Learning with Noisy Labels by Adaptive Gradient-Based Outlier Removal

Anastasiia Sedova[1,2⋆](✉), Lena Zellinger[1⋆], and Benjamin Roth[1,3]

[1]Research Group Data Mining and Machine Learning, University of Vienna, Austria
[2]UniVie Doctoral School Computer Science, University of Vienna, Austria
[3]Faculty of Philological and Cultural Studies, University of Vienna, Austria
{anastasiia.sedova, lena.zellinger, benjamin.roth}@univie.ac.at

Abstract. An accurate and substantial dataset is essential for training a reliable and well-performing model. However, even manually annotated datasets contain label errors, not to mention automatically labeled ones. Previous methods for label denoising have primarily focused on detecting outliers and their permanent removal – a process that is likely to over- or underfilter the dataset. In this work, we propose AGRA: a new method for learning with noisy labels by using Adaptive GRAdient-based outlier removal[1]. Instead of cleaning the dataset *prior* to model training, the dataset is dynamically adjusted *during* the training process. By comparing the aggregated gradient of a batch of samples and an individual example gradient, our method dynamically decides whether a corresponding example is helpful for the model at this point or is counter-productive and should be left out for the current update. Extensive evaluation on several datasets demonstrates AGRA's effectiveness, while a comprehensive results analysis supports our initial hypothesis: permanent hard outlier removal is not always what model benefits the most from.

1 Introduction

The quality and effectiveness of a trained model heavily depend on the quality and quantity of the training data. However, ensuring *consistent* quality in automatic or human annotations can be challenging, especially when those annotations are produced under resource constraints or for large amounts of data. As a result, real-world datasets often contain annotation errors, or "label noise", which can harm the model's overall quality.

Previous data-cleaning methods for noise reduction have attempted to improve the data quality by identifying and removing "noisy", i.e., mislabeled samples before model training. Some approaches detect noisy samples based on the disagreement between assigned and predicted labels in a cross-validation setting [36, 50], while others leverage knowledge transferred from a teacher model trained on clean data [33]. Such approaches typically rely on certain assumptions

⋆ Equal contribution
[1] We share our code at: https://github.com/anasedova/AGRA

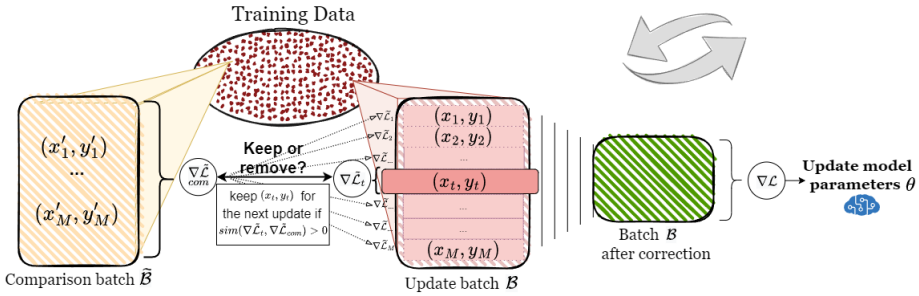

Fig. 1: AGRA method for learning with noisy data. Each sample in the *update batch* is decided to be either kept for further model training or removed depending on the similarity of its gradient to the aggregated gradient of the *comparison batch* sampled from the same data.

regarding the label noise, for instance, that the noise follows some particular distribution, is symmetric [22, 5], or class-conditioned [36]. However, the *true* data-generating process and noise level are usually unknown, and these methods easily over- or under-filter the data.

Another subtle problem arises from the *static* nature of these methods, as they do not address the cases when problematic training samples for one model actually be beneficial for another. Take the hypothetical – wrongly labeled – movie review:

"The movie was by no means great." – `POSITIVE`

Despite the incorrect label, a model that does not know anything about sentiment prediction still might learn the useful association between the word *great* and the class `POSITIVE`. Therefore, this sample could be a valuable contribution to the training process. On the other hand, the same sample might be confusing and deteriorating for the same model at a different training stage, when it already has learned to distinguish subtle language phenomena like negation.

In this paper, we reconsider the original motivation behind noise reduction: instead of tracing the outliers, we focus on *obtaining a model that remains unaffected by inconsistent and noisy samples*. To achieve that, we suggest to dynamically adjust the training set during the training process instead of denoising it beforehand. This idea is realized in our new method **AGRA** - a method for **A**daptive **GRA**dient-based outlier removal (see Figure 1). AGRA does not rely on some static, model-independent, implied properties but leverages gradients to measure the sample-specific impact on the current model. During classifier training, AGRA decides for each sample whether it is *useful* or not for a model

at the current training stage by comparing its gradient with an accumulated gradient of another batch that is independently sampled from the same dataset. Depending on the state of the classifier and the experimental setup, the sample is either used in the model update, excluded from the update, or assigned to an alternative label. Importantly, the effect of the sample may change in the next epoch when the model state has changed. Apart from that, we experimented with different loss functions and adapted an F_1-based loss function which optimizes the model directly towards the F_1 performance metric. Extensive experiments demonstrate the effectiveness of our method and show that the *correctness* of a training sample (as measured by manual annotation) is not the same as its *usefulness* for the training process. AGRA reliably detects the latter in a trade-off with the former, which is crucial for the performance of the trained classifier.

Overall, our main contributions are the following:

- We propose a new gradient-based method for adaptive outlier removal, AGRA, which dynamically identifies unusual and potentially harmful training samples during the learning process and corrects or removes them. Since in our setting labeling errors are unknown at training time, AGRA uses the detrimental effect on the model w.r.t. to a comparison batch as a proxy to harmfulness.
- We analyze the effect of cross-entropy- and F_1-based loss functions for computing the compared gradients and show that utilizing the F_1 loss can improve performance on multiple datasets.
- We demonstrate the effectiveness of our method on several benchmark datasets where our method outperforms other denoising methods trained in an analogous evaluation setup.

2 Related Work

The high demand for large-scale labeled training data to train a stable classifier forces researchers and practitioners to look for more feasible solutions than relying on domain experts to annotate the data [40, 47, 37]. The cost of such approaches is usually the annotation quality, and the resulting datasets often contain mislabeled samples. Moreover, label noise can also be detected in expert annotations due to different factors in the data collection process [42, 14]. As a result, even widely-used datasets may contain incorrect annotations [36], emphasizing the necessity for methods that enable the learning of reliable models despite the presence of label noise.

