mykola Pechenizkiy, Alexey Tsymbal, Seppo Puuronen, and Oleksandr Pechenizkiy. 2006. Class noise and supervised learning in medical domains: The effect of feature extraction. In *19th IEEE symposium on computer-based medical systems (CBMS'06)*. IEEE, 708–713.

[43] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems* 33 (2020), 19920–19930.

[44] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep One-Class Classification. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. 4393–4402.

[45] Andrea Schioppa, Katja Filippova, Ivan Titov, and Polina Zablotskaia. 2024. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems* 36 (2024).

[46] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8179–8186.

[47] Sanjay Seetharaman, Shubham Malaviya, Rosni Vasu, Manish Shukla, and Sachin Lodha. 2022. Influence based defense against data poisoning attacks in online learning. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 1–6.

[48] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. 2019. Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*. PMLR, 5907–5915.

[49] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2022. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[50] Yi Sui, Ga Wu, and Scott Sanner. 2021. Representer point selection via local jacobian expansion for post-hoc classifier explanation of deep neural networks and ensemble models. *Advances in neural information processing systems* 34 (2021), 23347–23358.

[51] Stefano Teso, Andrea Bontempelli, Fausto Giunchiglia, and Andrea Passerini. 2021. Interactive label cleaning with example-based explanations. *Advances in*

Neural Information Processing Systems 34 (2021), 12966–12977.
[52] Hugo Thimonier, Fabrice Popineau, Arpad Rimmel, Bich-Liên Doan, and Fabrice Daniel. 2022. Tracinad: measuring influence for anomaly detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–6.
[53] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. 2022. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *Comput. Surveys* 55, 8 (2022), 1–35.
[54] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. 2021. Data Collection and Quality Challenges in Deep Learning: A Data-Centric AI Perspective. *CoRR* 2112.06409 (2021).
[55] Hongzuo Xu, Guansong Pang, Yijie Wang, and Yongjun Wang. 2023. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[56] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, et al. 2022. Openood: Benchmarking generalized out-of-distribution detection. *Advances in Neural Information Processing Systems* 35 (2022), 32598–32611.
[57] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. 2018. Representer point selection for explaining deep neural networks. *NeurIPS* 31 (2018).
[58] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* 64, 3 (2021), 107–115.
[59] Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. 2018. Training set debugging using trusted items. In *AAAI*, Vol. 32.
[60] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *J. Mach. Learn. Res.* 20 (2019), 96:1–96:7.

A Appendix

A.1 Pipeline

The influence-based detection pipeline is summarized as follows: (i) a ML model is trained directly to the given training dataset that contains data glitches, (ii) the model's parameters are stored across all epochs (iii) the Influence Matrix is obtained using the TracIn, (iv) the proposed influence-based signals to detect potential data glitches are computed, and (v) repair actions are applied. Then one can retrain the model or continue the training after repairing the glitches for a few epochs. Regarding (v) the repair action for the mislabeled samples is to discover their unknown correct class using the CNCI signal. As for the anomalous samples considered in our work, there is no label error, they correspond to samples of either new concepts (far-isolated clusters) or outliers injected into the training dataset; thus, none of the classes or straightforward repairs are suitable for them. For the far-isolated clusters, a possible repair requires augmenting the neighborhood area with additional samples and assigning a new class to them, letting the classifier better learn their distinctive properties. Note that the repair of the studied anomalies is beyond the scope of this work.

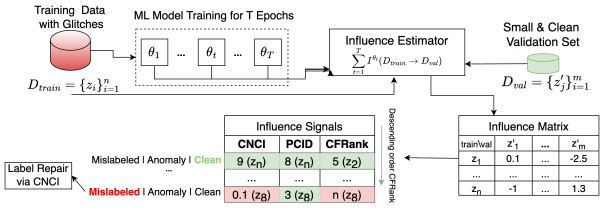
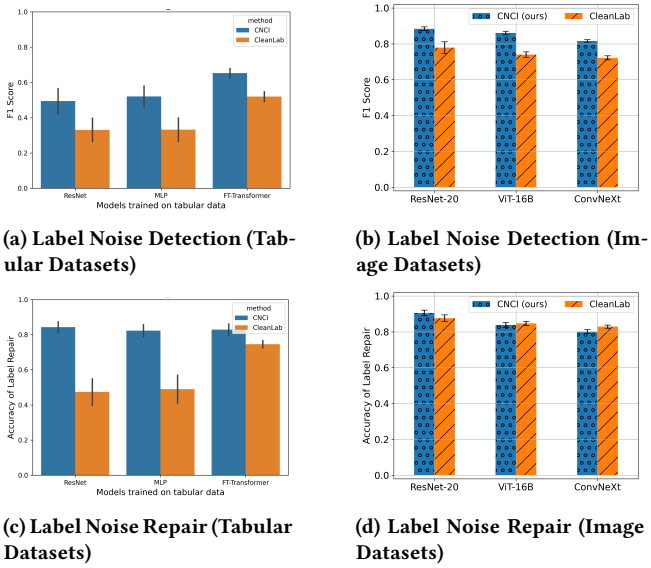


