The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Learning to Select Pivotal Samples for Meta Re-weighting

Yinjun Wu, Adam Stein, Jacob Gardner, Mayur Naik

University of Pennsylvania
wuyinjun@seas.upenn.edu, steinad@seas.upenn.edu, jacobrg@seas.upenn.edu, mhnaik@cis.upenn.edu

Abstract

Sample re-weighting strategies provide a promising mechanism to deal with imperfect training data in machine learning, such as noisily labeled or class-imbalanced data. One such strategy involves formulating a bi-level optimization problem called the *meta re-weighting problem*, whose goal is to optimize performance on a small set of perfect pivotal samples, called *meta samples*. Many approaches have been proposed to efficiently solve this problem. However, all of them assume that a perfect meta sample set is already provided while we observe that the selections of meta sample set is performance-critical. In this paper, we study how to *learn* to identify such a meta sample set from a large, imperfect training set, that is subsequently cleaned and used to optimize performance in the meta re-weighting setting. We propose a learning framework which reduces the meta samples selection problem to a weighted K-means clustering problem through rigorously theoretical analysis. We propose two clustering methods within our learning framework, Representation-based clustering method (RBC) and Gradient-based clustering method (GBC), for balancing performance and computational efficiency. Empirical studies demonstrate the performance advantage of our methods over various baseline methods.

Introduction

Recently, with the advent of the data-centric AI era (Miranda 2021; Polyzotis and Zaharia 2021; Hajij et al. 2021), there is an increasing concern about the *quality* of data for training neural network models. How to construct and maintain a high-quality data set is extremely challenging due to the existence of various defects in real-life data, e.g., imperfect labels or imbalanced distributions across classes. To tackle these issues, various techniques have been explored. One such example is the sample re-weighting strategy (Shu et al. 2019; Ren et al. 2018; Hu et al. 2019; Jiang et al. 2018; Chang, Learned-Miller, and McCallum 2017), which targets jointly learning to obtain *re-weighted* training samples and training neural nets upon them.

One promising strategy for learning to re-weight training samples is to leverage the framework of meta learning (Hospedales et al. 2021; Andrychowicz et al. 2016; Thrun and Pratt 2012) by formulating this problem as a bi-level optimization problem (Shu et al. 2019; Ren et al. 2018; Hu et al. 2019). In this approach, the weights of training samples are learned so that the performance of the models learned on the re-weighted training samples is maximized on a small set of perfect samples—referred to as *meta samples*. Existing works mainly focus on designing computationally efficient algorithms for solving this bi-level optimization problem. For example, (Shu et al. 2019) propose a *meta re-weighting* algorithm which alternates between updates to the model parameters and the sample weights. These algorithms, however, rely on the assumption that the meta sample set is *given*, and they construct this set by random sampling in their empirical studies. However, as the toy example in Figure 1 shows, randomly selected meta samples may perform worse than carefully selected ones by using our methods (62.9% vs. 87.1% on test accuracy), which we further verify in Section **"Experiments"**.

In this paper, we study how to learn to identify a set of meta samples from a large, imperfect training set such that the *meta re-weighting* performance is optimized. Specifically, we propose a framework which reduces the problem of selecting such meta samples to a weighted K-means clustering problem through rigorous theoretical analysis. This derivation basically transforms the formula for iteratively updating sample weights from the meta re-weighting algorithm into a weighted K-means clustering objective function. We can show that *optimizing this objective function can aid in effectively distinguishing high-quality training samples from low-quality ones by giving them more confident sample weights (i.e. weights close to 0 or 1)*. This objective function, however, requires the gradients of *each individual training sample* as input, which is computationally expensive. To facilitate efficient evaluation of this objective function, we propose two methods, i.e. Representation-based clustering method (RBC) and Gradient-based clustering method (GBC), which balance performance with computational efficiency. Specifically, by assuming that the gradients of the bottom layers of the neural nets are insignificant, RBC only utilizes the gradient of the last layer, which is efficiently calculated through feed-forward passes. In contrast, GBC samples model parameters such that the estimation of the objective function in the above K-means problem is unbiased. Due to the necessity of explicitly (but partially) computing sample-wise gradients, GBC is slower than RBC,

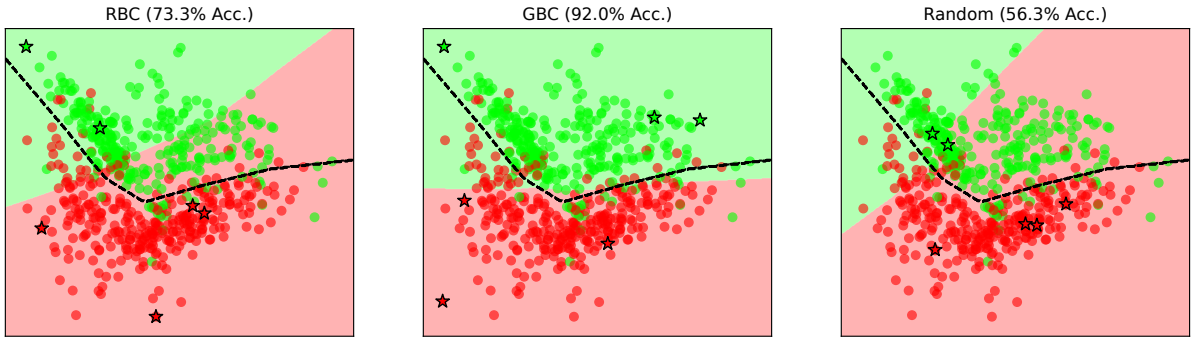

Figure 1: We produce a toy two-dimensional dataset by drawing 1000 samples from a mixture of four Gaussian distributions over two variables where the distributions are centered at the four vertices of the 2-dimensional hypercube. The upper two distributions are labeled green while the lower two are labeled red and 1% of the labels are flipped to introduce a small amount of noise to the ground truth. We visualize all the samples with their ground-truth labels in this figure. This toy dataset is then divided into 600 training, 240 testing, and 160 validation samples using a random partition, and a randomly selected 60% of the training labels are flipped. To learn a robust model (which is a neural network with two hidden layers in this example) on this noisy training set, we employ the meta-reweighting algorithm with 6 cleaned meta samples. We then show the selected meta samples (outlined with stars) and the learned decision boundaries (shaded regions) from our methods and the random selection method. The expected classifier, i.e., the learned classifier (black dotted line) on the "clean" training set is also visualized. By examining the learned classifier from these methods, we see that the one learned by random selections deviates farther from the expected classifier, thus leading to worse prediction performance than our methods (56.3% vs. 92.0%).

but can lead to better model performance in most cases.