Learning with Noisy Labels. There are multiple general strategies for handling potential label noise. *Data-cleaning* approaches separate the denoising process and the training of the final model: likely mislabeled samples are first identified and removed or corrected, and then the final model is trained on the cleaned dataset [22, 34]. The INCV algorithm [10] iteratively estimates the joint distribution between the true labels and the noisy labels using out-of-sample model

outputs obtained by cross-validation. On the basis of the estimated joint distribution, the number of labeling errors is gauged and likely mislabeled samples are removed. Cleanlab [36] estimates the confident joint between true and noisy labels relying on the assumption of class-dependent noise. Instead of defining a denoising system that would clean the data *before* the classifier training, AGRA joins the denoising and training into a single process where denoising happens *during* the classifier training. Moreover, AGRA does not make any assumption regarding the label noise distribution. Other approaches, commonly referred to as *model-centric*, focus on modifying the model architecture or the loss function to facilitate learning with noisy data. Wang et al. [49] add a noise-tolerant term to the cross-entropy loss, Ziyin et al. [35] propose a gambler's loss function, and Sukhbaatar et al. [45] add an additional noise layer to convolutional neural networks. Other authors have explored more intricate training strategies for learning with noisy annotations: for instance, Li, Socher, and Hoi [30] leverage ensemble methods, and Li et al. [31] exploit meta-learning techniques. In contrast to these approaches, AGRA does not put any restrictions on the loss function and does not alter the model architecture.

Outlier Detection. Outlier detection is crucial in many real-world applications, such as fraud detection and health diagnosis [48]. There are several general approaches for identifying outliers [48]: *distance-based* methods consider a sample an outlier if it is far away from its nearest neighbors [27, 17], *density-based* approaches declare samples in low-density regions as outliers [9, 7], *clustering-based* strategies identify samples that are not associated with a large cluster [12, 2]. AGRA defines outliers in terms of their utility at the current training step and aims at removing the ones that harm the current model.

Weak Supervision. To reduce the need for manual annotations, datasets can be labeled by automated processes, commonly referred to as *weak supervision* [29, 41, 13]. In the weakly supervised setting, expert knowledge and intuition are formalized into a set of rules, or labeling functions [37], which annotate the training samples with weak, potentially noisy labels. Various approaches to denoise the weakly supervised data include leveraging labeling functions aggregation techniques [37, 38], learning via user feedback and manual correction [21], separately modeling labeling-function-specific and task-specific information in the latent space [44], or utilizing a small set of manually annotated data in addition to the weakly supervised samples [25]. In contrast to these methods, AGRA is not restricted to the weakly supervised setting (although it can be applied for it, even if the labeling functions are not accessible); instead, it is applicable to any dataset that contains noisy labels, regardless of the labeling process used.

Gradient-based Approaches. AGRA is based on gradient comparisons, which were studied before in different contexts [53, 43, 51]. For instance, Zhao et al. [53] explored gradient matching for generating artificial data points that represent a condensed version of the original dataset. Unlike their approach, AGRA does not create any new data instances but adjusts how the already provided ones

are used during training. Shi et al. [43] leverage gradient matching for domain generalization. AGRA, on the other hand, tackles a different problem and does not explicitly assume distribution shifts in the data.

3 AGRA: Adaptive Gradient-Based Outlier Removal

The main goal of AGRA is detecting the instances that would harm the model in the current training stage and filter them out or assign them to another class before the update. In order to decide which samples are potentially harmful, the model gradients for each sample in the *update batch* (i.e., the batch used during the training process for the model update) are compared one by one with an aggregated gradient from another batch sampled from the same data (*comparison batch*). Informally, such an aggregated *comparison gradient* could be seen as an expected weight change on mostly clean data, assuming that the overall noise rate is not too high. If the *update gradient* of a sample from the update batch and the comparison gradient point in opposing directions, this could be an indication that the sample is harmful to the training process at this stage. We refer to such samples as *outliers* since they may have a negative impact on the current model update, even though they are not necessarily mislabeled. Each identified outlier is either removed from the update batch to prevent its influence on the weight update or reassigned to another class if doing so results in a higher, positive gradient similarity.

Unlike common denoising approaches that clean and fix the training dataset for the training process, AGRA does not make any decisions about removing or relabeling the samples *before* training the model. Instead, samples are relabeled or removed from the update batch *on the fly*, based on the model's current state. Their participation in gradient update can therefore be reconsidered in later epochs. If the model profits from an (even potentially mislabeled) sample during a particular training stage, this sample is kept. However, the same sample may be removed during another stage where it would harm the training process.

3.1 Notation

We denote the training set by $\mathcal{X} = ((x_1, y_1), ..., (x_T, y_T))$, where y_t denotes a potentially noisy label associated with the input x_t. Each y_t corresponds to one out of K classes $\{c_1, ..., c_K\}$. The task is to utilize $\mathcal{X}$ to learn a classifier $f(\cdot; \theta)$, parameterized by θ, using an *update loss function* $\mathcal{L}(x, y)$. Additionally, we define a *comparison loss function* $\widetilde{\mathcal{L}}(x, y)$ that is used for computing the compared gradients. AGRA does not put any restrictions on the used loss functions; the update loss $\mathcal{L}(x, y)$ and the comparison loss $\widetilde{\mathcal{L}}(x, y)$ can differ.

3.2 Algorithm Description

AGRA consists of a single model training loop. For each update batch $\mathcal{B}$, another batch $\widetilde{\mathcal{B}}$ of the same size is independently sampled from the training dataset $\mathcal{X}$.

While $\mathcal{B}$ is leveraged to adjust the model weights during training, $\widetilde{\mathcal{B}}$ represents the comparison batch that is used to detect outliers.

First, the batch-wise gradient on the comparison batch $\widetilde{\mathcal{B}}$ is computed with respect to the loss function $\widetilde{\mathcal{L}}$ and flattened into a vector, resulting in the comparison gradient $\nabla\widetilde{\mathcal{L}}_{com}$. Then, the gradient for each individual data point $(x_t, y_t) \in \mathcal{B}$ is calculated with respect to the loss $\widetilde{\mathcal{L}}$ and flattened, resulting in $\nabla\widetilde{\mathcal{L}}(x_t, y_t)$. Next, the pair-wise cosine similarity of each per-sample gradient with the comparison gradient is computed as given below[2].

$$sim_{y_t} = sim\left(\nabla\widetilde{\mathcal{L}}(x_t, y_t), \nabla\widetilde{\mathcal{L}}_{com}\right) = \frac{\nabla\widetilde{\mathcal{L}}(x_t, y_t) \cdot \nabla\widetilde{\mathcal{L}}_{com}}{||\nabla\widetilde{\mathcal{L}}(x_t, y_t)||_2 \, ||\nabla\widetilde{\mathcal{L}}_{com}||_2} \tag{1}$$

In the following, sim_{y_t} is referred to as the *similarity score given label* y_t.

