ENHANCING DATA ACCURACY: INTEGRATION OF CONFIDENT LEARNING AND CROWDLAB

JUN CHA

December 2023

1. Introduction

This paper extends the discussion from the last presentation on CROWDLAB [1] and Confident Learning [2]. The previous presentation highlighted the prevalence of label errors in well-known public datasets such as MNIST, ImageNet, and CIFAR. These errors, whether in training or evaluation datasets, can significantly affect model performance. In benchmarking scenarios, label errors can mislead assessments of a model's true performance. For example, although ResNet-50 typically outperforms ResNet-18 when both are trained on the ImageNet dataset, due to its higher capacity, label corrections in the evaluation set revealed that ResNet-18 actually outperforms ResNet-50. This phenomenon could be attributed to the higher capacity model's tendency to overfit mislabeled examples. Correcting label errors is particularly relevant in real-world applications due to its lower computational requirements and potential to outperform model-centric approaches.

With the exponential growth in data availability, the concept of crowdsourcing is rapidly gaining prominence. While the use of AI to generate labels and annotations is an emerging field, the prioritization of human intelligence, often through 'ghost workers,' remains crucial for enhancing the accuracy of data annotations. Crowdsourcing, which typically involves multiple annotators labeling the same examples to enhance label confidence, often encounters discrepancies due to conceptual misunderstandings, unrecognized instances, and typographical errors, among others. To address these real-world labeling challenges, efficient algorithms are needed. The CROWDLAB algorithm, which considers both classifier and annotator inputs could be a solution for these labeling inconsistencies.

Confident Learning and CROWDLAB represent significant contributions to the emerging field of Data-Centric AI. However, as with any rapidly evolving area, there are still notable limitations to be addressed. This paper aims to delve deeper into the implementation of these algorithms and their role in enhancing model performance through integration. Additionally, it will explore potential directions for further development in this field.

2. Methods

2.1. Confident Learning (CL).

Before implementing Confident Learning, it is important to acknowledge the types of label issues in a real-world scenario. Label errors can be categorized into four types: correctable, multi-label, non-agreement, and neither. A multi-label error occurs when an example contains multiple possible labels (e.g., an image depicting a

cat in a box). Non-agreement error arise in cases where an example is ambiguous or uncertain (e.g., a handwritten '4' that might be identified as '9'). 'Neither' category applies to examples that are neither multi-label errors nor non-agreement errors (e.g., an extremely low-quality picture). The correctable error, the primary focus of Confident Learning, happens when an example is truly mislabeled into a different class. However, it is important to note that Confident Learning does not detect random label errors.

There is one key assumption made when identifying label errors. The assumption is that labels are flipped based on an unknown transition matrix $p(\tilde{y}|y^*;\boldsymbol{x}) = p(\tilde{y}|y^*)$, where $\tilde{y}$ and y^* denote the observed, noisy label and the unobserved, latent, correct label, respectively. This assumption requires that the occurrence of labels errors depends on pairwise noise rates between classes, not on the data $\boldsymbol{x}$. Confident Learning algorithm focuses on finding *class-conditional* label noise, which depends on the class rather than on random flipping errors. This assumption is practical and valid for most datasets, as certain mislabelings are more probable than others (e.g., a fox is more likely to be mislabeled as a dog than as a missile). Moreover, this premise has been supported by previous research and offers a realistic approach to addressing real-world problems.

Confident Learning is based on the principles of **pruning** noisy data (as opposed to fixing label errors or modifying the loss function), using *soft-pruning*, **counting** to estimate noise, avoiding error-propagation in learned model weights from reweighting the loss with imperfect predicted probabilities, and **ranking** examples to train with confidence (as opposed to weighting by exact probabilities).