We further explore whether our methods select reasonable meta samples for re-weighting noisily labeled data and class-imbalanced data by conducting experiments on re-weighting MNIST, CIFAR, and Imagenet-10 datasets in the presence of noisy labels or imbalanced class distribution. The results show that with the same meta re-weighting algorithm, our methods outperform other sample selection strategies in most cases.

Related Work

Sample re-weighting The problem of re-weighting training samples for a neural network model has been extensively studied in the literature. Sample re-weighting can be beneficial for constructing robust neural network models in the presence of many defects in training data, such as corrupted labels (Han et al. 2018; Ren et al. 2018; Shu et al. 2019), biased distributions (Khan et al. 2017; Dong, Gong, and Zhu 2017), low cardinalities (Hu et al. 2019) and adversarial attacks (Holtz, Weng, and Mishne 2021). Other than solving this problem within the meta-learning framework (e.g., (Shu et al. 2019; Ren et al. 2018; Hu et al. 2019)), various strategies have been proposed for deriving sample weights. For example, in (Wang, Kucukelbir, and Blei 2017), the sample weights are modeled as a Bayesian latent variable and inferred through probabilistic models. In (Jiang et al. 2018), a mentor network is designed to derive the sample weights such that the target model does not overfit on samples with noisy labels, which falls within the curriculum learning (Bengio et al. 2009) framework. In (Kumar, Packer, and Koller 2010), the weights of training samples are determined by their training loss during the training process. However, as (Shu et al. 2019) suggests, these re-weighting techniques all perform worse than the meta re-weighting algorithm in the presence of label noise and distribution imbalance in training data.

Data efficiency As mentioned in Section **"Introduction"**, it is critical to obtain large amounts of high-quality training samples for deep neural nets. However, this can be expensive and time consuming since labeling typically requires non-trivial work from human annotators, especially in scientific domains (see e.g, (Karimi et al. 2020; Irvin et al. 2019)). High labeling cost is thus a strong motivator for studies on various label efficiency techniques, e.g., active learning (see a survey in (Ren et al. 2021) and some recent works (Mirzasoleiman, Bilmes, and Leskovec 2020)), semi-supervised learning (see (Van Engelen and Hoos 2020)), and weakly-supervised learning (see Snorkel (Ratner et al. 2017)) in the past few years. All of these studies aim at minimizing human labeling effort while maintaining relatively high model performance. Note that for the meta re-weighting problem, the construction of perfect meta samples also requires human labeling effort when label noise exists. Therefore, our framework shares the same spirit as the traditional label efficiency research.

Data valuation In the literature, other than active learning, there exists many techniques to quantify the importance of individual samples, e.g., influence function (Koh and Liang 2017) and its variants (Wu, Weimer, and Davidson 2021), Glister (Killamsetty et al. 2021), HOST-CP (Das et al. 2021), TracIn (Pruthi et al. 2020), DVRL (Yoon, Arik, and Pfister 2020) and Data Shapley value (Ghorbani and Zou 2019). However, among these methods, Data Shapley value (Ghorbani and Zou 2019) is very computationally expensive while others rely on the assumption that a set of "clean" validation samples (or meta samples) are given, which is thus

not suitable for our framework (we have more detailed discussions on Data Shapley value and its extensions in Appendix **"Appendix: more related work"**). We therefore do not include these solutions as baseline methods.

Background: The Meta Re-weighting Method

In this section, we present some necessary details on the meta re-weighting method from (Shu et al. 2019).

Suppose the meta re-weighting method is conducted on a large imperfect training set, $D_{\text{train}} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N}$ and a small perfect meta set $D_{\text{m}} = \{(\mathbf{x}_{\text{m},i}, y_{\text{m},i})\}_{i=1}^{M}$. Imagine that we want to learn a model parameterized by Θ, and the loss evaluated on a training sample $(\mathbf{x}_j, y_j)$ and a meta sample $(\mathbf{x}_{\text{m},i}, y_{\text{m},i})$ is denoted as $f_j(\Theta)$ and $f_{\text{m},i}(\Theta)$ respectively. We further denote the weight of each training sample j as w_j (between 0 and 1). Following (Shu et al. 2019), the meta re-weighting method jointly learns the weights $\mathbf{W} = \{w_j\}_{j=1}^{N}$ and the model parameter Θ by solving the following bi-level optimization problem:

$$\begin{aligned} &\min_{\mathbf{W}} \frac{1}{M}\sum\nolimits_{i=1}^{M} f_{\text{m},i}(\Theta^*(\mathbf{W})), \\ &\text{s.t. } \Theta^*(\mathbf{W}) = \text{argmin}_{\Theta} \frac{1}{N}\sum\nolimits_{j=1}^{N} w_j f_j(\Theta), \end{aligned} \tag{1}$$

in which $\Theta^*(\mathbf{W})$ denotes the learned model parameters on the training set weighted by $\mathbf{W}$. This problem can be efficiently solved by the meta re-weighting method proposed by (Shu et al. 2019), which can be abstracted with the following formulas [1]:

Meta re-weighting:

$$\hat{\Theta}(\mathbf{W}_t) = \Theta_t - \frac{\alpha_t}{N}\sum\nolimits_{j=1}^{N} w_{j,t}\nabla_{\Theta} f_j(\Theta)|_{\Theta=\Theta_t} \tag{2}$$

$$w_{j,t+1} = w_{j,t} - \frac{\eta_t}{M}\sum\nolimits_{i=1}^{M} \nabla_{w_j} f_{\text{m},i}(\hat{\Theta}(\mathbf{W}_t))|_{\mathbf{W}=\mathbf{W}_t} \tag{3}$$

$$\Theta_{t+1} = \Theta_t - \frac{\alpha_t}{N}\cdot\sum\nolimits_{j=1}^{N} w_{j,t+1}\nabla_{\Theta} f_j(\Theta)|_{\Theta=\Theta_t} \tag{4}$$

The above formulas show how to update the model parameter and sample weights at the t^{th} iteration. Among these formulas, Equation (2) tries to update the model parameter Θ_t given the current sample weights $\mathbf{W}_t = [w_{1,t}, w_{2,t}, \ldots, w_{N,t}]$, which is then employed for updating the sample weights in Equation (3). Afterwards, in Equation (4), the updated sample weights, $\mathbf{W}_{t+1}$, are inserted into Equation (2) to obtain the model parameters for the next iteration, i.e., Θ_{t+1}. This process is then repeated until the convergence.

[1] Note that these formulas are slightly different from the ones in (Shu et al. 2019) since the sample weights in (Shu et al. 2019) are produced by another neural net. But its learning algorithm is also applicable to the case where the sample weights are updated directly. We therefore start from this simple case. Further note that (Ren et al. 2018) and (Hu et al. 2019) solve Equation (1) in a similar manner. Therefore, although we develop our methods mostly based on (Shu et al. 2019), they are also potentially applicable to the solutions in (Ren et al. 2018) and (Hu et al. 2019). We therefore discuss how it can be extended to (Ren et al. 2018), in Appendix **"Generalization of our methods for (Ren et al. 2018)"**

Method

Unlike (Shu et al. 2019; Hu et al. 2019; Ren et al. 2018) where the meta set D_{m} is assumed to be given, our goal is to select this set from D_{train}. Once this meta set is selected and possibly cleaned by humans (when noisy labels exist), the meta re-weighting algorithm can be used. We hope that the resulting model performance is optimized with respect to the sample selection strategy. We observe that one critical property of such D_{m} is that it needs to produce *"significant" cumulative gradient updates (rather than near-zero gradient) in Equation* (3) *for every training sample* $j(= 1, 2, \ldots, N)$ *and every iteration* t *in the meta re-weighting algorithm*. This can thus guarantee that good training samples are efficiently up-weighted while bad training samples are efficiently down-weighted. Therefore, our goal is to maximize the magnitude of the sum of the gradient in Equation (3) evaluated for each training sample j, across all iterations:

$$\begin{aligned} &\max_{D_{\text{m}}} \left| \sum\nolimits_{\hat{\Theta}(\mathbf{W}_t)} 1/M \cdot \sum\nolimits_{i=1}^{M} \nabla_{w_j} f_{\text{m},i}(\hat{\Theta}(\mathbf{W}_t)) \right|, \\ &\text{for all } j = (1, 2, \ldots, N), \end{aligned} \tag{5}$$

which we rewrite as follows according to (Shu et al. 2019) (the constant coefficients are ignored below):

$$\max_{D_{\text{m}}} \left| \sum_{\hat{\Theta}(\mathbf{W}_t),\Theta_t} \sum_{i=1}^{M} \langle \nabla_{\Theta} f_{\text{m},i}(\Theta)|_{\Theta=\hat{\Theta}(\mathbf{W}_t)}, \nabla_{\Theta} f_j(\Theta)|_{\Theta=\Theta_t} \rangle \right|, \tag{6}$$

which thus represents *the Frobenius inner product of the gradient of the loss* between the meta sample i and the training sample j. If the above inner product is large enough, the weight of this sample will be significantly updated. Since we want to maximize the updates of the weight of each training sample, we sum up the above formula over all training samples, leading to:

$$\max_{D_{\text{m}}} \sum_{j=1}^{N} \left| \sum_{\hat{\Theta}(\mathbf{W}_t),\Theta_t} \sum_{i=1}^{M} \langle \nabla_{\Theta} f_{\text{m},i}(\Theta)|_{\Theta=\hat{\Theta}(\mathbf{W}_t)}, \nabla_{\Theta} f_j(\Theta)|_{\Theta=\Theta_t} \rangle \right|,$$

which can be further approximated as follows by leveraging the fact that $\hat{\Theta}(\mathbf{W}_t)$, is very close to Θ_t:

$$\max_{D_{\text{m}}} \sum_{j=1}^{N} \left| \sum_{i=1}^{M} \sum_{\Theta_t} \underline{\langle \nabla_{\Theta} f_{\text{m},i}(\Theta)|_{\Theta=\Theta_t}, \nabla_{\Theta} f_j(\Theta)|_{\Theta=\Theta_t} \rangle} \right| \tag{7}$$

Equation (7) can be further rewritten as the following Meta-Sample Search Objective (MSSO):

$$\text{MSSO} := \text{Equation (7)} = \max_{D_{\text{m}}} \sum_{j=1}^{N} \left| \sum_{i=1}^{M} \langle G_j, G_{\text{m},i} \rangle \right|, \tag{8}$$

in which, we define $G_j = \left[G_j^{(1)}, \ldots, G_j^{(t)}, \ldots \right]$ and $G_{\text{m},i} = \left[G_{\text{m},i}^{(1)}, \ldots, G_{\text{m},i}^{(t)}, \ldots \right]$ as block matrices formed by concatenating the gradients, $G_{\text{m},i}^{(t)} := \nabla_{\Theta} f_{\text{m},i}(\Theta)|_{\Theta=\Theta_t}$ and $G_j^{(t)} := \nabla_{\Theta} f_j(\Theta)|_{\Theta=\Theta_t}$, from each iteration into one matrix.