Figure 6: Influence-based data glitch detection, characterization, and repair pipeline.



(a) Label Noise Detection (Tabular Datasets)

(b) Label Noise Detection (Image Datasets)

(c) Label Noise Repair (Tabular Datasets)

(d) Label Noise Repair (Image Datasets)

Figure 8: CNCI and CleanLab detecting (a,b) and repairing (c,d) samples of Uniform Class Noise, averaged by ML model for tabular and image datasets. CNCI outperforms CleanLab for both tasks.

A.2 Additional and Supporting Experiments

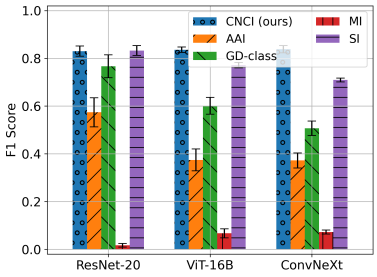
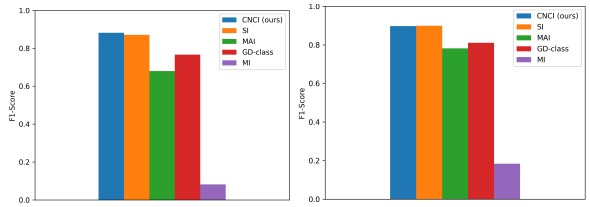


Figure 7: Detection of 10% class-based noise where all classes are contaminated. The experiment ran for 5 random seeds and the results were averaged by the foundation model on the three image datasets.

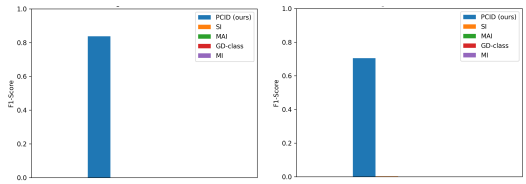
A.2.1 Experiment on a high-resolution image dataset. We have employed a challenging subset of ImageNet that contains 10 different dog breeds, taken from FastAI[8] to show the robustness of the proposed signals in a higher resolution dataset. This dataset contains 320x333 high-quality images. We injected 10% mislabeled samples in the train set (Fig. 9) and 10% Far-CA (Fig. 10) and ran the influence-based signals to measure the detection performance. Note that only ViT and ConvNeXt models were able to learn this dataset; we omitted to compute the signals for ResNet-20 due to very low accuracy. CNCI is on par with SI and outperforms the rest of influence-based

[8] https://github.com/fastai/imagenette?tab=readme-ov-file



(a) ConvNeXt (b) Vision Transformer

Figure 9: Performance of CNCI vs existing signals for detecting 10% uniform label noise on the high resolution dataset ImageNet-Dogs.



(a) ConvNeXt (b) Vision Transformer

Figure 10: Performance of PCID vs existing signals for detecting 10% Far-CA samples injected from Fashion-MNIST to ImageNet-Dogs.

signals. In comparison to SI, CNCI is also able to provide accurate label repairs while being more robust for a lower noise ratio.

A.2.2 Signals Detection Performance w.r.t. Glitch Ratio. In this experiment, we increase the glitch ratio starting from 1% up to 30%. For the mislabeled samples in Fig. 11, the CNCI preserves its detection accuracy for small and higher noise ratios, outperforming SI, especially for low noise ratios. Regarding the anomalies in Fig. 12, PCID achieves a better detection accuracy for every model/dataset combination as the Far-CA ratio increases. This may be attributed to the fact that the foundation models fit the anomalies better when they form bigger isolated clusters, resulting in a smaller overall positive influence.