CL framework requires two inputs: out-of-sample predicted probabilities $\hat{p}(\tilde{y};\boldsymbol{x},\boldsymbol{\theta})$ for any given model $\boldsymbol{\theta}$, and noisy labels $\tilde{y}$, from noisy data $\boldsymbol{X} := (\boldsymbol{x},\tilde{y})^n \in (\mathbb{R}^d,[m])^n$, where $[m]$ denotes the set of m unique class labels. To clarify the notation, in matrix form, the $n \times m$ matrix of out-of-sample predicted probabilities is $\hat{\boldsymbol{P}} := \hat{p}(\tilde{y};\boldsymbol{x},\boldsymbol{\theta})$, where n denotes the number of examples.

One of the key ideas of CL is to establish thresholds as proxies for the machine's self-confidence, on average, for each class j. Calculating the average out-of-sample predicted probabilities for each class j allows the identification of label errors, particularly when the given $\hat{p}(\tilde{y}=j)$ is less than the threshold for class j. This approach also accounts for overconfident models on certain classes, which will be discussed later in this paper.

$$t_j = \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y}=j;\boldsymbol{x},\boldsymbol{\theta})$$

The main algorithm of CL, as previously explained, requires two matrices: $\boldsymbol{C}_{\tilde{y}=i,y^*=j} \in \mathbb{N}_{\geq 0}{}^{m \times m}$ and an $m \times m$ matrix $\tilde{\boldsymbol{Q}}_{\tilde{y},y^*}$. The first matrix, $\boldsymbol{C}_{\tilde{y},y^*}$, represents the confident joint count in each class, $i \in [m]$ and $j \in [m]$, and the second matrix, $\tilde{\boldsymbol{Q}}_{\tilde{y},y^*}$, is the normalized joint distribution of $\boldsymbol{C}_{\tilde{y},y^*}$. This procedure comprises three steps: (1) estimate $\tilde{\boldsymbol{Q}}_{\tilde{y},y^*}$ to characterize class-conditional label noise, (2) **prune** noisy examples by **ranking**, (3) train with the errors removed. The confident joint is calculated as follows:

$$\boldsymbol{C}_{\tilde{y}=i,y^*=j}[i][j] := |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}| \text{ where}$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} := \left\{ \boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j;\boldsymbol{x},\boldsymbol{\theta}) \geq t_j, \quad j = \underset{l \in [m]: \hat{p}(\tilde{y}=l;\boldsymbol{x},\boldsymbol{\theta}) \geq t_l}{\operatorname{argmax}} \hat{p}(\tilde{y}=l;\boldsymbol{x},\boldsymbol{\theta}) \right\}$$

The confident joint yields the total count of mislabeled examples for each class, $\boldsymbol{C}_{\tilde{y}=i,y^*=j}$. The count of correct label class j examples being mislabeled as class i is represented on the off-diagonal of the matrix $\boldsymbol{C}_{\tilde{y},y^*}$, while the count of correctly labeled examples is on the diagonal. This is one of the key principles of the algorithm, **counting**. The complexity of this algorithm is $\mathcal{O}(m^2 + nm)$.

Following the estimation of $\tilde{\boldsymbol{Q}}_{\tilde{y},y^*}$ and $\boldsymbol{C}_{\tilde{y},y^*}$, any **rank** and **prune** approach can be used to clean the data. One primary approach to ranking noisy labels is using the model's self-confidence. The sets of examples counted in the off-diagonals of $\boldsymbol{C}_{\tilde{y},y^*}$, estimated as noisy labels, are ranked by $\hat{p}(\tilde{y} = i; \boldsymbol{x}, \boldsymbol{\theta})$. The lower the out-of-sample predicted probability, the higher the likelihood of an example having label errors. Additionally, using the normalized margin $|\hat{p}(\tilde{y} = i) - \hat{p}(\tilde{y} = j)|$, where $\{j \in [m] : j \neq i\}$, to rank examples can be practical.