Note that the meta sample set, D_{m}, needs to be selected from the training set, D_{train}. Thus, explicitly solving MSSO is computationally intractable since there are $\binom{N}{M}$ possible

selections of a meta set of size M. In what follows, we present an approximation to MSSO with rigorous guarantees, which can be effectively solved with a weighted K-means clustering algorithm.

Approximating MSSO

We show that with reasonable assumptions, solving MSSO is approximately equivalent to searching for a set of cluster centroids, $\mathcal{C} = \{C_i\}_{i=1}^M$, i.e.,:

$$\text{MSSO} \approx \max_{\mathcal{C}} \sum_{j=1}^{N} \left| \sum_{i=1}^{M} \langle G_j, C_i \rangle \right|$$

which can be approximated by solving the following M-clustering objective (MCO)

$$\text{MSSO} \approx \text{MCO} := \max_{\mathcal{C}} \sum_{j=1}^{N} \sum_{i=1}^{M} |\langle G_j, C_i \rangle|, \quad (9)$$

where MSSO is approximated by moving the absolute value to the inside of the sum. The approximation above can be justified by the following Theorem.

Theorem 1. *Suppose that for each sample i, the positive terms in the innermost sum of Equation* (8) *are dominant over the negative terms or vice versa, i.e.:*

$$\frac{|\sum_{\langle G_j, C_i \rangle > 0} \langle G_j, C_i \rangle|}{|\sum_{\langle G_j, C_i \rangle < 0} \langle G_j, C_i \rangle|} > D \gg 1,$$
$$\text{or } \frac{|\sum_{\langle G_j, C_i \rangle < 0} \langle G_j, C_i \rangle|}{|\sum_{\langle G_j, C_i \rangle > 0} \langle G_j, C_i \rangle|} > D \gg 1, \text{for all } i,$$

then solving MCO is a $\frac{D-1}{D+1}$-approximation to solving MSSO, i.e., $\frac{D-1}{D+1} \leq \frac{MSSO}{MCO} \leq 1$

The proof is included in Appendix **"Proof of Theorem 1"**. Intuitively, we can see that our approximation is perfect when each inner product in (8) is positive, and we have less of a guarantee of the effectiveness when a cluster is less homogeneous in the sign of the inner products between its members and centroid. Indeed, we found that the assumptions in the above theorem hold in most cases (see Appendix **"Supplemental experiments"**). Therefore, due to the closeness of MSSO and MCO, we focus on solving MCO rather than MSSO.

Solving MCO

MCO resembles the K-means clustering objective, so it is promising to solve it with the K-means clustering algorithm. As the first step toward this, MCO is transformed to the following form:

$$\text{MCO} = \max_{\mathcal{C}} \sum_{j=1}^{N} \|G_j\| \sum_{i=1}^{M} \|C_i\| \cdot |\text{cosine}(G_j, C_i)|, \quad (10)$$

which can be regarded as a weighted K-means clustering objective function. Specifically, the norm of each C_i is used for re-weighting the cosine similarity between each training sample j and each cluster centroid i, which is followed by re-weighting the overall similarity of each training sample j to *all* cluster centroids with the norm of G_j. Further details on how to tailor the vanilla K-means clustering algorithm to solve MCO are presented in Appendix **"Supplemental materials on the weighted K-means algorithm"**. After $\mathcal{C} = \{C_i\}_{i=1}^M$ is identified by this weighted K-means algorithm, the samples closest to each cluster centroid are returned as the selected meta samples, D_{m}[2].

Note that in Equation (9), collecting all G_j is very expensive. This is because j is over all training samples which can be very large, and G_j depends on all Θ_t, i.e., the model parameters at all iterations (see Equation (8)).

To address the above efficiency concerns, we firstly propose two methods, i.e., Representation-based clustering method (RBC) and Gradient-based clustering method (GBC) in Section **"Representation-based clustering method (RBC)"** and Section **"Gradient-based clustering method (GBC)"** respectively, for addressing the first concern. We further discuss how to sample from all $\Theta_t (t = 1, 2, \dots)$ in Section **"Sampling model parameters from history"** to handle the second concern.

Representation-based clustering method (RBC) RBC is built upon the assumption that the gradient of the model parameters on the bottom layers (i.e. those layers closer to the input) is less significant than the ones in the last layer. Due to the vanishing gradient problem, this assumption usually holds in practice. As a consequence, we only consider the gradients from the last layer in Equation (9), leading to the following approximations on G_j:

$$G_j = \mathbf{A}_j(\Theta_t) \tilde{\mathbf{x}}_j(\Theta_t)^\top, \quad (11)$$

in which $\tilde{\mathbf{x}}_j(\Theta_t)$ represents the input to the last linear layer in the neural network model produced by the training sample j, while $\mathbf{A}_j(\Theta_t)$ is defined as follows:

$$\mathbf{A}_j(\Theta_t) = \text{softmax}(\Theta_t^{(-1)} \tilde{\mathbf{x}}_j(\Theta_t)) - \text{onehot}(y_j) \quad (12)$$

in which $\Theta_t^{(-1)}$ represents the model parameters in the last layer. The detailed derivation of Equation (11) is included in Appendix **"Derivation of Equation** (11)**"**. Equation (11)-(12) shows that to obtain G_j, only forward passes on the models are needed, which makes this method very efficient.