The next step can be realized in two different settings:

1. *without an alternative label*: a data sample is removed from the update batch if its associated similarity score is non-positive and kept otherwise:

$$\mathcal{B} \leftarrow \begin{cases} \mathcal{B} \setminus \{(x_t, y_t)\}, & \text{if } sim_{y_t} \leq 0 \\ \mathcal{B}, & \text{otherwise} \end{cases}$$

2. *with an alternative label*: in addition to the options of removing a training instance or retaining it with its original annotation, the instance can also be included in the update with the alternative label y'. If such an alternative label y' is specified, the similarity $sim_{y'} = sim\left(\nabla\widetilde{\mathcal{L}}(x_t, y'), \nabla\widetilde{\mathcal{L}}_{com}\right)$ is additionally calculated using Eq. 1.

   Depending on the values of sim_{y_t} and $sim_{y'}$, the sample is handled as follows:
   - if the similarity score is non-positive given both y_t and y', the sample is removed from the batch,
   - if the similarity score given label y' is positive and higher than the similarity score given y_t, the original label y_t is changed to y',
   - if the similarity score given label y_t is positive and higher than or equal to the similarity score given y', the original label y_t is kept.

$$\mathcal{B} \leftarrow \begin{cases} \mathcal{B} \setminus \{(x_t, y_t)\}, & \text{if } sim_{y_t} \leq 0, sim_{y'} \leq 0 \\ \mathcal{B} \setminus \{(x_t, y_t)\} \cup \{(x_t, y')\}, & \text{if } sim_{y'} > 0, sim_{y'} > sim_{y_t} \\ \mathcal{B}, & \text{otherwise} \end{cases}$$

   The decision regarding the choice of an alternative label and its sensibility depends on the characteristics and requirements of the specific dataset. An intuitive approach is to use a negative class if it is present in the data (e.g., *"no_relation"* for relation extraction, or *"non_spam"* for spam detection).

After each sample in $\mathcal{B}$ was considered for removal or correction, the model parameters are updated with respect to $\mathcal{L}$ and $\mathcal{B}$ before the processing of the next batch starts. The method is summarized in Algorithm 1, and the graphical explanation is provided in Figure 1.

[2] The subscript x_t is omitted in the short-hand notation sim_{y_t} for brevity.

Algorithm 1: AGRA Algorithm for Single-Label Datasets

Input: training set $\mathcal{X}$, initial model $f(\cdot;\theta)$, number of epochs E, batch size M, (optionally: alternative label y')
Output: trained model $f(\cdot;\theta^*)$
for *epoch = 1,..., E* **do**
  for *batch* $\mathcal{B}$ **do**
    Sample a comparison batch $\widetilde{\mathcal{B}}$, $\widetilde{\mathcal{B}} \subset \mathcal{X}$, $|\widetilde{\mathcal{B}}| = M$
    Compute $\nabla\widetilde{\mathcal{L}}_{com}$ on $\widetilde{\mathcal{B}}$
    for $(x_t, y_t) \in \mathcal{B}$ **do**
      Compute $\nabla\widetilde{\mathcal{L}}(x_t, y_t)$
      $sim_{y_t} = sim\left(\nabla\widetilde{\mathcal{L}}(x_t, y_t), \nabla\widetilde{\mathcal{L}}_{com}\right)$ (Eq. 1)
      if *an alternative label* y' *is specified* **then**
        Compute $\nabla\widetilde{\mathcal{L}}(x_t, y')$
        $sim_{y'} = sim\left(\nabla\widetilde{\mathcal{L}}(x_t, y'), \nabla\widetilde{\mathcal{L}}_{com}\right)$ (Eq. 1)
        if $sim_{y_t} \leq 0$ *and* $sim_{y'} \leq 0$ **then**
          $\mathcal{B} \leftarrow \mathcal{B} \setminus \{(x_t, y_t)\}$
        if $sim_{y'} > 0$ *and* $sim_{y'} > sim_{y_t}$ **then**
          $\mathcal{B} \leftarrow \mathcal{B} \setminus \{(x_t, y_t)\} \cup \{(x_t, y')\}$
      else
        if $sim_{y_t} \leq 0$ **then**
          $\mathcal{B} \leftarrow \mathcal{B} \setminus \{(x_t, y_t)\}$
    $\theta \leftarrow Optim(\theta, \mathcal{B}, \mathcal{L})$

3.3 Comparison Batch Sampling

Since the comparison gradient is an essential component of AGRA's outlier detection, it should be sampled in a way that does not disadvantage instances of any class. For datasets with a fairly even class distribution, randomly selecting samples from the training data might be sufficient to get a well-balanced comparison batch. However, when dealing with imbalanced datasets, this approach may result in an underrepresentation of rare classes in the comparison batch. Consequently, the gradients of samples belonging to rare classes may not match the aggregated gradient computed almost exclusively on instances assigned to more common classes.

For such cases, AGRA provides *class-weighted sampling* in order to ensure a large enough fraction of minority class instances in the comparison batch. The weight for class c_k is computed as the inverted number of occurrences of class c_k in training set $\mathcal{X}$[3]:

$$\frac{1}{\sum_{t=1}^{|\mathcal{X}|} \mathbb{1}(y_t = c_k)}$$

[3] In our implementation, these weights are passed to `WeightedRandomSampler` [1] provided by the `torch` library. Note that `WeightedRandomSampler` does not assume that the weights sum up to 1.

As a result, samples of common classes are generally less likely to be included in the comparison batch than those of rare classes, turning the resulting comparison batch into a well-formed representative of all classes.

3.4 Selection of Comparison Loss Functions

AGRA does not imply any restrictions on the choice of the comparison loss function. For example, it can be combined with a standard **cross-entropy (CE)** loss function, which is suitable for both binary and multi-class classification problems, or **binary cross-entropy (BCE)**, which is commonly used in the multi-label setting. However, despite its effectiveness in many scenarios, the cross-entropy loss has been shown to exhibit overfitting on easy and under-learning on hard classes when confronted with noisy labels [49]. Overall, cross-entropy losses can hardly be claimed robust to noise, making learning with noisy data even more challenging.