Previously, it was mentioned that CL is robust to over-confident models. If a model exhibits either under-confidence or over-confidence bias for certain classes j, we can rewrite the out-of-sample predicted probability as $\hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) + \epsilon_j$. The new threshold for class $j \in [m]$ is defined as:

$$\begin{aligned} t_j^{\epsilon_j} &= \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) + \epsilon_j \\ &= t_j + \epsilon_j \end{aligned}$$

Consequently, the confident joint matrix is represented as:

$$\begin{aligned} \boldsymbol{C}^{\epsilon_j}_{\tilde{y}=i,y^*=j} &= |\{\boldsymbol{x} : \boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i}, \hat{p}(\tilde{y} = j|\boldsymbol{x}) + \epsilon_j \geq t_j + \epsilon_j\}| \\ &= |\{\boldsymbol{x} : \boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i}, \hat{p}(\tilde{y} = j|\boldsymbol{x}) \geq t_j\}| \\ &= \boldsymbol{C}_{\tilde{y}=i,y^*=j} \end{aligned}$$

This demonstrates the robustness of CL against per-class errors.

2.2. CROWDLAB (Classifier Refinement Of croWDsourced LABels).

Curating and creating a clean dataset is as important as identifying label errors in a dataset, as discussed in the previous section. In this section, we will explore effective methods for curating datasets labeled by multiple annotators, a practice also known as Crowdsourcing.

Consider a dataset of sampled pairs (X, Y), (feature, label), comprised of n examples, K classes, and m annotators. The example X_i denotes the ith example, and $Y_i \in [K]$, where $[K]$ denotes the set of K unique classes, is the class label. The subset of examples labeled by annotator j is denoted by $\mathcal{I}_j := \{i \in [n] : Y_{ij} \neq \emptyset\}$, and $\mathcal{I}_{j,+} := \{i \in \mathcal{I}_j : |\mathcal{J}_i| > 1\}$ is the subset of examples that are labeled by more than one annotator. For $j \in [m]$, $\mathcal{A}_j$ denotes the jth annotator, and Y_{ij} is the class that annotator j chose for X_i. The subset of annotators that labeled example i is denoted by $\mathcal{J}_i := \{j : Y_{ij} \neq \emptyset\}$. We assume that the consensus label, $\hat{Y}_i$, is the best estimate label for X_i. We use the Iverson bracket, denoted as $[P]$, which equals 1 if the proposition P is true, and 0 if it's false. This notation might lead to confusion; however, in this paper, the Iverson bracket $[P]$ is always used with a logical proposition P. Otherwise, $[x]$ denotes a set of unique x in $\{K, m, i\}$. There are three main components to consider in Crowdsourcing: (1) determining the consensus label, $\hat{Y}_i$, (2) measuring the confidence level of the consensus label, $\hat{p}(Y_i \mid X_i, \{Y_{ij}\})$, and (3) quantifying annotator quality based on Y_{ij} and $\hat{Y}_i$.

Instead of relying solely on annotators or a classifier to label the dataset X, the CROWDLAB algorithm considers both annotators and the classifier when determining the consensus label, $\hat{Y}_i$. The algorithm also employs weights for both the annotator and classifier, adjusting these weights according to the accuracy of either the annotator's or the model's performance for a certain class Y_i. This approach allows CROWDLAB to outperform other algorithms, such as Majority-vote, inter-annotator agreements, and Dawid-Skene, by leveraging key concepts from these algorithms and making further improvements.

The CROWDLAB algorithm estimates each example as follows:

$$\hat{p}_{CR}(Y_i \mid X_i, \{Y_{ij}\}) = \frac{w_{\mathcal{M}} \cdot \hat{p}_{\mathcal{M}}(Y_i \mid X_i) + \sum_{j \in \mathcal{J}_i} w_j \cdot \hat{p}_{\mathcal{A}_j}(Y_i \mid \{Y_{ij}\})}{w_{\mathcal{M}} + \sum_{j \in \mathcal{J}_i} w_j} \tag{1}$$

Here, $w_{\mathcal{M}}$ and $\hat{p}_{\mathcal{M}} \in \mathbb{R}^K$ represent the model weight and predicted probability of the classifier for the label Y_i, given data X_i, respectively. w_j and $\hat{p}_{\mathcal{A}_j} \in \mathbb{R}^K$ denote the annotator weight and a likelihood vector treating each annotator's label as a probabilistic prediction.

