CHEF: A Cheap and Fast Pipeline for Iteratively Cleaning Label Uncertainties

Yinjun Wu
University of Pennsylvania
wuyinjun@seas.upenn.edu

James Weimer
University of Pennsylvania
weimerj@seas.upenn.edu

Susan B. Davidson
University of Pennsylvania
susan@cis.upenn.edu

ABSTRACT

High-quality labels are expensive to obtain for many machine learning tasks, such as medical image classification tasks. Therefore, probabilistic (weak) labels produced by weak supervision tools are used to seed a process in which influential samples with weak labels are identified and cleaned by several human annotators to improve the model performance. To lower the overall cost and computational overhead of this process, we propose a solution called CHEF (CHEap and Fast label cleaning), which consists of the following three components. First, to reduce the cost of human annotators, we use INFL, which prioritizes the *most influential* training samples for cleaning and provides cleaned labels to save the cost of one human annotator. Second, to accelerate the sample selector phase and the model constructor phase, we use Increm-INFL to *incrementally* produce influential samples, and DeltaGrad-L to *incrementally* update the model. Third, we redesign the typical label cleaning pipeline so that human annotators iteratively clean smaller batch of samples rather than one big batch of samples. This yields better overall model performance and enables possible early termination when the expected model performance has been achieved. Extensive experiments show that our approach gives good model prediction performance while achieving significant speed-ups.

PVLDB Reference Format:
Yinjun Wu, James Weimer, and Susan B. Davidson. CHEF: A Cheap and Fast Pipeline for Iteratively Cleaning Label Uncertainties. PVLDB, 14(11): 2410 - 2418, 2021.
doi:10.14778/3476249.3476290

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/thuwuyinjun/Chef.

1 INTRODUCTION

There is a general consensus that the success of advanced machine learning models depends on the availability of extremely large training sets with high-quality labels. Unfortunately, obtaining high-quality labels may be prohibitively expensive. For example, labeling medical images typically requires the effort of experts with domain knowledge. To produce labels at large scale with low cost, weak supervision tools—such as Snorkel [31]—can be used to automatically generate *probabilistic labels* (or *weak labels*) for unlabeled training samples by leveraging labeling functions [31].

Proceedings of the VLDB Endowment, Vol. 14, No. 11 ISSN 2150-8097.
doi:10.14778/3476249.3476290

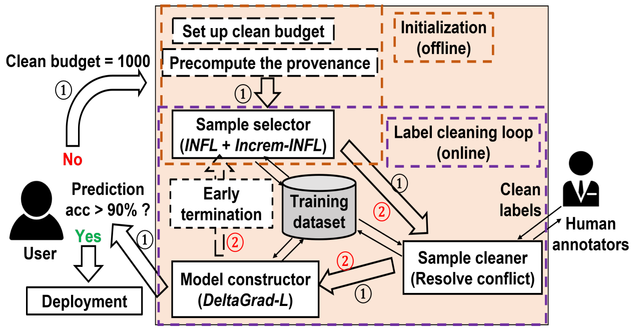

Figure 1: The iterative pipeline of cleaning uncertainties from the labels of training set.

It has been shown in [2, 27, 34], however, that imperfect labeling functions can produce inferior probabilistic labels, thus hurting the downstream model quality. Therefore, it is necessary to perform additional *cleaning operations* to clean such label uncertainties [27].

The label cleaning process is typically *iterative* [21, 24], and requires multiple rounds (see Figure 1, loop labeled ①). First, given a *cleaning budget* B, the top-B influential training samples with probabilistic labels are selected (the *sample selector phase*). Second, for those selected samples, cleaned labels are provided by human annotators (the *annotation phase*). Third, the ML model is calculated using the updated training set (the *model constructor phase*), and returned to the user. If the resulting model performance is not good enough, the process is repeated with an additional budget B'. Otherwise, it is deployed. Note that since each of these phases may be performed *repeatedly*, it is important that they be as efficient as possible. It is also noteworthy that for some applications—such as the medical image classification task—it is essential to have multiple human annotators for label cleaning to alleviate their labeling errors [17] in the *annotation phase*, thus incurring substantial time overhead and financial cost. **In this paper, we propose a solution called CHEF (CHEap and Fast label cleaning), to reduce the time overhead and cost of the label cleaning pipeline and simultaneously enhance the overall model performance.** Details of the overall design of CHEF are given next.

Sample selector phase. Finding the most influential training samples can be done with several different influence measures, e.g., the influence function [20], the Data Shapley values [18], the noisy label detection algorithms [9, 16], the active learning technique [33] or using a bi-level optimization solution [41]. Unfortunately, these do not work well for cleaning weak labels. We therefore develop a variant of the influence function called INFL which can simultaneously detect the most influential samples and suggest cleaned labels. One key technical challenge in the efficient implementation of INFL concerns the explicit evaluation of gradients on *every* training sample. **We address this challenge by developing Increm-INFL, which removes uninfluential training samples early and can thus *incrementally* recommend the most influential training samples to human annotators.**

Human annotation phase. After influential samples are selected, the next step is for human annotators to clean the labels of those samples. Recall that *multiple* human annotators may be used to independently label each training sample, and inconsistencies between the labels are resolved, e.g., by majority vote [17]. **To reduce the cost of the human annotation phase, we consider the suggested clean labels from the *sample selector phase* as one alternative labeler, which can be combined with results provided by the human annotators to reduce annotation cost.**

Model constructor phase. In previous work [38], we developed a provenance-based algorithm called DeltaGrad for *incrementally updating model parameters* after the deletion or addition of a small subset of training samples, and showed that it was significantly faster than recalculating the model from scratch. Since the result of the human annotation phase can be regarded as the deletion of top-B samples with probabilistic labels, and insertion of those same samples with cleaned labels, we can adapt DeltaGrad for this setting. This algorithm is called DeltaGrad-L. **To accelerate the model constructor phase, rather than retraining from scratch after cleaning the labels of a small set of training samples, we *incrementally* update the model using DeltaGrad-L.**

Redesign of the cleaning pipeline. The final contribution of this paper, which is enabled by the reduced cost of the sample selection, human annotation, and model construction phases, is a re-design of the pipeline in Figure 1 (see the loop $\textcircled{2}$). Rather than providing all top-B influential training samples (and suggesting how to fix the label uncertainty) at once, the sample selector gives the human annotator the next top-b influential training samples, where b is smaller than B and is specified by the user. The model is then *refreshed* using the cleaned labels, and the next top-b samples to be given to the human annotator are calculated. This continues until the initial budget B has been exhausted or the expected prediction performance is reached (thus terminating early). **This can not only improve the overall model performance, but also lead to early termination, thus further saving the cost of human annotation.** Note that to enable incremental computation by Increm-INFL and DeltaGrad-L, some "provenance" information is necessary, and can be pre-computed offline in an *Initialization step* prior to the start of loop $\textcircled{2}$.

We demonstrate the effectiveness of CHEF using several crowd-sourced datasets as well as real medical image datasets. Our experiments show that CHEF achieves up to 54.7x speed-up in the sample selector phase, and up to 7.5x speed-up in the model constructor phase. Furthermore, by using INFL and smaller batch sizes b, the overall model quality can be improved.

Summarizing, the contributions of this paper include:

- A solution called CHEF which can significantly reduce the overall cost of label cleaning by 1) reducing the cost of the Sample selector phase, the Human annotation phase and the Model constructor phase respectively and 2) redesigning the label cleaning pipeline to enable better model performance and early stopping in the human annotation phase.
- Extensive experiments which show the effectiveness of CHEF on real crowd-sourced datasets and medical image datasets.