Aiming at reducing this effect, we adapted an $\boldsymbol{F_1}$ **loss function** which directly represents the performance metric and aims to maximize the F_1 score. The F_1 loss function is similar to the standard F_1 score with one major difference: the predicted labels used for the calculation of true positives, false positives, and false negatives are replaced by the model outputs transformed into predicted probabilities by a suitable activation function. This modification enables the F_1 score to become differentiable, making it compatible with gradient-based learning methods. In contrast to previous research on leveraging the F_1 score as a loss function [8], we investigate F_1 loss variants outside of the multi-label setting and gauge its efficacy in the presence of label noise.

For the multi-class single-label case, the F_1 loss is based on macro-F_1 metric:

$$\mathcal{L}_{F_{1_M}}(\mathcal{B}) = 1 - \frac{1}{K}\sum_{k=1}^{K}\frac{2\widehat{tp}_k}{2\widehat{tp}_k + \widehat{fp}_k + \widehat{fn}_k + \epsilon}$$

where

$$\widehat{tp}_k = \sum_{t=1}^{M} \hat{y}_{t,k} \times \mathbb{1}\left(y_t = c_k\right),$$
$$\widehat{fp}_k = \sum_{t=1}^{M} \hat{y}_{t,k} \times \left(1 - \mathbb{1}\left(y_t = c_k\right)\right),$$
$$\widehat{fn}_k = \sum_{t=1}^{M} \left(1 - \hat{y}_{t,k}\right) \times \mathbb{1}\left(y_t = c_k\right)$$

and $\hat{y}_{t,k}$ denotes the predicted probability of class k for sample t after application of the softmax, $\times$ represents the element-wise product, and $\epsilon = 1e-05$ in our experiments. The F_1 loss for the multi-class multi-label setting is also based on the macro-F_1 score, while for the binary single-label setting, it is based on the F_1

score of the positive class. The exact formulas for these variants are provided in Appendix B. Our experiments demonstrate that the F_1 loss function is beneficial as a comparison loss for some datasets compared to the classic cross-entropy loss. However, we emphasize that its use is not mandatory for our algorithm: AGRA is compatible with any loss function.

4 Experiments

In this section, we demonstrate the performance of our algorithm on several noisy datasets, compare it with various baselines, and analyze the obtained results.

4.1 Datasets

We evaluate our method AGRA on seven different datasets. First, we choose three weakly supervised datasets from the Wrench [52] benchmark: (1) **YouTube** [3] and (2) **SMS** [4, 6] are spam detection datasets, and (3) **TREC** [32, 6] is a dataset for question classification. The labeling functions used to obtain noisy annotations based on keywords, regular expressions, and heuristics are provided in previous work [52, 6]. Next, we evaluate AGRA on two weakly supervised topic classification datasets in African languages, namely (4) **Yorùbá** and (5) **Hausa** [20]; the keyword-based labeling functions were provided by the datasets' authors. In order to obtain noisy labels for the training instances of the above datasets, we apply the provided labeling functions and use simple majority voting with randomly broken ties. Samples without any rule matches (which are 12% in (1), 59% in (2), 5% in (3), and none in (4) and (5)) are assigned to a random class.

Apart from NLP datasets, we also conduct experiments on two image datasets: (6) **CIFAR-10** [28], for which the noisy labels were generated by randomly flipping the clean labels following Northcutt et al. [36] with 20% noise and 0.6 sparsity, and (7) **CheXpert**, a multi-label medical imaging dataset [24]. Since the CheXpert test set is not revealed in the interest of the CheXpert competition, the original hand-labeled validation set was used as a test set as in previous works [18], while a part of the training set was kept for validation purposes. We used the noisy training annotations provided by Irvin et al. [24], which were obtained by applying the CheXpert labeler to the radiology reports associated with the images[4]. Since CheXpert is a multi-label classification task, we adapt our algorithm to the multi-label setting by performing the gradient comparison with respect to each output node, allowing to ignore individual entries of the label vector. The exact algorithm can be found in Appendix B.

The dataset statistics are collected in Table 1. Each dataset's noise amount was calculated by comparing the noisy training labels to the gold labels. The gold training labels are not provided for CheXpert, so its noise rate value is missing in the table. More details about dataset preprocessing, as well as the label distributions of the datasets, are provided in Appendix C.

[4] The reports are not publicly accessible; only the noisy labels are available for the training data. The gold labels are not provided.

Dataset	#Class	#Train	#Dev	#Test	%Noise
YouTube	2	1586	120	250	18.8
SMS	2	4571	500	500	31.9
TREC	6	4965	500	500	48.2
Yorùbá	7	1340	189	379	42.3
Hausa	5	2045	290	582	50.6
CheXpert	12	200599	22815	234	-
CIFAR-10	10	50000	5000	5000	20

Table 1: Datasets statistics. The percentage of noise is calculated by comparing the noisy labels to the gold-standard annotations.

4.2 Baselines

We compare AGRA towards seven baselines. For datasets that include gold training labels (i.e., all datasets in our experiments except CheXpert), we trained a (1) **Gold** model with ground-truth labels; it can serve as an upper-bound baseline. (2) **No Denoising** baseline entails simple model training with the noisy labels, without any additional data improvement. (3) **DP** [39] stands for the Data Programming algorithm, which improves the imperfect annotations by learning the structure within the labels and rules in an unsupervised fashion by a generative model. (4) **MeTaL** [38] combines signals from multiple weak rules and trains a hierarchical multi-task network. (5) **FlyingSquid** [15] rectifies the noisy annotations using an Ising model by a triplet formulation. The experiments with the above baselines were realized using the Wrench framework [52]. In addition to the methods (3), (4), and (5) that are specifically designed for the weakly supervised setting, we also compare AGRA with two baselines that have broader applicability for learning with noisy labels: (6) **CORES**^{2} [11], which utilizes confidence regularization to sieve out samples with corrupted labels during training, and (7) **Cleanlab** [36], which aims at detecting noisy annotations by estimating the joint distribution between noisy and true labels using the out-of-sample predicted probabilities.

Since DP, MeTaL, and FlyingSquid require access to annotation rules and rule matches, they cannot be applied to non-weakly supervised datasets or other datasets for which this information is not available (such as CheXpert, for which the reports used for annotation are not publicly released). In contrast, Cleanlab, CORES^2, and AGRA directly utilize noisy labels and do not require additional information regarding the annotations, making them more broadly applicable.

4.3 Experimental Setup

AGRA was implemented based on Python using the PyTorch library. We evaluate our method with a logistic regression classifier optimized with Adam[5]. For

[5] AGRA can also be used with any PyTorch-compatible deep model as our method has no model-related limitations.