Prior to calculating the weights, we need an initial noisy consensus label, $\hat{Y}_i$, using the majority-vote algorithm: $\hat{Y}_i = \text{argmax}_k(\sum_{j \in \mathcal{J}_i}[Y_{ij} = k])$. This algorithm tends to face tie-breaking problems when $|k| \neq 1$. Therefore, we provide four tie-breakers in the algorithm: 1) classifier's predicted probabilities, if available, 2) empirical class frequencies, 3) creating/using annotator qualities, and 4) random selection of the class. For tie-breaker 1, we pick the class with the highest model's probabilities, which indicates the classifier's self-confidence. The idea of using empirical class frequencies is derived from the goal of preventing larger class imbalance by selecting a minority class. Creating or using annotator qualities allows us to weigh each annotator's contribution differently. If annotator quality, a_j, is not available, then we calculate the annotator quality using the existing consensus label: $a_j = \frac{1}{N}\sum_{i \in \mathcal{I}_j}[Y_{ij} = \hat{Y}_i]$. In case of a tie, we select a random class to break the tie.

Knowing the initial consensus label allows us to calculate the weights, $w_{\mathcal{M}}$ and w_j; the steps are as follows:

$$w_j = 1 - \frac{1 - s_j}{1 - A_{MLC}} \tag{2}$$

$$w_{\mathcal{M}} = \left(1 - \frac{1 - A_{\mathcal{M}}}{1 - A_{MLC}}\right) \cdot \sqrt{\frac{1}{n} \cdot \sum_i |\mathcal{J}_i|} \tag{3}$$

Our baseline for these weights is A_{MLC}, the estimated accuracy of the most common overall class. Therefore, the weights for both annotators and models are baseline-normalized versions of each annotator's overall agreement with other annotators and the overall accuracy of the model. Additionally, this algorithm allows for greater weighting of the model for examples that are labeled by fewer annotators.

To calculate the estimated accuracy of the most common overall class, A_{MLC}, we first need to define Y_{MLC}: the overall most labeled class across all examples' annotations, such that $Y_{MLC} := \text{argmax}_k \sum_{ij}[Y_{ij} = k]$. Once we have selected the most common overall class, Y_{MLC}, we can estimate the accuracy of this class, A_{MLC}, compared to our consensus label, $\hat{Y}_i$.

$$A_{MLC} = \frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} \left[Y_{MLC} = \hat{Y}_i \right]$$
$$\text{where} \quad \mathcal{I}_+ = \{i \in [n] : |\mathcal{J}_i| > 1\}$$

Following a similar concept, we can calculate the model accuracy, $A_{\mathcal{M}}$. In this formula, $Y_{i,\mathcal{M}} := \text{argmax}_k \, \hat{p}_{\mathcal{M}}(Y_i = k \mid X_i) \in [K]$ is the class predicted by our model for X_i. The calculation of model accuracy is as follows:

$$A_{\mathcal{M}} = \frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} \left[Y_{i,\mathcal{M}} = \hat{Y}_i \right]$$

For annotator accuracy, s_j represents annotator j's agreement with other annotators who labeled the same examples:

$$s_j = \frac{\sum_{i \in \mathcal{I}_j} \sum_{\ell \in \mathcal{J}_i, \ell \neq j} [Y_{ij} = Y_{i\ell}]}{\sum_{i \in \mathcal{I}_j} (|\mathcal{J}_i| - 1)}$$

Following these steps allows us to calculate the weights as shown in (2) and (3). To calculate (1), the per-annotator predicted class probability vector, $\hat{p}_{\mathcal{A}_j}$, needs to be calculated.