The rest of this paper is organized as follows. In Section 2, we summarize related work. Preliminary notation, definitions and assumptions are given in Section 3, followed by our algorithms, INFL, Increm-INFL and DeltaGrad-L in Section 4. Experimental results are discussed in Section 5, and we conclude in Section 6.

2 RELATED WORK

Incremental updates on ML models In the past few years, several approaches for incrementally maintaining different types of models have emerged [5, 11, 20, 38, 39], which address important practical problems such as GPDR [32] and training sample valuation [10]. The DeltaGrad-L algorithm in the model constructor phase is adapted from our DeltaGrad algorithm [38], which addresses the problem of incrementally updating strongly convex models after a small subset of training samples are deleted or added. Note that this problem is related to the classical *materialized view maintenance problem* as mentioned in [39], if we consider ML models as *views*.

Data cleaning for ML models Diagnosing and cleaning errors or noise in training samples has attracted considerable attention [9, 16], and is typically addressed iteratively [1, 21, 24]. For example, the authors of [16] observed that the noisily labeled samples were memorized by the model in the overfitting phase, which can be detected through transferring the model status back to the underfitting phase. [9] identifies and fixes the noisy labels by jointly analyzing the probability that one noisy label is flipped by the human annotators and how this label update influences the model performance. However, it explicitly assumes that the noisy labels are either 1 or 0, and is therefore not applicable for probabilistic labels. The approach in [21] detects errors in both feature values and labels. However, it explicitly assumes that the uncleaned samples are harmful and thus excluded in the training process. In contrast, we follow the principle of [31] by "including" the training samples with uncertain labels in the training phase.

Detecting the most influential training samples with uncertainties As discussed in [1], it is important to prioritize the most influential training samples for cleaning. This can depend on various influence measures, e.g., the uncertainty-based measures in active learning [33], the influence function [20], the data Shapley value [18], the training loss [14, 16], etc. However, to our knowledge, none of these techniques can be used to automatically suggest possibly cleaned labels, apart from [41]. The applicability of [41] is also limited due to its poor scalability, and some of the above methods (including [41]) are not applicable in the presence of probabilistic labels and their regularization.

3 PRELIMINARIES

In this section, we introduce essential notation and assumptions, and then describe the influence function and DeltaGrad.

3.1 Notation

A C-class classification task is a classification task in which the number of classes is C. Suppose that the goal is to construct a machine learning model on a training set, $\mathcal{Z} = \mathcal{Z}_d \bigcup \mathcal{Z}_p$, in which $\mathcal{Z}_d = \{\mathbf{z}_i\}_{i=1}^{N_d} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_d}$ and $\mathcal{Z}_p = \{\tilde{\mathbf{z}}_i\}_{i=1}^{N_p} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i=1}^{N_p}$, denoting a set of N_d training samples with deterministic labels and N_p training samples with probabilistic labels, respectively. A *probabilistic label*, $\tilde{y}_i$, is represented by a probabilistic vector of length C, in which the value in the c_{th} entry ($c = 1, 2, \dots, C$) denotes the probability that $\tilde{\mathbf{z}}_i$ belongs to the class c. The performance of the model constructed on $\mathcal{Z}$ is then validated on a validation dataset $\mathcal{Z}_{\text{val}}$ and tested on a test dataset $\mathcal{Z}_{\text{test}}$. Note that the size of $\mathcal{Z}_{\text{val}}$ and $\mathcal{Z}_{\text{test}}$ are typically small, consisting of samples with ground-truth labels or deterministic labels verified by the human annotators. Due to the possibly negative effect brought by the uncleaned training samples with probabilistic labels, it is reasonable to regularize those samples in the following objective function (e.g. see [35]):

$$F(\mathbf{w}) = \frac{1}{N}\left[\sum_{i=1}^{N_d} F(\mathbf{w}, \mathbf{z}_i) + \sum_{i=1}^{N_p} \gamma F(\mathbf{w}, \tilde{\mathbf{z}}_i)\right] \tag{1}$$

In the formula above, we use $\mathbf{w}$ to represent the model parameter, $F(\mathbf{w}, \mathbf{z})$ to denote the loss incurred on a sample $\mathbf{z}$ with the model parameter $\mathbf{w}$ and γ ($0 < \gamma < 1$, specified by users) to denote the weight on the uncleaned training samples. Furthermore, the first order gradient of this loss can be denoted by $\nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z})$, and the second order gradient (i.e. the Hessian matrix) by $\mathbf{H}(\mathbf{w}, \mathbf{z})$. We further use $\nabla_{\mathbf{w}} F(\mathbf{w})$ and $\mathbf{H}(\mathbf{w})$ to denote the first order gradient and the Hessian matrix averaged over all weighted training samples.

To optimize Equation (1), stochastic Gradient Descent (SGD) can be applied. At each SGD iteration t, one essential step is to evaluate the first-order gradients of a randomly sampled mini-batch of training samples, $\mathcal{B}_t$ (we denote the size of $\mathcal{B}_t$ as $|\mathcal{B}_t|$), i.e.:

$$\nabla_{\mathbf{w}} F(\mathbf{w}, \mathcal{B}_t) = \frac{1}{|\mathcal{B}_t|} \sum_{\mathbf{z} \in \mathcal{B}_t} \gamma_{\mathbf{z}} \nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z}),$$

in which γ_z is 1 if $z \in \mathcal{Z}_d$ and γ otherwise.

Plus, since loop $\textcircled{2}$ in Figure 1 may be repeated for multiple rounds, we use $\mathcal{Z}^{(k)}$ to denote the updated training dataset after k rounds and $\mathbf{w}^{(k)}$ to represent the model constructed on $\mathcal{Z}^{(k)}$.

3.2 Assumptions

We make two assumptions: the *strong convexity assumption*, and the *small cleaning budget assumption*.

Strong convexity assumption Following [38], we focus on the models satisfying μ*–strong convexity*, meaning that the minimal eigenvalue of each Hessian matrix $\mathbf{H}(\mathbf{w}, \mathbf{z})$ is always greater than a non-negative constant μ for arbitrary $\mathbf{w}$ and $\mathbf{z}$. One such model is the logistic regression model with L2 regularization.

Small cleaning budget assumption Since manually cleaning labels is time-consuming and expensive, we assume that *the cleaning budget B is far smaller than the size of training set, $\mathcal{Z}$.*

3.3 Influence function

The influence function method [20] is originally proposed to estimate how the prediction performance on one test sample $\mathbf{z}_{\text{test}}$ is varied if we delete one training sample $\mathbf{z}$, or add an *infinitely small* perturbation on the feature of $\mathbf{z}$. This is formulated as follows:

$$\mathcal{I}_{\text{del}}(\mathbf{z}) = -\nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z}_{\text{test}})^\top \mathbf{H}^{-1}(\mathbf{w}) \nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z})$$
$$\mathcal{I}_{\text{pert}}(\mathbf{z}) = -\nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z}_{\text{test}})^\top \mathbf{H}^{-1}(\mathbf{w}) \nabla_{\mathbf{x}} \nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z}).$$

We can then leverage $\mathcal{I}_{\text{del}}(\mathbf{z})$ and $\mathcal{I}_{\text{pert}}(\mathbf{z})\delta$ to approximate the additional errors incurred on the test sample $\mathbf{z}_{\text{test}}$ after deleting the training sample $\mathbf{z}$, or perturbing the feature of $\mathbf{z}$ by δ.