Gradient-based clustering method (GBC) Unlike RBC, GBC is applicable to general cases where the gradients generated by the bottom neural layers may be significant. To facilitate efficient evaluations of MCO, we importance sample the network layers from the model, such that we can obtain an unbiased estimation of Equation (9). Then G_j is constructed by concatenating the gradients calculated in those sampled layers.

Specifically, first of all, the underlined part of Equation (7) (which is the essential part of Equation (9)) can be rewritten in terms of a sum over the model parameters at each layer $l \in [1, 2, \dots, L]$, i.e.:

$$\begin{aligned} &\langle \nabla_\Theta f_{\text{m},i}(\Theta)|_{\Theta=\Theta_t}, \nabla_\Theta f_j(\Theta)|_{\Theta=\Theta_t} \rangle \\ &= \left[\sum_{l=1}^{L} \langle \nabla_{\Theta^{(l)}} f_{\text{m},i}(\Theta)), \nabla_{\Theta^{(l)}} f_j(\Theta)) \rangle \right]_{\Theta=\Theta_t}, \end{aligned} \quad (13)$$

[2]We notice that other strategies, e.g., (Auvolat et al. 2015), can be employed to solve MCO, which, however, do not perform well and are thus ignored.

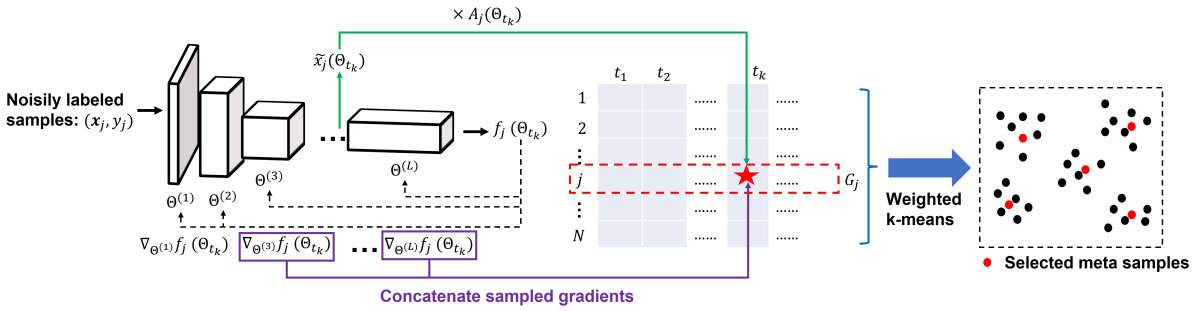

Figure 2: Overview of our methods, RBC and GBC. We use $\rightarrow$ (green colored arrow) and $\rightarrow$ (purple colored arrow) to denote the data flow of RBC and GBC respectively. Specifically, at each sampled time step t_k, for each input training sample $(\mathbf{x}_j, y_j)$, RBC combines its feature vector from the input to the last layer of the model, $\tilde{\mathbf{x}}_j$, and the coefficient, $\mathbf{A}_j(\Theta_{t_k})$ (defined in Equation (12)) while GBC concatenate the gradients from the sampled layers in the model. We then concatenate the above calculated results from all the time steps $t_1, t_2, \ldots$ to compose the input to weighted K-means clustering algorithm, G_j (see the red dotted box), which is then used for determining the meta samples.

in which $\Theta^{(l)}$ represents the model parameters at the l^{th} layer. Then the above formula could be rewritten as follows:

Equation (13)

$$= A \cdot \left[\sum_{l=1}^{L} \frac{A^{(l)}}{A} \langle \frac{\nabla_{\Theta^{(l)}} f_{\mathrm{m},i}(\Theta)}{\sqrt{A^{(l)}}}, \frac{\nabla_{\Theta^{(l)}} f_j(\Theta)}{\sqrt{A^{(l)}}} \rangle \right]_{\Theta=\Theta_t}, \quad (14)$$

in which, $A^{(l)} = \| \frac{1}{N} \sum_{j=1}^{N} \nabla_{\Theta^{(l)}} f_j(\Theta) \|_F^2$ and $A = \sum_{l=1}^{L} A^{(l)}$

Then we can conduct importance sampling (with replacement) on the L innermost sums in Equation (14) for several times (say 5 times)[3], in which the probability of selecting the $l^{th}(l = 1, 2, \ldots, L)$ term is $A^{(l)}/A$. This leads to an unbiased estimation of Equation (14) and significant speed-ups.

Sampling model parameters from history It is worth noting that Θ_t is unknown before we obtain all meta samples (see Equation (2)-(4)), but it is essential for determining the meta samples (see Equation (7)). Therefore, we propose to cache the model parameter $\tilde{\Theta}_t(t = 1, \ldots, T)$ during the training process without any available meta samples, which is regarded as an approximation of Θ_t.

In addition, as mentioned above, G_j depends on the model parameters from all the time steps, which is thus very expensive to evaluate. We uniformly sample several time steps, instead of using all $\tilde{\Theta}_t$, to get an unbiased estimation of MCO.

In the end, we visually present both RBC and GBC equipped with this sampling technique in Figure 2 and include their pseudo-code in Algorithm 3 in Appendix **"Details of the adapted K-means algorithm"**.

Applications

We demonstrate the effectiveness of our methods for two applications, i.e., re-weighting a training set with noisy labels and re-weighting an imbalanced training set. In what follows, we discussed how to tailor RBC and GBC to these two applications.

Re-weighting a Training Set with Noisy Labels

To re-weight a noisily labeled training set, we can select a subset of meta samples from the training set and obtain their clean labels from human annotators. Note that for RBC and GBC, the evaluation of the gradients depend on the clean labels of the meta samples while these clean labels are obtained from human annotators *after* RBC or GBC is invoked. To address this chicken or the egg issue, we observe that if the loss function is the cross-entropy loss, then the sample-wise gradient, $\nabla_\Theta f_j(\Theta)$, can be broken into two parts, the *label-free part* and the *label-dependent part*. Due to the unavailability of the clean labels, we therefore only leverage the *label-free part* as the input to RBC and GBC.