	YouTube (Acc)	**SMS** (F1)	**TREC** (Acc)	**Yorùbá** (F1)	**Hausa** (F1)	**Avg.**	**CIFAR** (Acc)	**CXT** (AUR)
Gold	94.8±0.8	95.4±1.0	89.5±0.3	57.3±0.4	78.5±0.3	83.1	83.6±0.0	–
No Denoising	87.4±2.7	71.7±1.4	58.7±0.5	44.6±0.4	39.7±0.8	60.4	82.4±0.2	82.7±0.1
Weak Supervision								
DP [39]	90.8±1.0	44.1±6.7	54.3±0.5	**47.8**±1.7	40.9±0.6	55.6	–	–
MeTaL [38]	92.0±0.8	18.3±7.8	50.4±1.7	38.9±3.1	45.5±1.1	49.0	–	–
FS [15]	84.8±1.2	16.3±6.0	27.2±0.1	31.9±0.7	37.6±1.0	39.6	–	–
Noisy Learning								
CORES^2 [11]	88.8±3.6	85.8±1.8	61.8±0.5	43.0±0.7	**51.2**±0.5	66.1	83.4±0.1	–
Cleanlab [36]	91.3±1.2	80.6±0.3	60.9±0.4	43.8±1.3	40.3±0.3	63.4	83.3±0.0	81.2±0.2
AGRA	**93.9**±0.7	**87.7**±1.2	**63.6**±0.7	46.9±1.5	46.2±1.6	**67.7**	**83.6**±0.0	**83.9**±0.3

Table 2: Experimental results on NLP and image datasets averaged across five runs and reported with standard deviation.

text-based datasets, we use TF-IDF feature vectors to represent the data. The CheXpert images were encoded with a fine-tuned EfficientNet-B0 [46], and the CIFAR-10 images were encoded with a fine-tuned ResNet-50 [19] (following previous work [36]). More details on the data encoding and resulting feature vectors are provided in Appendix D. In our experiments with TF-IDF representations, we found that the gradient entries corresponding to the biases of the model strongly influence the computed similarity scores despite being feature-independent. Hence, we exclude the elements corresponding to the biases when determining the gradient similarity for sparse features. To make our experiments consistent, we apply the same strategy to CIFAR-10 and CheXpert.

For each dataset, we report the same evaluation metrics as in previous works: commonly used accuracy and F_1 scores and macro-AUROC (Area Under the Receiver Operating Characteristics) for CheXpert [24][6]. The hyper-parameters were selected with a grid search; more details and the selected parameter values are provided in Appendix E[7]. After training each model for 10 epochs (5 epochs for CheXpert), we reload the best model state based on validation performance and evaluate it on the test set.

4.4 Results

The results of the experiments across the seven datasets are summarized in Table 2. AGRA is the best-performing method overall for three weakly-supervised

[6] The AUROC was computed on the nine classes which have more than one positive observation in the test set.

[7] The experiments are performed on a single Tesla V100 GPU on Nvidia DGX-1. The parameter grid search takes between 24 and 96 hours; the model run with the selected parameters takes between 1 and 5 minutes depending on parameter values and dataset size.

	No Weighted Sampling		Weighted Sampling	
	CE/CE	CE/F_1	CE/CE	CE/F_1
YouTube	92.0 ± 1.0	$\mathbf{93.9 \pm 0.7}$	91.9 ± 0.5	93.4 ± 0.8
YouTube†	90.5 ± 1.0	–	92.0 ± 0.7	–
SMS	79.0 ± 3.2	61.1 ± 5.2	$\mathbf{87.7 \pm 1.2}$	49.1 ± 3.0
SMS†	71.1 ± 3.1	–	86.3 ± 1.2	–
TREC	61.6 ± 0.6	62.1 ± 0.4	62.8 ± 1.1	$\mathbf{63.6 \pm 0.7}$
Yorùbá	44.3 ± 2.5	44.2 ± 1.4	43.5 ± 1.0	$\mathbf{46.9 \pm 1.5}$
Hausa	41.2 ± 0.4	40.9 ± 0.6	43.8 ± 2.8	$\mathbf{46.2 \pm 1.6}$
CheXpert	82.6 ± 0.6	$\mathbf{83.9 \pm 0.3}$	–	–
CIFAR	82.2 ± 0.2	83.5 ± 0.0	83.1 ± 0.0	$\mathbf{83.6 \pm 0.0}$

Table 3: AGRA experimental test results with different settings: use of class-weighted sampling, [training loss]/[comparison loss]. The results marked with † are obtained by AGRA with an alternative label. All results are averaged across 5 runs and reported with standard deviation.

NLP datasets, providing better results than the methods specifically designed for weakly supervised data. Among the text-based datasets, the average improvement achieved by AGRA over FlyingSquid, MeTaL, and DP is 28.1 percentage points (pp), 18.7pp, and 12.1pp correspondingly. Compared to the baselines designed for weakly supervised data, Cleanlab and CORES^2 worked better on average, but AGRA also demonstrates an improvement over them (by 4.3pp and 1.6pp, respectively). Notably, AGRA improves the results of *all* datasets over simple training without additional denoising (7.3pp improvement on average). For image datasets, AGRA also demonstrates an improvement over Cleanlab and CORES^2 as well as the no denoising baseline; for the Cifar dataset, our method demonstrates even the same result as the model trained with gold labels. The other baselines are not applicable to these datasets[8].

4.5 Ablation Study

Table 3 shows the AGRA performance across all comparison losses and comparison batch sampling strategies, the best of which was included in Table 2. Overall, it outperforms the baselines on most of the datasets in the vast majority of settings. For the binary YouTube and SMS datasets, we also perform experiments with an alternative label (the negative *"non_spam"* class for both datasets). However, the models trained with the alternative label setting are not the best-performing AGRA configuration for either dataset (although they

[8] The weak supervision baselines cannot be run on CIFAR since it is a non-weakly supervised dataset; they also cannot be run on CheXpert as we do not have access to the labeling function matches. Furthermore, CORES^2 is not applicable for CheXpert as it does not support multi-label settings.

sometimes outperform the corresponding settings without the alternative label). Utilizing an F_1-based comparison loss function instead of standard cross entropy proved beneficial on all datasets except SMS.