As [20] indicates, by evaluating the training sample influence with the above influence function, the "harmful" training samples on the model prediction (i.e. the one with negative influence) can be distinguished from the "helpful" ones (i.e. the one with positive influence). We can then prioritize the most "harmful" training samples with probabilistic labels for cleaning. In practice, due to the invisibility of the test samples in most cases, the validation set is used instead, leading to the following modified influence functions:

$$\mathcal{I}_{\text{del}}(\mathbf{z}) = -\nabla_{\mathbf{w}} F(\mathbf{w}, \mathcal{Z}_{\text{val}})^\top \mathbf{H}^{-1}(\mathbf{w}) \nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z}) \tag{2}$$
$$\mathcal{I}_{\text{pert}}(\mathbf{z}) = -\nabla_{\mathbf{w}} F(\mathbf{w}, \mathcal{Z}_{\text{val}})^\top \mathbf{H}^{-1}(\mathbf{w}) \nabla_{\mathbf{x}} \nabla_{\mathbf{w}} F(\mathbf{w}, \mathbf{z}) \tag{3}$$

The two formulas above also follow the modified influence function in [41] which uses a set of trusted validation samples instead of test samples to estimate the influence of each training sample.

3.4 DeltaGrad

As introduced in [38], DeltaGrad is used to incrementally update the parameters of a *strongly convex model* after the removal of a small subset of training samples, $\mathcal{R}$ ($|\mathcal{R}| \ll N$), and the addition of another small subset of training samples, $\mathcal{A}$ ($|\mathcal{A}| \ll N$), on the training dataset $\mathcal{Z}$; both $\mathcal{R}$ and $\mathcal{A}$ can be empty. Before the above modifications on the training dataset $\mathcal{Z}$, suppose we derive the gradients on a randomly sampled mini-batch $\mathcal{B}_t$ and calculate the model parameter, $\mathbf{w}_t$, at the t_{th} SGD iteration, Then after $\mathcal{R}$ is deleted and $\mathcal{A}$ is added, to obtain the updated model parameter $\mathbf{w}_t^I$ at the t_{th} SGD iteration, it is essential to evaluate the gradients on the following updated mini-batch $\mathcal{B}_t'$, i.e., $(\mathcal{B}_t - \mathcal{R}) \cup \mathcal{A}_t$. Here, $\mathcal{B}_t - \mathcal{R}$ represents the remaining training samples in $\mathcal{B}_t$ after $\mathcal{R}$ is deleted, while $\mathcal{A}_t$ denotes a randomly sampled mini-batch from $\mathcal{A}$. Note that $\mathcal{B}_t'$ can be further rewritten as $(\mathcal{B}_t - (\mathcal{B}_t \bigcap \mathcal{R})) \cup \mathcal{A}_t$. As a result, the gradient on $\mathcal{B}_t'$ can be evaluated as follows:

$$\begin{aligned}\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t') = \frac{1}{|\mathcal{B}_t'|} \big[&|\mathcal{B}_t| \nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t) \\ &- |\mathcal{B}_t \cap \mathcal{R}| \nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t \cap \mathcal{R}) + |\mathcal{A}_t| \nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{A}_t) \big],\end{aligned} \tag{4}$$

The latter two gradients in the above formula, $\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{A}_t)$ and $\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t \cap \mathcal{R})$, can be efficiently calculated due to the small size of $\mathcal{R}$ and $\mathcal{A}$. As a result, computing $\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t)$ becomes the dominant overhead in evaluating Equation (4) when the mini-batch size is large. Hence, DeltaGrad aims to reduce the overhead of this term by incrementally computing it using the Cauchy-mean value theorem [22] with the approximate Hessian matrix, $\mathbf{B}_t$, as follows:

$$\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t) \approx \mathbf{B}_t(\mathbf{w}_t^I - \mathbf{w}_t) + \nabla_{\mathbf{w}} F(\mathbf{w}_t, \mathcal{B}_t). \tag{5}$$

in which, the product $\mathbf{B}_t(\mathbf{w}_t^I - \mathbf{w}_t)$ is calculated using the L-BFGS algorithm [28] while the gradient term $\nabla_{\mathbf{w}} F(\mathbf{w}_t, \mathcal{B}_t)$ is cached during the training phase on the original training dataset $\mathcal{Z}$.

As described in [38], although this approximation is faster than computing $\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t)$ explicitly, the approximation errors are not negligible. To balance between the approximation error and efficiency in DeltaGrad, $\nabla_{\mathbf{w}} F(\mathbf{w}_t^I, \mathcal{B}_t)$ is explicitly evaluated in the first j_0 SGD iterations and every T_0 SGD iterations afterwards, where T_0 and j_0 are pre-specified hyper-parameters. Note that the evaluation of Equation (5) also requires to cache and reuse the last m_0 explicitly computed gradients, in which m_0 is also a hyper-parameter. In [38], j_0, T_0 and m_0 are referred to as the number of "burn-in" iterations, the period and the history size, respectively. We refer readers to [38] for more details.

4 METHODOLOGY

In this section, we describe the system design in detail for the sample selector phase (Section 4.1), the model constructor phase (Section 4.2) and the human annotation phase (Section 4.3).

4.1 The sample selector phase

Sample selection accomplishes two things: 1) it calculates the training sample influence using INFL in order to prioritize the most influential uncleaned training samples for cleaning, and simultaneously suggests possibly cleaned labels for them (see Section 4.1.1); and 2) it filters out uninfluential training samples early using Increm-INFL at each round of loop ② (see Section 4.1.2).

4.1.1 INFL. The goal of INFL is to calculate the influence of an uncleaned training sample, $\tilde{\mathbf{z}}$, by estimating how much additional error will be incurred on the validation set $\mathcal{Z}_{\text{val}}$ if 1) the probabilistic label of $\tilde{\mathbf{z}}$ is updated to some deterministic label; and 2) $\tilde{\mathbf{z}}$ is up-weighted to 1 after it is cleaned. To capture this intuition, we propose the following modified influence function[1]:

$$\begin{aligned}&\mathcal{I}_{\text{pert}}(\tilde{\mathbf{z}}, \delta_y, \gamma) \approx N \cdot (F(\mathbf{w}^U, \mathcal{Z}_{\text{val}}) - F(\mathbf{w}, \mathcal{Z}_{\text{val}}))\\&= -\nabla_{\mathbf{w}} F(\mathbf{w}, \mathcal{Z}_{\text{val}})^\top \mathbf{H}^{-1}(\mathbf{w}) \left[\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}}) \delta_y + (1-\gamma) \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})\right],\end{aligned} \tag{6}$$

in which δ_y denotes the difference between the original probabilistic label of $\tilde{\mathbf{z}}$ and one deterministic label (ranging from 1 to C) and $\mathbf{w}^U$ denotes the updated model parameters after the label is cleaned and $\tilde{\mathbf{z}}$ is up-weighted. To calculate δ_y, the deterministic label is first converted to its one-hot representation, i.e. a vector of length C taking 1 in the c_{th} entry ($c = 1, 2, \dots, C$) for the label c and taking 0 in all other entries (recall that C represents the number of classes).

To recommend the most influential uncleaned training samples to the human annotators and suggest possibly cleaned labels, we 1) explicitly evaluate Equation (6) for each uncleaned training sample for *all possible deterministic labels*, 2) prioritize the most "harmful" training samples for cleaning, i.e. the ones with the smallest negative influence values after their labels are updated to *some* deterministic labels, and 3) suggest those deterministic labels as the potentially cleaned labels for the human annotators.