A.2.3 Detection of Outlier Samples. In this experiment, we are interested in detecting outlier samples in the form of corruptions. As outliers, we consider the samples (images) that have been corrupted under a common corruption mechanism. We employ two popular corrupted versions of MNIST, namely MNIST-10-c[9], and CIFAR-10, namely CIFAR-10-c[10]. We corrupted the 10% of the training samples using "Brightness" and "Stripe" corruptions using the code[11] of the paper [34]. We chose these two corruptions as they seem to affect the models' performances as shown in [34]. In Fig. **??**, PCID exhibits the best performance among the rest of the influence signals and outperforms the anomaly detectors for the two corruptions on CIFAR-10. PCID is substantially outperformed by Isolation Forest in MNIST. However, as in the case of Far-CA samples, PCID is the most robust signal on both datasets.

[9]https://zenodo.org/records/3239543
[10]https://zenodo.org/records/2535967
[11]https://github.com/google-research/mnist-c

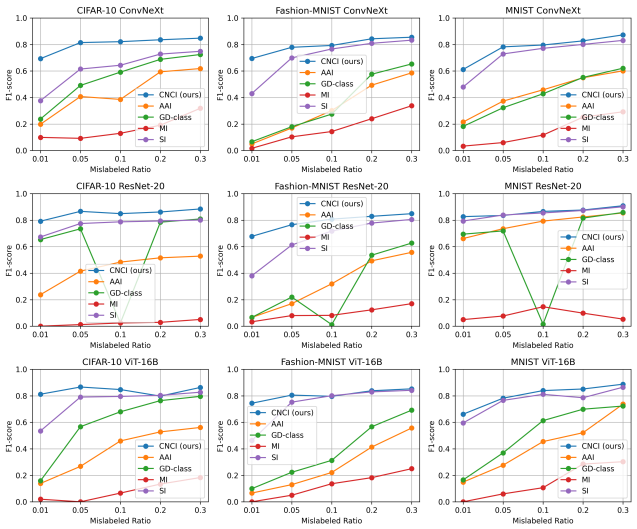


Figure 11: Influence-based signals' detection performance w.r.t. increasing uniform class noise ratio. CNCI exhibits robust performance for increasing noise ratio, outperforming SI on average for lower noise settings.

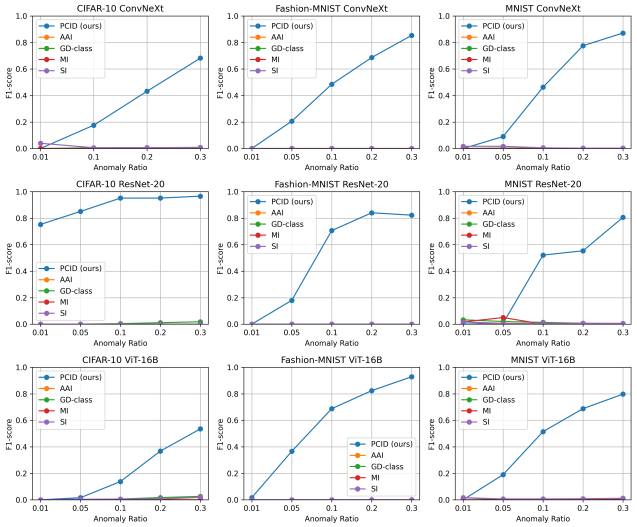


Figure 12: Influence-based signals' average precision w.r.t. increasing Far-CA ratio.

	Clean Samples		Mislabeled		Far-CA	
Dataset	Loss	Acc.	Loss	Acc.	Loss	Acc.
MNIST	1.6	97%	2.35	5%	1.47	100%
F-MNIST	1.67	91%	2.35	9%	1.48	100%
CIFAR-10	1.76	95%	2.38	6%	1.59	100%

Table 3: ResNet-20 average loss and accuracy on clean, mislabeled, and Far-CA samples using 10% glitch ratio over the three vision datasets. ResNet-20 perfectly fits the Far-CA achieving 100% accuracy and a smaller average loss than the clean samples. On the other hand, the model does not learn the mislabeled samples.