Although we only use the label-free part, in Appendix **"Analysis of the gradient with and without label-free part"** (see Theorem 1), we theoretically analyze under what conditions the label-dependent part is insignificant to determining which cluster each training sample belongs to after the weighted k-means clustering algorithm is invoked. Those conditions are satisfied by a large portion of the training samples through our empirical studies (see Appendix **"Supplemental experiments"**), thus justifying the effectiveness of discarding the label-dependent part.

Re-weighting a Class-Imbalanced Training Set

Unlike the case where the labels are noisy, we assume clean labels in the class-imbalanced training set. As a consequence, we evaluate the sample-wise gradient $\nabla_\Theta f_j(\Theta)$ as a whole rather than removing the label-dependent part from it.

Experiments

We demonstrate the effectiveness of our methods for training deep neural nets on image classification datasets, MNIST (Deng 2012), CIFAR-10 (Krizhevsky, Hinton et al. 2009)

[3] we conduct the importance sampling once for all the samples so that the dimension of G_j is the same among all the samples. Although it is not rigorously correct, the empirical studies show that this approximation could achieve good performance

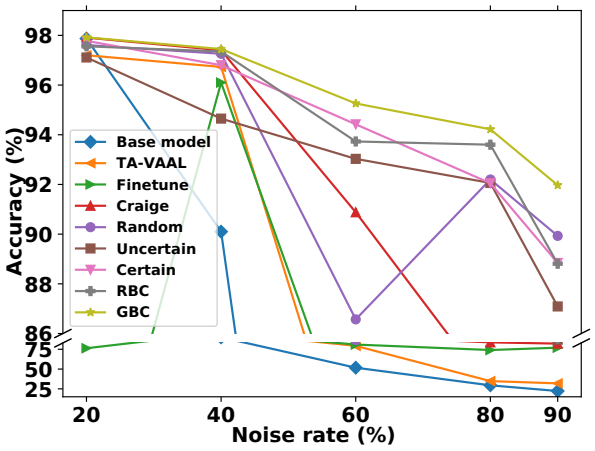

Figure 3: Test performance on MNIST dataset with varied noisy rate

and CIFAR-100 (Krizhevsky, Hinton et al. 2009), and Imagenet-10 (Russakovsky et al. 2015)[4]. By following (Shu et al. 2019) and (Ren et al. 2018), we consider the occurrence of noisy labels and class imbalance respectively on the training set. All the code is publicly available[5].

Experimental Set-Up

For the MNIST dataset, we train a LeNet model (LeCun et al. 1998) and for CIFAR-10, CIFAR-100 and Imagenet-10 dataset, we train a ResNet-34 model (He et al. 2016). All the hyper-paremters are reported in Appendix **"Supplemental experiments"**.

Noisy Label Experiments

We first study how our methods perform in the presence of two types of synthetic label noise, i.e., *uniform noise* and *adversarial noise*, and one type of *real noise*:

- *uniform noise*: all labels can be uniformly flipped at random to any other label with probability $p/100$, in which p is a percent specified by users. This has been explored in (Shu et al. 2019) and (Ren et al. 2018);
- *adversarial noise*: the labels for a subset of samples, chosen at random, are determinisitcally mapped to another label (e.g., selected samples with label 0 are all given label 1). This is meant to simulate an extreme case where the labels are *adversarially* flipped and has been explored in some prior works (e.g., (Li et al. 2022))
- *real noise*: We leverage the real human labeling errors on CIFAR-10 and CIFAR-100 datasets provided by (Wei et al. 2021).

We present the results with one fixed noise rate, $p = 60$, for synthetic labeling noise on MNIST, CIFAR and Imagenet-10 and the effect of varied p is also explored on the MNIST dataset (see Figure 3). Surprisingly, we found that 60% uniform noise only reduces the model accuracy on MNIST and CIFAR-10 by a few percent and cleaning the labeling noise in such settings only marginally improve the performance. We therefore only report the results on MNIST and CIFAR-10 with adversarial noise. We also present the results with real labeling noise on CIFAR-100 but we ignored the same experiments on CIFAR-10 since the real labeling noise also only slightly hurts the model performance. Throughout this experiment, we compare RBC and GBC against the following baseline methods:

- **Random selection** (*Random*): We uniformly at random select meta samples from the training set;
- **Fine-tuning**: We fine-tune the model using only the selected meta samples, selected by Random;
- **Active learning**: We select meta samples using 1) **Uncertainty based selection** (*Uncertain*) (Lewis and Gale 1994) by selecting the *most uncertain* training samples, 2) **Certainty based selection** (*Certain*) by selecting the *most certain* training samples and 3) two state-of-the-art active learning solutions, **Task-Aware Variational Adversarial Active Learning** (*TA-VAAL*) (Kim et al. 2021) and **craige** (*craige*) (Mirzasoleiman, Bilmes, and Leskovec 2020)
- **RBC-k**: We use the original K-means clustering algorithm rather than the weighted version proposed in Section **"Solving MCO"** to determine the meta samples in RBC.

Note that for both of our methods and the above baseline methods, the labels of the selected meta samples are cleaned by human annotators, which is simulated by replacing their noisy labels with ground-truth labels. This thus justifies the use of the perfectly labeled benchmark datasets (rather than real datasets with unreliable labels). As a result, for fair comparison, our methods and the above baseline methods share the same labeling budget, which is set as 20, 50, 200 and 50 for MNIST, CIFAR-10, CIFAR-100 and Imagenet-10 respectively (which includes the labeled samples in the pre-training phase). Recall that both RBC and GBC depend on the model parameters from history. We therefore pre-train the models without any clean labels as the warm-up phase.