$$\hat{p}_{\mathcal{A}_j}(Y_i \mid \{Y_{ij}\}) = \begin{cases} P & \text{when } Y_{ij} = k \\ \frac{1-P}{K-1} & \text{when } Y_{ij} \neq k \end{cases}$$

Here, P denotes the average annotator agreement across examples with more than one annotation. P can be calculated as follows:

$$P = \frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} \frac{1}{|\mathcal{J}_i|} \sum_{j \in \mathcal{J}_i} \left[Y_{ij} = \widehat{Y}_i \right]$$

This likelihood is shared across annotators. P denotes a simple estimate of the accuracy of labels from a typical annotator. Using these provided formulas allows us to calculate the formula from (1). As we obtain the post-predicted probabilities, we are able to achieve a better consensus label $\hat{Y}_i$.

2.2.1. ***Annotator Quality***.

Beyond finding the consensus label, quantifying annotator quality based on their annotations could further improve crowdsourcing method. To rank annotators more precisely based on their annotations requires a more practical algorithm. Therefore, the annotator quality function, $a_j \in [0, 1]$, can be rewritten as:

$$a_j = \bar{w} Q_j + (1 - \bar{w}) A_j \tag{4}$$

This function represents a quality score for each annotator, using the average label quality score, Q_j, and the annotator's agreement with consensus labels, A_j. Each Q_j and A_j can be calculated as follows:

$$A_j = \frac{1}{|\mathcal{I}_{j,+}|} \sum_{i \in \mathcal{I}_{j,+}} \left[Y_{ij} = \hat{Y}_i \right]$$
$$Q_j = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \hat{p}_{CR}(Y_i \mid X_i, \{Y_{ij}\})$$

By using both the agreement with the consensus label and average label qualities, this allows differentiation among annotators who share the same agreement rate by using the average label qualities (i.e., favoring the label quality of one annotator that is higher among a group of annotators sharing the same agreement rate). Also, the weight $\bar{w}$ is shared across all annotators.

$$\bar{w} = \frac{w_{\mathcal{M}}}{w_{\mathcal{M}} + w_0}$$
$$\text{where} \quad w_0 = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} w_j \cdot |\mathcal{J}_i|$$

Here, $w_{\mathcal{M}}$ and w_j were computed in the previous section, and n, m denote the number of examples and the number of annotators, respectively. Determining the annotator quality a_j for annotator j using this approach allows for more precise ranking of annotators and leaves room for improvements when CROWDLAB algorithm is iterated using this annotator quality to obtain a better initial consensus label.

2.3. **Integration: CL + CROWDLAB.**

In this section, we will discuss the potential improvements for the CROWDLAB algorithm. The CROWDLAB algorithm can be enhanced when the classifier is trained with datasets that have fewer label errors. To obtain a better predicted probability, $\hat{p}_{\mathcal{M}}(Y_i \mid X_i)$, and subsequently a better consensus label, $\hat{Y}$, the following steps are proposed:

(1) Train the initial predicted probability of a model $\boldsymbol{\theta}$, with noisy data $\boldsymbol{X} := (\boldsymbol{x}, \tilde{y})^n$, where $\tilde{y}$ denotes our initial majority-vote $\hat{y}$ consensus label with tie breakers (as discussed in section 2.2).
(2) Using the out-of-sample predicted probabilities $\hat{p}(\tilde{y}; \boldsymbol{x}, \boldsymbol{\theta})$ and noisy labels $\tilde{y}$, apply the CL algorithm, as shown in section 2.1, to create cleaner data.
(3) With the less noisy dataset from step 2, re-train the classifier $\boldsymbol{\theta}$.
(4) Iterate the CROWDLAB algorithm with the more accurate dataset.

This process enhances the model's performance and curates a more robust dataset with fewer label errors when the dataset is labeled by multiple annotators. The results of this methodology are shown in section 3.