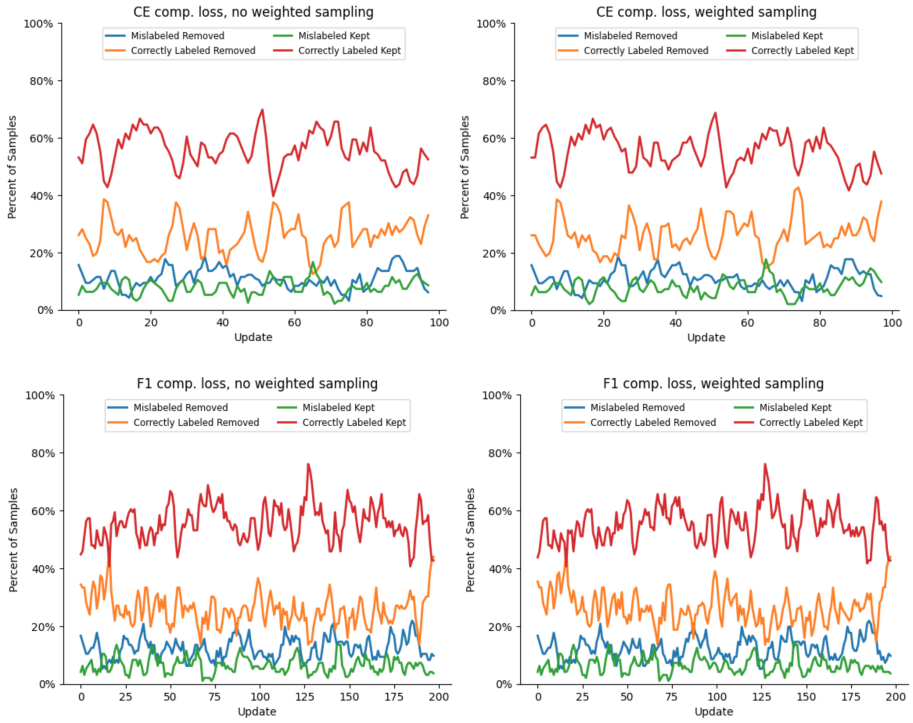

Fig. 2: Case study on the YouTube dataset. The plots represent the percentage of samples in each batch that were correctly kept, correctly removed, falsely kept and falsely removed during the training of the best-performing models for all combinations of comparison losses and sampling strategies.

As expected, weighted comparison batch sampling turns out to be especially helpful for imbalanced datasets such as Hausa (for which the most popular class is represented by 53.7% training samples, while the least frequent class only covers 7.9%) and TREC (56.6% and 1.0% correspondingly; see detailed statistics in Appendix C). On the other hand, the fairly balanced YouTube dataset performs marginally better without it.

4.6 Case study

Finally, we provide a more fine-grained analysis of our AGRA method on the example of the YouTube dataset. By comparing the noisy labels to the manual

labels provided for this dataset, we calculate the fraction of samples in each batch that are (1) mislabeled and removed, (2) correctly labeled and removed, (3) mislabeled and kept, (4) correctly labeled and kept. These statistics are reflected in Figure 2 for all supported combinations of comparison losses and comparison batch sampling strategies[9]. A remarkable trend is that the correctness of removed samples appears to be not crucial for training a reliable model. In fact, the amount of mislabeled samples kept and correctly labeled samples removed is high for many batches, yet all configurations outperform the baselines (excluding MeTaL which ties with the CE-based settings). This observation affirms our hypothesis that mislabeled samples can be beneficial at certain stages of the training process, and cleaning the dataset by filtering out all presumably mislabeled samples before training (as is done in common data-cleaning methods) might be a suboptimal approach. Moreover, the number of "falsely" kept (*mislabeled kept*) and removed (*correctly labeled removed*) samples in some cases exceeds the amount of "correctly" removed (*mislabeled removed*) and kept (*correctly labeled kept*) ones respectively. This observation reinforces our point that the usefulness of a sample at the current training stage cannot be solely determined by whether it is mislabeled or not. Weighted comparison batch sampling seems to only have a minor influence on the training process for YouTube. This observation can likely be explained by the already balanced noisy label distribution of the YouTube dataset.

5 Conclusion

In this work, we address the challenge of training a classifier using noisy labels. Most importantly, we reconsider the goal of learning with noisy annotations and focus on training a stable and well-performing classifier rather than obtaining clean and error-free data. Instead of following the traditional approach of first denoising the data and then training a classifier on the cleaned data, we propose a novel integrated approach that dynamically adjusts the *use* of the dataset during the learning process. In our new algorithm AGRA, samples from which the model can benefit at the current training stage are retained for updating, while the ones that may hinder the learning process are disregarded or relabeled. Our algorithm outperforms several recent baselines for training with noisy data.

6 Ethical Statement

Our method can improve the model predictions and produce more useful results, but we cannot promise they are perfect, especially for life-critical domains like healthcare. Data used for training can have biases that machine learning methods may pick up, and one needs to be careful when using such models in actual applications. We relied on datasets that were already published and did not hire anyone to annotate them for our work.

[9] The hyper-parameters chosen by the grid search were used for each depicted run. The exact values are provided in Appendix E.

7 Acknowledgement

This research was funded by the WWTF through the project "Knowledge-infused Deep Learning for Natural Language Processing" (WWTF Vienna Research Group VRG19-008).

References

1. Torch.utils.data. https://pytorch.org/docs/stable/data.html (2023), accessed: June 23, 2023
2. Al-Zoubi, M.B.: An effective clustering-based approach for outlier detection. European Journal of Scientific Research **28**(2), 310–316 (2009)
3. Alberto, T.C., Lochter, J.V., Almeida, T.A.: Tubespam: Comment spam filtering on youtube. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). pp. 138–143 (2015)
4. Almeida, T.A., Hidalgo, J.M.G., Yamakami, A.: Contributions to the study of sms spam filtering: New collection and results. In: Proceedings of the 11th ACM Symposium on Document Engineering. p. 259–262 (2011)
5. Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Unsupervised label noise modeling and loss correction. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019 (2019)
6. Awasthi, A., Ghosh, S., Goyal, R., Sarawagi, S.: Learning from rules generalizing labeled exemplars. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020 (2020)
7. Bai, M., Wang, X., Xin, J., Wang, G.: An efficient algorithm for distributed density-based outlier detection on big data. Neurocomputing **181**, 19–28 (2016)
8. Bénédict, G., Koops, H.V., Odijk, D., de Rijke, M.: Sigmoidf1: A smooth f1 score surrogate loss for multilabel classification. Transactions on Machine Learning Research (2022)
9. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)
10. Chen, P., Liao, B.B., Chen, G., Zhang, S.: Understanding and utilizing deep neural networks trained with noisy labels. In: International Conference on Machine Learning. pp. 1062–1070 (2019)
11. Cheng, H., Zhu, Z., Li, X., Gong, Y., Sun, X., Liu, Y.: Learning with instance-dependent label noise: A sample sieve approach. arXiv preprint arXiv:2010.02347 (2020)
12. Elahi, M., Li, K., Nisar, W., Lv, X., Wang, H.: Efficient clustering-based outlier detection algorithm for dynamic data stream. In: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery. vol. 5, pp. 298–304. IEEE (2008)
13. Fang, Z., Kong, S., Wang, Z., Fowlkes, C.C., Yang, Y.: Weak supervision and referring attention for temporal-textual association learning. CoRR **abs/2006.11747** (2020)
14. Frénay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems **25**(5), 845–869 (2014)