Overall performance We present the test accuracy in Table 1[6] after running the meta re-weighting algorithm with meta samples selected by different methods. As indicated by this table, the clustering-based methods, RBC-k, RBC and GBC can significantly outperform other methods in most cases and the performance gains are up to 6% (see the performance difference between GBC and Certain in column "adversarial" of CIFAR-100 dataset). Furthermore, RBC consistently outperforms RBC-K, which suggests the weighted K-means algorithm is capable of identifying a better set of meta samples than the original K-means algorithm. It is also worth noting that with real labeling noise (see the column "real" of CIFAR-100 dataset), both RBC and GBC can outperform all the baseline methods, which leads to at least 1.3% improvement with respect to the state-of-the-art active learning methods (see the comparison between RBC, GBC and craige in the column "real" of CIFAR-100 dataset).

[4]Imagenet-10 is a subset of ImageNet and produced by following (Li et al. 2021)

[5]https://github.com/thuwuyinjun/meta_sample_selections

[6]We report the validation accuracy for Imagenet-10 since the ground-truth labels of test samples are invisible

Dataset	MNIST	CIFAR-10	CIFAR-100			Imagenet-10	
Noise type	adversarial	adversarial	uniform	adversarial	real	uniform	adversarial
Base model	51.74±1.52	40.24±0.39	43.63±2.30	27.15±0.40	49.33	72.22	38.00
Random	85.67±0.90	76.02±2.01	42.30±4.68	45.33±1.70	57.54	93.33	59.77
Certain	81.84±0.89	70.78±5.00	45.95±4.20	47.06±2.10	57.48	91.20	58.22
Uncertain	76.38±0.54	74.45±6.10	36.67±0.20	44.65±0.65	56.23	85.22	51.00
Fine-tuning	53.39±1.22	23.07±7.58	25.28±1.13	24.88±1.10	51.24	70.44	35.67
TA-VAAL	79.31±0.23	61.46±4.65	31.07±2.56	38.79±0.86	44.50	86.34	43.26
craige	92.84±0.14	78.55±1.03	39.85±1.23	44.61±1.21	57.92	88.90	61.00
RBC-K	93.78±0.61	75.71±1.22	49.32±0.35	49.51±0.43	59.24	91.33	54.67
RBC	93.00±1.01	79.20±0.64	49.56±0.53	50.60±1.51	**59.25**	**94.22**	**63.67**
GBC	**94.26±0.24**	**80.88±1.46**	**50.88±1.90**	**53.14±1.33**	**59.25**	94.00	**63.67**

Table 1: Test accuracy on MNIST, CIFAR-10 and CIFAR-100 dataset with synthetic noise (noise rate 60%) and real noise

Dataset	CIFAR-10	CIFAR-100
BaseModel	61.45±0.60	28.14±0.57
Random	65.96±1.74	29.29±0.46
Uncertain	64.46±1.20	28.39±0.21
Certain	66.05±1.19	28.52±0.15
Fine-tuning	60.04±1.69	29.73±0.06
TA-VAAL	61.58±1.21	30.89±1.09
craige	66.60±0.89	29.56±1.46
RBC-K	65.96±1.02	30.77±1.23
RBC	**68.18±1.58**	31.78±1.10
GBC	67.37±1.51	**33.87±0.66**

Table 2: Test performance on imbalanced CIFAR-10 and CIFAR-100 dataset with imbalanced factor 200

Random	RBC	GBC
0.922	**0.958**	0.949

Table 3: The AUC scores of the sample weights on MNIST with noise rate 80%

Efficiency of RBC We also observe a trade-off between performance and speed when comparing GBC and RBC. According to Table 1, GBC performs better than RBC in most cases while the former is slower than the latter (2.5 hours VS 3 mins) to construct G_j. Note that the running time of RBC is negligible in comparison to the running time of the meta re-weighting algorithm, which is around 4 mins per epoch and there are hundreds of epochs in total.

Robustness against varied noise rate As indicated by Figure 3, both RBC and GBC outperform all the baseline methods across all the noise rates and the performance gains become even larger with more samples being noisily labeled (up to 2%). This indicates the robustness of our methods against a varied level of label noise.

Class Imbalance Experiments

For evaluating our method on class imbalanced data, we follow (Cui et al. 2019) to produce the long-tailed CIFAR dataset. Specifically, we down-sample some classes so that the ratio between the number of training samples in the largest class and that in the smallest one (which is denoted the *imbalance factor*) is large. In Table 2, we report the results with imbalance factor 200 on CIFAR-10 and CIFAR-100 dataset. As shown in Table 2, our method, RBC, outperforms all the baseline methods for CIFAR-10 and CIFAR-100 and the performance gain is up to 3.10%.

Studies on the learned sample weights Recall that our methods depend on the assumption that larger updates to the sample weights will more effectively result in the weights of the noisy and clean samples approaching 0 and 1 respectively. We therefore empirically inspect the sample weights learned by Random, RBC and GBC. Specifically, we calculate the AUC between the learned sample weights and the cleanness of the sample labels (1 for clean while 0 for corrupt). We report this quantity for MNIST with 80% noisy labels in Table 3 for the entire training set. As Table 3 shows, the AUC scores of RBC and GBC are higher than that of Random, thus suggesting the capability for RBC and GBC to better distinguish between clean and noisy samples.

Other Experimental Results

Due to the space limit, all other experimental results are presented in Appendix: **"Supplemental experiments"**, including the experiments with the effect of varied number of meta samples, the effect of the number of sampled gradients in RBC and GBC (recall that both approximate Equation (10) through sampling according to Section: **"Solving MCO"**), and some qualitative studies.