Comparison to [41] As discussed earlier, DUTI [41] can also recommend the most influential training samples for cleaning and suggest possibly cleaned labels, which is accomplished through solving a bi-level optimization problem. However, solving this problem is computationally challenging, and therefore this method cannot be used in real-time over multiple rounds (i.e. in loop ②).

Computing $\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$ At first glance, it seems that the term $\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$ cannot be calculated using auto-differentiation packages such as Pytorch, since it involves the partial derivative with respect to the label of $\tilde{\mathbf{z}}$. However, we notice that this partial derivative can be explicitly calculated when the loss function $F(\mathbf{w}, \tilde{\mathbf{z}})$ is the cross-entropy function, which is the most widely used objective function in the classification task. Specifically, the instantiation of the loss function $F(\mathbf{w}, \tilde{\mathbf{z}})$ into the cross-entropy function becomes:

$$F(\mathbf{w}, \tilde{\mathbf{z}}) = -\sum_{k=1}^{C} \tilde{y}^{(k)} \log(p^{(k)}(\mathbf{w}, \tilde{\mathbf{x}})), \tag{7}$$

In this formula above, $\tilde{y} = [\tilde{y}^{(1)}, \tilde{y}^{(2)}, \dots, \tilde{y}^{(C)}]$ is the label of an input sample $\tilde{\mathbf{z}} = (\tilde{\mathbf{x}}, \tilde{y})$ and $[p^{(1)}(\mathbf{w}, \tilde{\mathbf{x}}), p^{(2)}(\mathbf{w}, \tilde{\mathbf{x}}), \dots, p^{(C)}(\mathbf{w}, \tilde{\mathbf{x}})]$ represents the model output given this input sample, which is a probabilistic vector of length C depending on the model parameter $\mathbf{w}$ and the input feature $\tilde{\mathbf{x}}$. Then we can observe that Equation (7) is a linear function of the label $\tilde{y}$. Hence, $\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$ can be explicitly evaluated as:

$$\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}}) = [-\nabla_{\mathbf{w}} \log(p^{(1)}(\mathbf{w}, \tilde{\mathbf{x}})), \dots, -\nabla_{\mathbf{w}} \log(p^{(C)}(\mathbf{w}, \tilde{\mathbf{x}}))] \tag{8}$$

As a result, each $-\nabla_{\mathbf{w}} \log(p^{(c)}(\mathbf{w}, \tilde{\mathbf{x}})), c = 1, 2, \dots, C$ can be calculated with the auto-differentiation package.

Computing $\mathbf{H}^{-1}(\mathbf{w})$ Recall that $\mathbf{H}(\mathbf{w})$ denotes the Hessian matrix averaged on all training samples. Rather than explicitly calculating its inverse, by following [20], we leverage the conjugate gradient method [25] to approximately compute the Matrix-vector product $\nabla_{\mathbf{w}} F(\mathbf{w}, \mathcal{Z}_{\text{val}})^\top \mathbf{H}^{-1}(\mathbf{w})$ in Equation (6).

4.1.2 Increm-INFL. The goal of using INFL is to quantify the influence of all uncleaned training samples and select the Top-b influential training samples for cleaning. But in loop ②, this search space could be reduced by employing Increm-INFL. Specifically, other than the initialization step, we can leverage *Increm-INFL* to prune away most of the uninfluential training samples early in the following rounds, thus only evaluating the influence of a small set of candidate influential training samples in those rounds. Suppose this set of samples is denoted as $\mathcal{Z}_{inf}^{(k)}$ for the round k; the derivation of this set is outlined in Algorithm 1. As this algorithm indicates, the first step is to effectively estimate the maximal perturbations of Equation (6) at the k_{th} cleaning round for each uncleaned training sample $\tilde{\mathbf{z}}$ and each possible label change δ_y (see line 2), which are assumed to take $\mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma)$ (see Theorem 1 for its definition) as the perturbation center. Then the first part of $\mathcal{Z}_{inf}^{(k)}$ consists of all the training samples which produce the Top-b smallest values of $\mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma)$ with a given δ_y (see line 6). For those b smallest values, we also collect the maximal value of their upper bound, L. We then include in $\mathcal{Z}_{inf}^{(k)}$ all the remaining training samples whose lower bound, is smaller than L with certain δ_y (see line 5). This indicates the possibility of those samples becoming the Top-b influential samples.

As described above, it is critical to estimate the maximal perturbation of Equation (6) for each uncleaned training sample, $\tilde{\mathbf{z}}$, and each label perturbation, δ_y, which requires the following theorem.

Theorem 1. *For a training sample $\tilde{\mathbf{z}} = (\tilde{\mathbf{x}}, \tilde{y})$ which has not been cleaned before the k^{th} round of loop ②, the following bounds hold for Equation (6) evaluated on the training sample $\tilde{\mathbf{z}}$ and a label perturbation δ_y:*

$$\left| -\mathcal{I}_{pert}^{(k)}(\tilde{\mathbf{z}}, \delta_y, \gamma) - \mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma) - \frac{1-\gamma}{2} e_1 \mu - \sum_{j=1}^{C} \delta_{y,j} e_1 \|\boldsymbol{H}^{(j)}(\boldsymbol{w}^{(k)}, \tilde{\mathbf{z}})\| \right|$$

$$\leq \sum_{j=1}^{C} |\delta_{y,j}| e_2 \|\boldsymbol{H}^{(j)}(\boldsymbol{w}^{(k)}, \tilde{\mathbf{z}})\| + \frac{1-\gamma}{2} e_2 \mu,$$

[1]The derivation of this formula is similar to that of the original influence function [20], which is included in the extended technical report [40]

in which,
$\mathcal{I}_0(\tilde{\boldsymbol{z}}, \delta_y, \gamma) = \boldsymbol{v}^\top [\nabla_y \nabla_{\boldsymbol{w}} F(\boldsymbol{w}^{(0)}, \tilde{\boldsymbol{z}})\delta_y + (1-\gamma)\nabla_{\boldsymbol{w}} F(\boldsymbol{w}^{(0)}, \tilde{\boldsymbol{z}})]$, $\boldsymbol{v}^\top = -\nabla_{\boldsymbol{w}} F(\boldsymbol{w}^{(k)}, \mathcal{Z}_{val})^\top \boldsymbol{H}^{-1}(\boldsymbol{w}^{(k)})$, $\delta_y = [\delta_{y,1}, \delta_{y,2}, \dots, \delta_{y,C}]$,
$\boldsymbol{H}^{(j)}(\boldsymbol{w}^{(k)}, \tilde{\boldsymbol{z}}) = \int_0^1 -\nabla^2_{\boldsymbol{w}} \log(p^{(j)}(\boldsymbol{w}^{(0)} + s(\boldsymbol{w}^{(k)} - \boldsymbol{w}^{(0)}), \tilde{\boldsymbol{x}}))ds$,
$\mu = \| \int_0^1 \boldsymbol{H}(\boldsymbol{w}^{(0)} + s(\boldsymbol{w}^{(k)} - \boldsymbol{w}^{(0)}), \tilde{\boldsymbol{z}})ds\|$, *and*
$e_1 = \boldsymbol{v}^\top(\boldsymbol{w}^{(k)} - \boldsymbol{w}^{(0)})$, $e_2 = \|\boldsymbol{v}\|\|\boldsymbol{w}^{(k)} - \boldsymbol{w}^{(0)}\|$.