However, there are shortcomings to this approach. As Confident Learning involves soft-pruning, if the size of the dataset is limited, pruning a small subset may not significantly impact the model's performance [3]. Moreover, filtering out certain annotations could further impact the quantification of each annotator, as it does not account for their mistakes. This could be addressed by initially running CROWDLAB to determine their initial annotator quality, which could provide insight into each annotator's mistake rates. However, this approach might lead to higher computational demands, and we did not implement this methodology in this paper.

3. Experiments

In this section, we briefly discuss the implementation of our algorithms and their results. Due to hardware limitations on a local computer, running extensive datasets was not feasible. Therefore, we generated a theoretical dataset with parameters $K = 3$, $n = 5000$, and $m = 50$, accompanied by a correctly labeled vector for testing

purposes. For the model $\boldsymbol{\theta}$, we employed a simple KNN classifier with 10-fold cross-validation. The methods for dataset creation and model implementation are adapted from the introductory DCAI course at MIT IAP 2023. Additionally, we compared our implementation with the open-source CROWDLAB library from CLEANLAB to assess the efficacy of our approach.

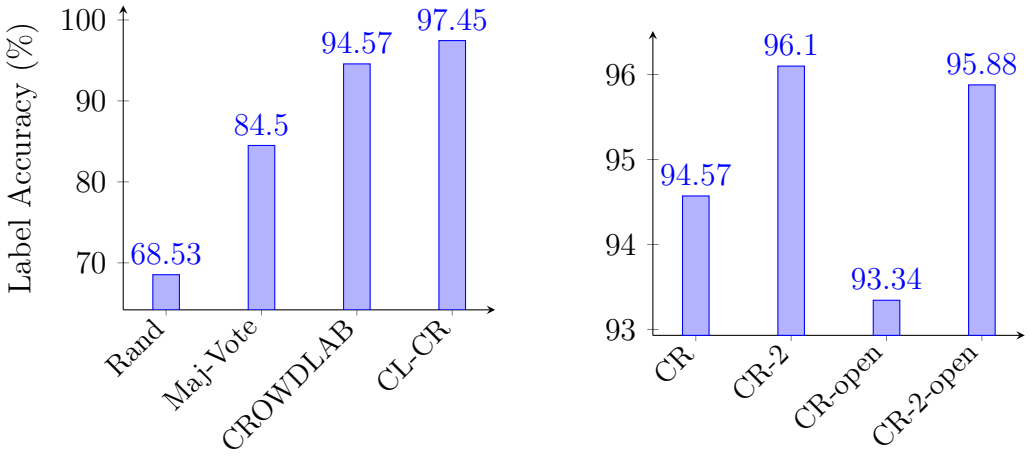

FIGURE 1. Left: Label accuracy for various methods. Right: CROWDLAB accuracy comparison between personal implementation and open-source version. CR-2 denotes CROWDLAB iterated twice.

Figure 1 demonstrates that the CROWDLAB algorithm, when combined with Confident Learning, outperforms standalone CROWDLAB implementation. Moreover, the performance of CROWDLAB using open-source functions shows minimal variance compared to our implementation. Iterating CROWDLAB indicates a performance enhancement, suggesting that improving the initial consensus label $\hat{Y}$ can lead to a more accurate final consensus label.