Conclusion

In this work, we propose a clustering-based framework for selecting pivotal samples to improve performance of meta re-weighting in the presence of various defects on training data. Based on our theoretical analysis, we show that selecting pivotal samples can be reduced to a weighted K-means algorithm under reasonable assumptions. To efficiently evaluate this algorithm we propose two methods, RBC and GBC, which can balance the computational efficiency and prediction performance. Through empirical studies on noisily labeled and class-imbalanced image classification benchmark datasets, we can demonstrate that our technique could select a better set of pivotal samples for meta re-weighting algorithm than other sample selection techniques, thereby resulting in better model performance.

Acknowledgments

We thank our anonymous reviewers for valuable feedback. This research was supported by grants from DARPA (#FA8750-19-2-0201) and NSF award (IIS-2145644).

References

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; and De Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29.

Auvolat, A.; Chandar, S.; Vincent, P.; Larochelle, H.; and Bengio, Y. 2015. Clustering is efficient for approximate maximum inner product search. *arXiv preprint arXiv:1507.05910*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Chang, H.-S.; Learned-Miller, E.; and McCallum, A. 2017. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30.

Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9268–9277.

Das, S.; Singh, A.; Chatterjee, S.; Bhattacharya, S.; and Bhattacharya, S. 2021. Finding High-Value Training Data Subset Through Differentiable Convex Programming. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 666–681. Springer.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6): 141–142.

Dong, Q.; Gong, S.; and Zhu, X. 2017. Class rectification hard mining for imbalanced deep learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 1851–1860.

Ghorbani, A.; and Zou, J. 2019. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, 2242–2251. PMLR.

Hajij, M.; Zamzmi, G.; Ramamurthy, K. N.; and Saenz, A. G. 2021. Data-Centric AI Requires Rethinking Data Notion. *arXiv preprint arXiv:2110.02491*.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Holtz, C.; Weng, T.-W.; and Mishne, G. 2021. Learning Sample Reweighting for Adversarial Robustness. https://openreview.net/forum?id=7zc05Ua_HOK. Accessed: 2023-02-13.

Hospedales, T. M.; Antoniou, A.; Micaelli, P.; and Storkey, A. J. 2021. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hu, Z.; Tan, B.; Salakhutdinov, R. R.; Mitchell, T. M.; and Xing, E. P. 2019. Learning data manipulation for augmentation and weighting. *Advances in Neural Information Processing Systems*, 32.

Irvin, J.; Rajpurkar, P.; Ko, M.; Yu, Y.; Ciurea-Ilcus, S.; Chute, C.; Marklund, H.; Haghgoo, B.; Ball, R.; Shpanskaya, K.; et al. 2019. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 590–597.

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, 2304–2313. PMLR.

Karimi, D.; Dou, H.; Warfield, S. K.; and Gholipour, A. 2020. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical Image Analysis*, 65: 101759.

Khan, S. H.; Hayat, M.; Bennamoun, M.; Sohel, F. A.; and Togneri, R. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8): 3573–3587.

Killamsetty, K.; Sivasubramanian, D.; Ramakrishnan, G.; and Iyer, R. 2021. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8110–8118.

Kim, K.; Park, D.; Kim, K. I.; and Chun, S. Y. 2021. Task-aware variational adversarial active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8166–8175.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, 1885–1894. PMLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf. Accessed: 2009-04-08.

Kumar, M.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lewis, D. D.; and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, 3–12. Springer.

Li, S.; Xia, X.; Ge, S.; and Liu, T. 2022. Selective-supervised contrastive learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 316–325.

Li, Y.; Hu, P.; Liu, Z.; Peng, D.; Zhou, J. T.; and Peng, X. 2021. Contrastive clustering. In *Proceedings of the AAAI*

Conference on Artificial Intelligence, volume 35, 8547–8555.

Miranda, L. J. 2021. Towards data-centric machine learning: a short review. *Ljvmiranda921. Github. Io*.

Mirzasoleiman, B.; Bilmes, J.; and Leskovec, J. 2020. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, 6950–6960. PMLR.

Polyzotis, N.; and Zaharia, M. 2021. What can Data-Centric AI Learn from Data and ML Engineering? *arXiv preprint arXiv:2112.06439*.

Pruthi, G.; Liu, F.; Kale, S.; and Sundararajan, M. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930.

Ratner, A. J.; Bach, S. H.; Ehrenberg, H. R.; and Ré, C. 2017. Snorkel: Fast training set generation for information extraction. In *Proceedings of the 2017 ACM international conference on management of data*, 1683–1686.

Ren, M.; Zeng, W.; Yang, B.; and Urtasun, R. 2018. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, 4334–4343. PMLR.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B. B.; Chen, X.; and Wang, X. 2021. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9): 1–40.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252.

Shu, J.; Xie, Q.; Yi, L.; Zhao, Q.; Zhou, S.; Xu, Z.; and Meng, D. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32.

Thrun, S.; and Pratt, L. 2012. *Learning to learn*. Springer Science & Business Media.

Van Engelen, J. E.; and Hoos, H. H. 2020. A survey on semi-supervised learning. *Machine Learning*, 109(2): 373–440.

Wang, Y.; Kucukelbir, A.; and Blei, D. M. 2017. Robust probabilistic modeling with bayesian data reweighting. In *International Conference on Machine Learning*, 3646–3655. PMLR.

Wei, J.; Zhu, Z.; Cheng, H.; Liu, T.; Niu, G.; and Liu, Y. 2021. Learning with Noisy Labels Revisited: A Study Using Real-World Human Annotations. In *International Conference on Learning Representations*.

Wu, Y.; Weimer, J.; and Davidson, S. B. 2021. CHEF: a cheap and fast pipeline for iteratively cleaning label uncertainties. *Proceedings of the VLDB Endowment*, 14(11): 2410–2418.

Yoon, J.; Arik, S.; and Pfister, T. 2020. Data valuation using reinforcement learning. In *International Conference on Machine Learning*, 10842–10851. PMLR.