To reduce the time overhead, the integrated Hessian matrices, $\int_0^1 \mathbf{H}(\mathbf{w}^{(0)} + s(\mathbf{w}^{(k)} - \mathbf{w}^{(0)}), \tilde{\mathbf{z}})ds$ and $\mathbf{H}^{(j)}(\mathbf{w}^{(k)}, \tilde{\mathbf{z}})$, are approximated by their counterparts evaluated at $\mathbf{w}^{(0)}$, i.e., $\mathbf{H}(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})$ and $-\nabla^2_{\mathbf{w}} \log(p^{(j)}(\mathbf{w}^{(0)}, \tilde{\mathbf{x}}))$. As a consequence, the bounds can be calculated by applying several linear algebraic operations on $\mathbf{v}$, $\mathbf{w}^{(k)}$, $\mathbf{w}^{(0)}$ and some pre-computed formulas, i.e., the norm of the Hessian matrices, $\|-\nabla^2_{\mathbf{w}} \log(p^{(j)}(\mathbf{w}^{(0)}, \tilde{\mathbf{x}}))\|$ and $\|\mathbf{H}(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})\|$, and the gradients, $\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})$ and $\nabla_{\mathbf{w}} F(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})$, which can be computed as "provenance" information in the initialization step. Note that pre-computing $\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})$ and $\nabla_{\mathbf{w}} F(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})$ is quite straightforward by leveraging Equation (8). Then the remaining question is how to compute $\| -\nabla^2_{\mathbf{w}} \log(p^{(j)}(\mathbf{w}^{(0)}, \tilde{\mathbf{x}}))\|$ and $\|\mathbf{H}(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})\|$ efficiently without explicitly evaluating the Hessian matrices. Since those two terms calculate the norm of one Hessian matrix, we therefore only take one of them as a running example to describe how to compute them in a feasible way, as shown below.

Pre-computing $\|\mathbf{H}(\mathbf{w}^{(0)}, \tilde{\mathbf{z}})\|$ Since 1) a Hessian matrix is symmetric (due to its positive definiteness); and 2) the L2-norm of a symmetric matrix is equivalent to its eigenvalue with the largest magnitude [26], the L2 norm of one Hessian matrix is thus equivalent to its largest eigenvalue. To evaluate this eigenvalue, we use the Power method [12], which is discussed in the extended technical report [40].

Time complexity of Increm-INFL By assuming that there are n samples left after Increm-INFL is used, the dimension of vectorized $\mathbf{w}$ is m, and the running time of computing the vector $\mathbf{v}$ and the gradient ($\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$ or $\nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$) is denoted by $O(v)$ and $O(\text{Grad})$ respectively, the time complexity of Increm-INFL is $O(v) + NC(O(Cm) + O(m) + O(C)) + ncO(\text{Grad})$ (see the extended technical report [40] for a detailed analysis).

Algorithm 1 Increm-INFL

```
Input: The number of samples to be cleaned at the $k_{th}$ round: $b$
Output: A set of prioritized training samples for cleaning: $\mathcal{Z}_{inf}^{(k)}$
1: Initialize $\mathcal{Z}_{inf}^{(k)} = \{\}$
2: Calculate $\mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma)$ and the perturbation bound on this term by using Theorem 1 for each uncleaned sample, $\tilde{\mathbf{z}} = (\tilde{\mathbf{x}}, \tilde{y})$, and each label perturbation, $\delta_y = \tilde{y} - onehot(c), (c = 1, 2, \dots, C)$
3: Add the training samples producing Top-$b$ smallest $\mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma)$ to $\mathcal{Z}_{inf}^{(k)}$
4: obtain the largest perturbation upper bound, $L$, for all Top-$b$ smallest $\mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma)$
5: For any remaining training sample, $\tilde{\mathbf{z}}$, if its lower perturbation bound of $\mathcal{I}_0(\tilde{\mathbf{z}}, \delta_y, \gamma)$ is smaller than $L$ with a certain $\delta_y$, add it to $\mathcal{Z}_{inf}^{(k)}$
6: Return $\mathcal{Z}_{inf}^{(k)}$
```

4.2 The model constructor phase (DeltaGrad-L)

At the k_{th} round of loop $\textcircled{2}$, after the human annotators clean the labels for a set of b influential training samples, $\mathcal{R}^{(k)}$, provided by the Sample selector, the next step is to update the model parameters in the Model constructor. To reduce the overhead of this step, we can regard the process of updating labels as two sub-processes, i.e. the deletions of the training samples, $\mathcal{R}^{(k)}$ (with the associated weight, γ), and the additions of those training samples with the cleaned labels (with the updated weight, 1), thus facilitating the use of DeltaGrad. Specifically, the following modifications to Equation (4) are required: 1) the input deleted training samples should be $\mathcal{R}^{(k)}$; 2) the input cached model parameters and the corresponding gradients become the ones obtained at the $k-1_{st}$ round of the loop $\textcircled{2}$; 3) instead of randomly sampling a mini-batch $\mathcal{A}_t$ from the added training samples $\mathcal{A}$, $\mathcal{A}_t$ should be replaced with the removed training samples from $\mathcal{B}_t$, i.e., $\mathcal{B}_t \bigcap \mathcal{R}^{(k)}$, but with updated labels; 4) the cleaned training samples and the uncleaned training samples so far are weighted by 1 and γ respectively (this only slightly modifies how the loss is calculated for each mini-batch). Recall that in DeltaGrad, there are three hyper-parameters, i.e. the number of "burn-in" iterations, j_0, the period, T_0, and the history size, m_0, which are thus also essential for DeltaGrad-L.

4.3 The human annotation phase

As discussed in Section 1, the Sample selector not only suggests which samples should be cleaned, but also suggests the candidate cleaned labels, which can be regarded as one independent label annotator. When multiple annotators exist, we aggregate their labels by using majority vote to resolve possible label conflicts.

Discussion Note that only DeltaGrad-L and Increm-INFL explicitly rely on the strong convexity assumption on the model class. In contrast, INFL is potentially applicable for general machine learning models. Hence, we also conducted some initial experiments to evaluate the performance of INFL when neural network models are used (see the extended technical report [40]).

5 EXPERIMENTS

We conducted extensive experiments in Python 3.6 and PyTorch 1.7.0 [29]. All experiments were conducted on a Linux server with an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 3 GeForce 2080 Titan RTX GPUs and 754GB of main memory.

5.1 Experimental setup

Two types of datasets are used, one of which is annotated with ground-truth labels but no human generated labels, while the other is fully annotated with crowdsourced labels but only partially annotated by ground-truth labels. The former type (referred to as *Fully clean datasets*) is used to evaluate the quality of labels suggested by INFL by comparing them against the ground-truth labels. The latter type (referred to as *Crowdsourced datasets*) is used for evaluating the performance of our methods in more realistic settings. The two types of datasets are briefly described next.

Fully clean datasets: Three real medical image datasets are used: MIMIC-CXR-JPG (MIMIC for short) [19], Chexpert [17] and Diabetic Retinopathy Detection (Retina for short) [13]. The datasets are used to identify whether one or more diseases or findings exist for each image sample. In the experiments, we are interested in predicting the existence of findings called "Lung Opacity", "the referable Diabetic Retinopathy" and "Cardiomegaly" for MIMIC, Retina and Chexpert, respectively.