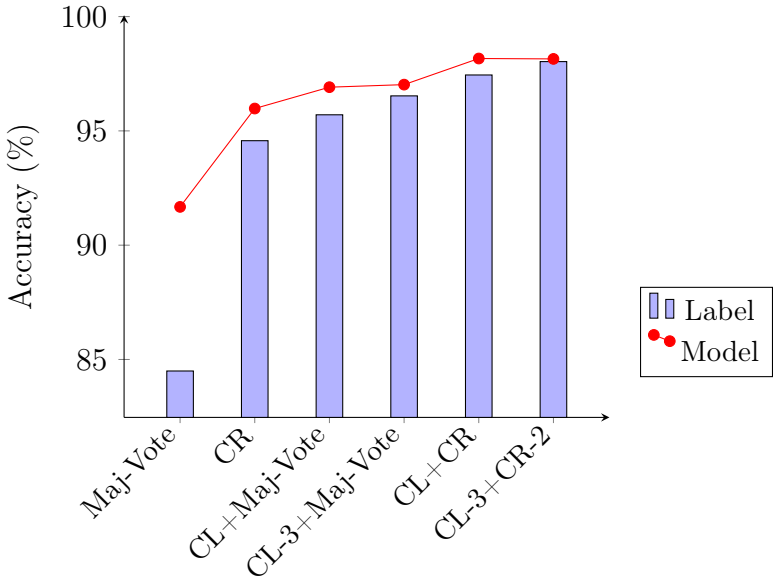

FIGURE 2. Comparative accuracy of methods for labels and models.

Figure 2 reveals that iterative application of Confident Learning to refine noisy labels, combined with CROWDLAB iterations, significantly enhances the accuracy of the consensus label. Given the simplicity of our model-a K-NN Classifier with 10-fold CV-we noted a decline in actual model performance as label accuracy increased. However, this also indicates that enhancing consensus label accuracy can substantially improve overall model performance.

4. Discussion

In the previous section, we observed an increase in model accuracy concurrent with the improvement in label accuracy, indicating that cleaner and more robust input data contributes significantly to model performance. Furthermore, iterative applications of both the Confident Learning (CL) and CROWDLAB algorithms demonstrated notable enhancements in label accuracy and model performance. While this study did not observe divergence in iterating these two algorithms, it is anticipated that both CL and CROWDLAB outputs may diverge upon further iterations. This data-centric approach, in contrast to more computationally intensive model-centric methods, can substantially improve model performance without excessive computational demands.

Confident Learning effectively identifies class-conditional label errors by establishing class thresholds, while CROWDLAB enhances dataset robustness by considering both annotators and classifier outputs, thus drawing a line between human and computer reliability. The presence of label errors in training or benchmark datasets can significantly distort the perceived accuracy of models. Implementing these algorithms enables a more accurate assessment of true model performance by uncovering hidden label errors.

4.1. **Limitations.** As a newly emerging concept, data-centric AI approaches, including Confident Learning, face certain limitations. Primarily, CL targets class-conditional errors and may overlook random errors, necessitating human intervention or additional algorithms for detection. A significant oversight of CL is its inability to recognize situations where one or more true labels are completely missed, a scenario more prevalent in multi-label settings and complex tasks like segmentation or detection. Thus, while CL is a step forward, it is not yet a comprehensive solution.

Conversely, CROWDLAB's effectiveness is closely tied to the underlying model's performance. Although this algorithm can utilize predicted probabilities from any classifier, its efficacy is contingent on the model's initial accuracy. For instance, in tie-breaking scenarios for initial consensus labeling, the model's predicted probability is often the primary criterion. If the model's outputs are inherently noisy, the algorithm may disproportionately rely on these outputs, irrespective of individual annotator accuracy.

4.2. **Further Directions.** Acknowledging the limitations outlined, future work should focus on enhancing Confident Learning's robustness, particularly in multi-labeled data scenarios. While it currently detects errors in such datasets, there is room for increased efficiency. Additionally, in crowdsourcing contexts, integrating algorithms like Active Learning could streamline the annotation process, reducing the reliance on human labor and optimizing resource utilization.

References

[1] Hui Wen Goh, Ulyana Tkachenko, and Jonas Mueller. Crowdlab: Supervised learning to infer consensus labels and quality scores for data with multiple annotators, 2023.

[2] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels, 2022.

[3] Arun Thundyill Saseendran, Lovish Setia, Viren Chhabria, Debrup Chakraborty, and Aneek Barman Roy. Impact of data pruning on machine learning algorithm performance, 2019.