Crowdsourced datasets: Three realistic crowdsourced datasets are used: Fashion 10000 (Fashion for short)[2] [23], Fact Evaluation Judgement (Fact for short)[3], and Twitter sentiment analysis (Twitter for short)[4]. Only a small portion of samples in the datasets have ground-truth labels while the rest are labeled by crowdsourcing workers (e.g., the labels of the Fashion dataset are collected through the Amazon Mechanical Turk (AMT) crowdsourcing platform). The Fashion dataset is an image dataset for distinguishing fashionable images from unfashionable ones; the Fact dataset uses RDF triples to represent facts about public figures and the classification task is to judge whether or not each fact is true; and the Twitter dataset consists of plain-text tweets on major US airlines for sentiment analysis, i.e., identifying positive or negative tweets. For the Fashion and Fact datasets, extra text is also associated with each sample, e.g. user comments on each image in Fashion and the evidence for each fact in Fact, which is critical for producing probabilistic labels (see discussion below).

Since some samples in the datasets have missing or unknown ground-truth labels, we remove them in the experiments. Also, except for MIMIC which has 579 validation samples and 1628 test samples, other datasets do not have well-defined validation and test set. For example, as of the time the experiments were performed, the test samples of Chexpert had not been released. To remedy this, we partition the Chexpert validation set into two parts to create validation and test sets, each of which have 234 samples. Since there was no validation set for Retina, we randomly select roughly 10% of the training samples, i.e., 3512 samples, as the validation set. Similarly, for the Twitter and Fact datasets, we randomly partition the set of samples with ground-truth labels as the validation set and test set, and regard all the other samples as the training set. Since ground-truth labels are not available in the Fashion dataset, we randomly select roughly 0.5%[5] of the samples as the validation set and test set, each containing 146 samples. The "ground-truth" labels for those samples are determined by aggregating human annotated labels using majority vote. The remaining samples in this dataset are then regarded as training samples. In the end, the six datasets, i.e., MIMIC, Retina, Chexpert, Fashion, Fact and Twitter include ~78k,~31k,~38k,~29k,~38k and ~12k training samples.

Producing probabilistic labels Due to the lack of probabilistic labels or labeling functions [31] for the datasets, we can choose to leverage [3], [37] or [7] to *automatically* derive suitable labeling functions and thus probabilistic labels in the experiments. Note that [3] and [37] deal with text data (including the text associated with image data) while [7] targets pure image data. However, the time and space complexity of [7] is quadratic in the dataset size, and does not scale to large image datasets such as our *Fully clean datasets*. Furthermore, no text information is available for images in *Fully clean datasets*, so it is not feasible to use [3] or [37]. As a result, random probabilistic labels are produced for all training samples. For *Crowdsourced datasets*, we apply [3] on the extra text information in Fashion (e.g. user comments for each image) and the plain-text tweets in Twitter dataset to produce probabilistic labels. For Fact dataset, the two texts for each sample (i.e. the RDF triples and the associated evidence) are compared using [6] to generate labeling functions.

Human annotator setup For *Crowdsourced datasets*, we can use the crowdsourced labels as the cleaned labels for the uncleaned training samples. However, such labels are unavailable in *Fully clean datasets*. To remedy this, we note that the error rate of manually labeling medical images is typically between 3% and 5% , but sometimes can be up to 30% [4]. We therefore produce synthetic human annotated labels by flipping the ground truth labels of a randomly selected 5% of the samples [6]. We assume three independent annotators, and aggregate their labels as the cleaned labels using majority vote (denoted INFL (one)). Since INFL can suggest cleaned labels, those labels can be used as cleaned labels by themselves for the uncleaned samples (denoted INFL (two)) or be combined with labels provided by two other simulated human annotators for label cleaning (denoted INFL (three)).

Model constructor setup Throughout the paper we assume that strong convexity holds on the ML models. Therefore, in this section, to justify the performance advantage of our design as a whole (including Increm-INFL, DeltaGrad-L and INFL), we focus on a scenario where pre-trained models are leveraged for feature transformation and then a logistic regression model is used for classification, which has emerged as a convention for medical image classification tasks [30]. Specifically, in the experiments, we use a pre-trained ResNet50 [15] for the image datasets (*Fully clean datasets* and Fashion), and use a pre-trained BERT-based transformer [8] for the text datasets (Fact and Twitter). Stochastic gradient descent (SGD) is then used in the subsequent training process with a mini-batch size of 2000, and weight $\gamma = 0.8$ on the uncleaned samples. Early stopping is also applied to avoid overfitting. Other hyper-parameters for model training are varied across different datasets (see Table 1). We also vary γ for a more extensive comparison in [40].

As discussed in Section 1, other than the initialization step, we can construct the models by either retraining from scratch (denoted Retrain) or leveraging DeltaGrad-L for incremental updates. For the latter case, the hyper-parameters of DeltaGrad-L are configured as $m_0 = 2$, $j_0 = 10$ and $T_0 = 20$ for all six datasets.

Table 1: The hyper-parameters for each dataset

Dataset	MIMIC	Retina	Chexpert	Fashion	Fact	Twitter
Learning rate	0.0005	0.001	0.02	0.05	0.005	0.01
L2 regularization	0.05	0.05	0.05	0.001	0.01	0.01
# of epochs	150	200	200	200	150	400

[2]available at http://skuld.cs.umass.edu/traces/mmsys/2014/user05.tar (accessed 26 Jul. 2021)
[3]available at https://sites.google.com/site/crowdscale2013/shared-task/task-fact-eval (accessed 26 Jul. 2021)
[4]available at https://github.com/naimulhuq/Capstone/blob/master/Data/Airline-Full-Non-Ag-DFE-Sentiment%20(raw%20data).csv (accessed 26 Jul. 2021)
[5]This ratio is determined based on the observation that in the Twitter and Fact datasets, the percentage of samples with ground-truth labels is less than 1% of the size of the entire dataset.

[6]Recall that although the samples have probabilistic labels, their true labels are known by construction.

Sample selector setup We assume that the clean budget $B = 100$, meaning that 100 training samples are cleaned in total. We further vary the number of samples to be cleaned at each round, i.e. the value of b.

Baseline against INFL We compare INFL against several baseline methods that can also prioritize the most influential samples for label cleaning in the sample selector, including the original version of the influence function, i.e. Equation (2) [20] (denoted by INFL-D), two active learning methods—least confidence based sampling method (denoted by Active (one)) and entropy based sampling method (denoted by Active (two)) [33]—and one noisy sample detection algorithm, O2U [16]. Note that for fair comparison, Equation (1) is used for model training no matter which method is used in the sample selector.

Note that there also exist many other potential baseline methods, such as DUTI [41] and TARS [9]. However, they are not applicable in the presence of either probabilistic labels or the regularization on uncleaned training samples. Therefore, we adjust the experimental set-up to create a fair comparison between INFL and those methods, which is included in the extended technical report [40].

Baseline against DeltaGrad-L and Increm-INFL Recall that DeltaGrad-L incrementally updates the model after some training samples are cleaned. We compare this with retraining the model from scratch (denoted as Retrain). We also compare the running time for selecting the influential training samples with and without Increm-INFL. When Increm-INFL is not used, it is denoted as Full.

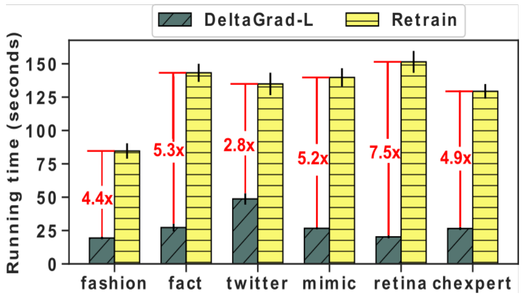

Figure 2: Comparison of running time between DeltaGrad-L and Retrain

5.2 Experimental design

In this section, we design the following three experiments:

Exp1 In this experiment, we compared the model prediction performance after INFL and other baseline methods (including INFL-D, Active (one), Active (two), and O2U) are applied to select 100 training samples for cleaning. Recall that there are three different strategies that INFL can use to provide cleaned labels and their performance is compared. To show the benefit of using a smaller batch size b, we choose two different values for b, i.e. 100 and 10. Since the ground-truth labels are available for all samples in *Fully clean datasets*, we count how many of them match the labels suggested by INFL.

Exp2 This experiment compares the running time of selecting the Top-b (with $b = 10$) influential training samples (denoted Time$_{inf}$) with and without using Increm-INFL at each round in the *Sample selector* phase. Recall that the most time-consuming step to evaluate Equation (6) is to compute the class-wise gradients for each sample, $\nabla_y \nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$, and the sample-wise gradients, $\nabla_{\mathbf{w}} F(\mathbf{w}, \tilde{\mathbf{z}})$. Therefore, its running time (denoted as Time$_{grad}$) is also recorded. For Increm-INFL, the time to compute the bounds in Theorem 1 is also included in Time$_{inf}$.

Exp3 The main goal of this experiment is to explore the difference in running time between Retrain and DeltaGrad-L for updating the model parameters in the *Model constructor* phase. In addition, the model parameters produced by DeltaGrad-L and Retrain are not exactly the same [38], which could lead to different influence values for each training sample and thus produce different models in subsequent cleaning rounds. Therefore, we also explore whether such differences produce divergent prediction performance for DeltaGrad-L and Retrain.

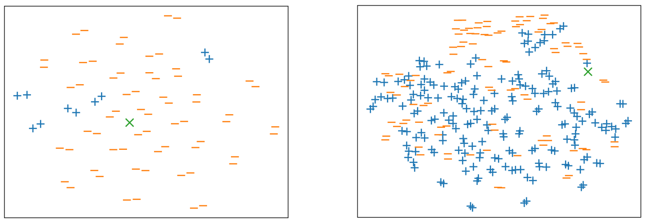

Figure 3: Visualization of the validation samples, test samples and the most influential training sample S ('+', '-' and 'X' denote the positive ground-truth samples, negative ground-truth samples and the sample S respectively)

5.3 Experimental results

Exp1 Experimental results are given in Table 2[7]. We observe that with fixed b, e.g., 10, INFL (two) performs best across almost all datasets. Recall that INFL (two) uses the derived labels produced by INFL as the cleaned labels without additional human annotated labels. Due to its superior performance, especially on *Crowdsourced datasets*, this implies that the quality of the labels provided by INFL could actually be better than that of the human annotated labels.

To further understand this phenomenon, we compared the labels suggested by INFL against their ground-truth labels for *Fully clean datasets*. It turns out that over 70% are equivalent (89 for Retina, 79 for Chexpert and 95 for MIMIC). Note that even the ground-truth labels of these three datasets are not 100% accurate. In the Chexpert dataset, for example, the ground-truth labels are generated by an automated labeling tool rather than being labeled by human annotators, thereby leading to possible labeling errors. Those erroneous labels may not match the labels provided by INFL, thus leading to worse model performance (see the performance difference between INFL (one) and INFL (two) for Chexpert dataset).

However, the above comparison could not be done for *Crowdsourced datasets* due to the lack of ground-truth labels. We therefore investigate the relationship between the samples with ground-truth

[7]Due to space, the error bars of F1 scores are not shown, but are included in the extended technical report [40]

Table 2: Comparison of the model prediction performance (F1 score) after 100 training samples are cleaned

		b=100			b=10							
	uncleaned	INFL (one)	INFL (two)	INFL (three)	INFL-D	Active (one)	Active (two)	O2U	INFL (one)	INFL (two)	INFL (two) + DeltaGrad	INFL (three)
MIMIC	0.6284	0.6292	0.6293	0.6293	0.6283	0.6287	0.6287	0.1850	0.6292	**0.6293±0.0011**	**0.6292±0.0005**	0.6292
Retina	0.5565	0.5580	**0.5582**	0.5581	0.5556	0.5568	0.5568	0.1314	0.5579	**0.5582±0.0003**	**0.5610±0.0010**	0.5581
Chexpert	0.5244	0.5286	0.5297	0.5289	0.5246	0.5246	0.5246	0.5281	0.5287	**0.5300±0.0018**	**0.5295±0.0030**	0.5291
Fashion	0.5140	0.5178	0.5177	0.5177	0.5143	0.5145	0.5145	0.5152	0.5178	**0.5181±0.0131**	**0.5195±0.0144**	0.5180
Fact	0.6595	0.6601	**0.6609**	0.6603	0.6596	0.6600	0.6600	0.6598	0.6601	**0.6609±0.0043**	**0.6609±0.0065**	0.6602
Twitter	0.6485	0.6594	0.6680	0.6594	0.6518	0.6515	0.6515	0.6490	0.6578	**0.6697±0.0058**	**0.6597±0.0027**	0.6586

Table 3: Running time of Increm-INFL and Full

	Time_{inf} (s)		Time_{grad} (s)	
	Full	Increm-INFL	Full	Increm-INFL
MIMIC	151.4±0.5	**2.77±0.03 (54.7x)**	145.4±0.7	**0.17±0.03 (855x)**
Retina	74.0±0.6	**1.36±0.04 (54.4x)**	70.8±0.6	**0.21±0.03 (337x)**
Chexpert	72.5±0.2	**17.9±1.9 (4.1x)**	69.3±0.2	**14.7±1.5 (4.7x)**
Fashion	66.4±3.6	**8.7±0.6 (7.6x)**	57.1±3.3	**0.81±0.07 (70.5x)**
Fact	73.8±4.0	**6.1±0.8 (12.1x)**	72.5±6.0	**4.7±0.1 (15.4x)**
Twitter	33.1±2.3	**14.1±0.4 (2.3x)**	30.2±1.1	**12.7±0.1 (2.4x)**

labels and the influential samples identified by INFL. Specifically, we use t-SNE [36] to visualize the samples with ground-truth labels for the Twitter and Fashion datasets after the feature transformation using the pre-trained models (see Figure 3). As described in Section 5.1, those samples belong to the validation or test set. In addition, in this figure, we indicate the position of the most influential training sample S identified by INFL. As this figure indicates, the sample S is proximal to the samples with negative ground-truth labels for the Twitter dataset (positive for the Fashion dataset). To guarantee accurate predictions on those nearby ground-truth samples, it is therefore reasonable to label S as negative (positive for Fashion dataset), which matches the labels provided by INFL but differs from ones given by the human annotators. This indicates the high quality of the labels given by INFL. Thus, when high-quality human labelers are unavailable, INFL can be an alternative labeler for reducing the labeling cost without harming the labeling quality.

Table 2 also exhibits the benefit of using smaller batch sizes b since it results in better model performance when INFL, especially INFL (two), is used for some datasets (e.g, see its model performance comparison between $b = 100$ and $b = 10$ for the Twitter dataset in Table 2). Intuitively, INFL only quantifies the influence of cleaning *single* training sample rather than multiple ones. Therefore, the larger b is, the more likely that INFL selects a sub-optimal set of b samples for cleaning. Ideally, b should be one, meaning that one training sample is cleaned at each round. However, this can inevitably increase the number of rounds and thus the overall overhead. Intuitively speaking, a medium size of b is a reasonable choice to balance the model performance and the total running time, which is empirically demonstrated in [40].

Exp2 In this experiment, we compare the running time of Increm-INFL and Full in selecting the Top-b influential training samples (Time_{inf}) at each cleaning round of the loop ② (with $b = 10$). Due to space, we only include results for the last round in Table 3, which are similar to results in other rounds. As Table 3 indicates, Increm-INFL is up to 54.7x faster than Full, which is due to the significantly decreased overhead of computing the class-wise gradients for each sample (i.e. Time_{grad}) when Increm-INFL is used. To further illustrate this point, we also record the number of candidate influential training samples whose influence values are explicitly evaluated with and without using Increm-INFL. The result indicates that due to the early removal of uninfluential training samples using Increm-INFL, we only need to evaluate the influence of a small portion of training samples, thus reducing Time_{grad} by up to two orders of magnitude and thereby significantly reducing the total running time, Time_{inf}. In addition, we observe that Increm-INFL always returns the same set of influential training samples as Full, which thus guarantees the correctness of Increm-INFL.

Exp3 Experimental results of **Exp3** are shown in Figure 2. The first observation is that DeltaGrad-L can achieve up to 7.5x speed-up with respect to Retrain on updating the model parameters. As shown in Section 5.2, the models updated by DeltaGrad-L are not exactly the same as those produced by Retrain, which might cause different model performance between the two methods. However, we observe that the models constructed by those two methods have almost equivalent prediction performance (see the second to last column in Table 2). This indicates that it is worthwhile to leverage DeltaGrad-L for speeding up the model constructor.

6 CONCLUSIONS

In this paper, we propose CHEF, which can reduce the overall cost of the label cleaning pipeline and achieve better model performance than other approaches; it may also allow early termination in the human annotation phase. Extensive experimental studies show the effectiveness of our solution over a broad spectrum of real datasets when strongly convex models are used. How to extend CHEF, in particular, Increm-INFL and DeltaGrad-L, beyond strongly convex models is left as future work.

ACKNOWLEDGMENTS

This material is based upon work that is in part supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0047.

REFERENCES

[1] Leonel Aguilar Melgar, David Dao, Shaoduo Gan, Nezihe M Gürel, Nora Hollenstein, Jiawei Jiang, Bojan Karlaš, Thomas Lemmin, Tian Li, Yang Li, et al. 2021. Ease. ML: A Lifecycle Management System for Machine Learning. In *11th Annual Conference on Innovative Data Systems Research (CIDR 2021)(virtual)*. CIDR.

[2] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. 2019. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*. 362–375.

[3] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. 2020. Interactive Weak Supervision: Learning Useful Heuristics for Data Labeling. *arXiv preprint arXiv:2012.06046* (2020).

[4] Adrian P Brady. 2017. Error and discrepancy in radiology: inevitable or avoidable? *Insights into imaging* 8, 1 (2017), 171–182.

[5] Jonathan Brophy and Daniel Lowd. 2020. DART: Data Addition and Removal Trees. *arXiv preprint arXiv:2009.05567* (2020).

[6] Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2019. Data Programming using Continuous and Quality-Guided Labeling Functions. *arXiv preprint arXiv:1911.09860* (2019).

[7] Nilaksh Das, Sanya Chaba, Renzhi Wu, Sakshi Gandhi, Duen Horng Chau, and Xu Chu. 2020. GOGGLES: Automatic Image Labeling with Affinity Coding. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1717–1732.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Mohamad Dolatshah, Mathew Teoh, Jiannan Wang, and Jian Pei. [n.d.]. Cleaning Crowdsourced Labels Using Oracles for Statistical Classification. *Proceedings of the VLDB Endowment* 12, 4 ([n. d.]).

[10] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*. PMLR, 2242–2251.

[11] Antonio Ginart, Melody Y Guan, Gregory Valiant, and James Zou. 2019. Making ai forget you: Data deletion in machine learning. *arXiv preprint arXiv:1907.05012* (2019).

[12] Gene H Golub and Henk A Van der Vorst. 2000. Eigenvalue computation in the 20th century. *J. Comput. Appl. Math.* 123, 1-2 (2000), 35–65.

[13] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. 2016. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama* 316, 22 (2016), 2402–2410.

[14] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*. 8527–8537.

[15] 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.

[16] Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. 2019. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 3326–3334.

[17] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. 2019. CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Thirty-Third AAAI Conference on Artificial Intelligence*.

[18] 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.

[19] Alistair EW Johnson, Tom J Pollard, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Yifan Peng, Zhiyong Lu, Roger G Mark, Seth J Berkowitz, and Steven Horng. 2019. MIMIC-CXR-JPG, a large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042* (2019).

[20] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 1885–1894.

[21] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. 2016. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment* 9, 12 (2016), 948–959.

[22] EB Leach and MC Sholander. 1978. Extended mean values. *The American Mathematical Monthly* 85, 2 (1978), 84–90.

[23] Babak Loni, Lei Yen Cheung, Michael Riegler, Alessandro Bozzon, Luke Gottlieb, and Martha Larson. 2014. Fashion 10000: an enriched social image dataset for fashion and clothing. In *Proceedings of the 5th acm multimedia systems conference*. 41–46.

[24] Mohammad Mahdavi, Felix Neutatz, Larysa Visengeriyeva, and Ziawasch Abedjan. 2019. Towards automated data cleaning workflows. *Machine Learning* 15 (2019), 16.

[25] James Martens. 2010. Deep learning via hessian-free optimization.. In *ICML*, Vol. 27. 735–742.

[26] Carl D Meyer. 2000. *Matrix analysis and applied linear algebra*. Vol. 71. Siam.

[27] Mona Nashaat, Aindrila Ghosh, James Miller, and Shaikh Quader. 2020. WeSAL: Applying active supervision to find high-quality labels at industrial scale. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

[28] Jorge Nocedal. 1980. Updating quasi-Newton matrices with limited storage. *Mathematics of computation* 35, 151 (1980), 773–782.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[30] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. 2019. Transfusion: Understanding transfer learning for medical imaging. *arXiv preprint arXiv:1902.07208* (2019).

[31] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.

[32] Lawrence Ryz and Lauren Grest. 2016. A new era in data protection. *Computer Fraud & Security* 2016, 3 (2016), 18–20.

[33] Burr Settles. 2009. Active learning literature survey. (2009).

[34] L Smyth. 2020. Training-ValueNet: A new approach for label cleaning on weakly-supervised datasets. (2020).

[35] Sainbayar Sukhbaatar and Rob Fergus. 2014. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080* 2, 3 (2014), 4.

[36] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[37] Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 12. NIH Public Access, 223.

[38] Yinjun Wu, Edgar Dobriban, and Susan Davidson. 2020. DeltaGrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*. PMLR, 10355–10366.

[39] Yinjun Wu, Val Tannen, and Susan B Davidson. 2020. PrIU: A Provenance-Based Approach for Incrementally Updating Regression Models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 447–462.

[40] Yinjun Wu, James Weimer, and Susan B Davidson. 2021. CHEF: A Cheap and Fast Pipeline for Iteratively Cleaning Label Uncertainties (Technical Report). *arXiv preprint arXiv:2107.08588* (2021).

[41] Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. 2018. Training set debugging using trusted items. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.