15. Fu, D., Chen, M., Sala, F., Hooper, S., Fatahalian, K., Re, C.: Fast and three-rious: Speeding up weak supervision with triplet methods. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. vol. 119, pp. 3280–3291 (13–18 Jul 2020)
16. Garbin, C., Rajpurkar, P., Irvin, J., Lungren, M.P., Marques, O.: Structured dataset documentation: a datasheet for chexpert. CoRR **abs/2105.03020** (2021)
17. Ghoting, A., Parthasarathy, S., Otey, M.E.: Fast mining of distance-based outliers in high-dimensional datasets. Data Mining and Knowledge Discovery **16**, 349–364 (2008)
18. Giacomello, E., Lanzi, P.L., Loiacono, D., Nassano, L.: Image embedding and model ensembling for automated chest x-ray interpretation. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2021)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. pp. 770–778 (06 2016)
20. Hedderich, M.A., Adelani, D.I., Zhu, D., Alabi, J.O., Markus, U., Klakow, D.: Transfer learning and distant supervision for multilingual transformer models: A study on african languages. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020. pp. 2580–2591 (2020)
21. Hedderich, M.A., Lange, L., Klakow, D.: ANEA: distant supervision for low-resource named entity recognition. CoRR **abs/2102.13129** (2021)
22. Huang, J., Qu, L., Jia, R., Zhao, B.: O2u-net: A simple noisy label detection approach for deep neural networks. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 3325–3333 (2019)
23. Inkawhich, N.: Finetuning torchvision models. https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html (2017), (accessed: June 13, 2022)
24. Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., et al.: Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In: Proceedings of the AAAI conference on artificial intelligence. pp. 590–597 (2019)
25. Karamanolakis, G., Mukherjee, S., Zheng, G., Awadallah, A.H.: Self-training with weak supervision. In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y. (eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021. pp. 845–863 (2021)
26. Ke, A., Ellsworth, W., Banerjee, O., Ng, A.Y., Rajpurkar, P.: Chextransfer: performance and parameter efficiency of imagenet models for chest x-ray interpretation. In: Proceedings of the Conference on Health, Inference, and Learning. pp. 116–124 (2021)
27. Knox, E.M., Ng, R.T.: Algorithms for mining distancebased outliers in large datasets. In: Proceedings of the international conference on very large data bases. pp. 392–403. Citeseer (1998)
28. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
29. Li, J., Chen, W., Huang, X., Yang, S., Hu, Z., Duan, Q., Metaxas, D.N., Li, H., Zhang, S.: Hybrid supervision learning for pathology whole slide image classification. In: de Bruijne, M., Cattin, P.C., Cotin, S., Padoy, N., Speidel, S., Zheng, Y., Essert, C. (eds.) Medical Image Computing and Computer Assisted Intervention -

MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27 - October 1, 2021, Proceedings, Part VIII (2021)
30. Li, J., Socher, R., Hoi, S.C.: Dividemix: Learning with noisy labels as semi-supervised learning. In: ICLR (2020)
31. Li, J., Wong, Y., Zhao, Q., Kankanhalli, M.S.: Learning to learn from noisy labeled data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5051–5059 (2019)
32. Li, X., Roth, D.: Learning question classifiers. In: COLING 2002: The 19th International Conference on Computational Linguistics (2002)
33. Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., Li, L.J.: Learning from noisy labels with distillation. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 1928–1936 (2017)
34. Lipton, Z.C., Wang, Y., Smola, A.J.: Detecting and correcting for label shift with black box predictors. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018. vol. 80, pp. 3128–3136 (2018)
35. Liu, Z., Chen, B., Wang, R., Liang, P.P., Salakhutdinov, R., Morency, L., Ueda, M.: Learning not to learn in the presence of noisy labels. CoRR **abs/2002.06541** (2020)
36. Northcutt, C., Jiang, L., Chuang, I.: Confident learning: Estimating uncertainty in dataset labels. Journal of Artificial Intelligence Research **70**, 1373–1411 (2021)
37. Ratner, A., Bach, S.H., Ehrenberg, H., Fries, J., Wu, S., Ré, C.: Snorkel: Rapid training data creation with weak supervision. The VLDB Journal **29**(2), 709–730 (2020)
38. Ratner, A., Hancock, B., Dunnmon, J., Sala, F., Pandey, S., Ré, C.: Training complex models with multi-task weak supervision. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 4763–4771 (07 2019)
39. Ratner, A.J., De Sa, C.M., Wu, S., Selsam, D., Ré, C.: Data programming: Creating large training sets, quickly. Advances in neural information processing systems **29** (2016)
40. Raykar, V.C., Yu, S.: Eliminating spammers and ranking annotators for crowdsourced labeling tasks. Journal of Machine Learning Research **13**(16), 491–518 (2012)
41. Ren, W., Li, Y., Su, H., Kartchner, D., Mitchell, C., Zhang, C.: Denoising multi-source weak supervision for neural text classification. In: Cohn, T., He, Y., Liu, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020. vol. EMNLP 2020, pp. 3739–3754 (2020)
42. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
43. Shi, Y., Seely, J., Torr, P.H., Siddharth, N., Hannun, A., Usunier, N., Synnaeve, G.: Gradient matching for domain generalization. arXiv preprint arXiv:2104.09937 (2021)
44. Stephan, A., Kougia, V., Roth, B.: SepLL: Separating latent class labels from weak supervision noise. In: Findings of the Association for Computational Linguistics: EMNLP 2022. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2022)
45. Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., Fergus, R.: Training convolutional networks with noisy labels. arXiv preprint arXiv:1406.2080 (2014)

46. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019)
47. Tratz, S., Hovy, E.: A taxonomy, dataset, and classifier for automatic noun compound interpretation. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. pp. 678–687 (Jul 2010)
48. Wang, H., Bah, M.J., Hammad, M.: Progress in outlier detection techniques: A survey. Ieee Access **7**, 107964–108000 (2019)
49. Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., Bailey, J.: Symmetric cross entropy for robust learning with noisy labels. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 322–330 (2019)
50. Wang, Z., Shang, J., Liu, L., Lu, L., Liu, J., Han, J.: Crossweigh: Training named entity tagger from imperfect annotations. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019 (2019)
51. Wei, J.: Label noise reduction without assumptions. Dartmouth College Undergraduate Theses. 164. (2020)
52. Zhang, J., Yu, Y., Li, Y., Wang, Y., Yang, Y., Yang, M., Ratner, A.: WRENCH: A comprehensive benchmark for weak supervision. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2021)
53. Zhao, B., Mopuri, K.R., Bilen, H.: Dataset condensation with gradient matching. In: International Conference on Learning Representations (2021)

A F_1-Based Losses

This section contains the formulas for the F_1-based loss functions used in the experiments. In the following, $\hat{y}_{t,k}$ denotes the model output for sample t corresponding to the k-th class, c_k, after application of an activation function. In the binary single-label setting, where $y_t \in \{0, 1\}$, the positive class is assumed to be denoted by label 1. A small stabilizer ϵ was added to the denominator of the F_1 formula, inspired by past research on F_1-based loss functions[10]; $\epsilon = 1e-05$ was used in the experiments.

Binary-F_1 Loss for Single-Label Settings The F_1 loss geared towards binary classification tasks aims to maximize the F_1 score of the positive class. Given a batch $\mathcal{B}$, the loss can be computed by the following formula.

$$\mathcal{L}_{F_1}(\mathcal{B}) = 1 - \frac{2\widehat{tp}}{2\widehat{tp} + \widehat{fp} + \widehat{fn} + \epsilon},$$

where

$$\begin{aligned}
\widehat{tp} &= \sum_{t=1}^{M} \hat{y}_{t,1} \times y_t, \\
\widehat{fp} &= \sum_{t=1}^{M} \hat{y}_{t,1} \times (1 - y_t), \\
\widehat{fn} &= \sum_{t=1}^{M} (1 - \hat{y}_{t,1}) \times y_t,
\end{aligned}$$

$y_t \in \{0, 1\}$ and $\hat{y}_{t,1}$ denotes the predicted probability for the positive class for sample t after application of the softmax.

Macro-F_1 Loss for Multi-Label Settings In the multi-label case, we use a macro-averaged F_1 loss, which averages the differentiable F_1 scores of all K classes.

$$\mathcal{L}_{F_{1_M}}(\mathcal{B}) = 1 - \frac{1}{K} \sum_{k=1}^{K} \frac{2\widehat{tp}_k}{2\widehat{tp}_k + \widehat{fp}_k + \widehat{fn}_k + \epsilon},$$

where

[10] The stabilizer was also used in https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d, accessed: 2022-08-11

$$\widehat{tp}_k = \sum_{t=1}^{M} \hat{y}_{t,k} \times y_{t,k},$$
$$\widehat{fp}_k = \sum_{t=1}^{M} \hat{y}_{t,k} \times (1 - y_{t,k}),$$
$$\widehat{fn}_k = \sum_{t=1}^{M} (1 - \hat{y}_{t,k}) \times y_{t,k},$$

$y_{t,k} \in \{0, 1\}$ and $\hat{y}_{t,k}$ denotes the predicted probability of class k for sample t after application of the sigmoid function.

B AGRA Multi-Label Setting

In the multi-label setting, the annotation for each data point is given by a binary vector of size K, where K is the number of classes. The AGRA multi-label setting allows to ignore individual entries of this vector for the update. For the following formulation of the multi-label variant of AGRA, the label-specific gradients are denoted by $\left(\nabla\widetilde{\mathcal{L}}\left(x_t, y_t\right)\right)_k$ and $\left(\nabla\widetilde{\mathcal{L}}_{com}\right)_k$, where $k \in \{1, ..., K\}$, and the corresponding similarity score is given by $sim^k_{y_t} = sim\left(\left(\nabla\widetilde{\mathcal{L}}\left(x_t, y_t\right)\right)_k, \left(\nabla\widetilde{\mathcal{L}}_{com}\right)_k\right)$. If a particular label $y_{t,k}$ should not be considered for the update, due to $sim^k_{y_t}$ being non-positive, it is replaced with the label i. For the model update, we use a masked version of the BCE loss that ensures that the replaced labels do not influence the weight update. Namely,

$$\widetilde{\mathcal{L}}_{BCE}(\mathcal{B}) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\widetilde{M}_k} \sum_{t=1}^{M} -[\log(\hat{y}_{t,k}) y_{t,k} + \log(1 - \hat{y}_{t,k})(1 - y_{t,k})] \mathbb{1}(\widetilde{y}_{t,k} \neq i)$$

where $\widetilde{M}_k$ denotes the number of retained labels for class k and $\widetilde{y}$ denotes the masked label vector. Pseudocode for the AGRA multi-label setting can be found in Algorithm 2.

Algorithm 2: AGRA Algorithm for Multi-Label Datasets

Input: training set $\mathcal{X}$, initial model $f(\cdot;\theta)$, number of epochs E, batch size M, number of classes K, label assigned to ignored entries i
Output: trained model $f(\cdot;\theta^*)$
for *epoch = 1,..., E* **do**
  for *batch* $\mathcal{B}$ **do**
    Sample a comparison batch $\widetilde{\mathcal{B}}$, $\widetilde{\mathcal{B}} \subset \mathcal{X}$, $|\widetilde{\mathcal{B}}| = M$
    Compute $\nabla\widetilde{\mathcal{L}}_{com}$ on $\widetilde{\mathcal{B}}$
    for $(x_t, y_t) \in \mathcal{B}$ **do**
      Compute $\nabla\widetilde{\mathcal{L}}(x_t, y_t)$
      Set up corrected label vector $\widetilde{y}_t \leftarrow y_t$
      for *k=1, ..., K* **do**
        $sim^k_{y_t} = sim\left(\left(\nabla\widetilde{\mathcal{L}}(x_t, y_t)\right)_k, \left(\nabla\widetilde{\mathcal{L}}_{com}\right)_k\right)$ (Eq.1)
        if $sim^k_{y_t} \leq 0$ **then**
          $\widetilde{y}_{t,k} \leftarrow i$
      $\mathcal{B} \leftarrow \mathcal{B} \setminus \{(x_t, y_t)\} \cup \{(x_t, \widetilde{y}_t)\}$
    $\theta \leftarrow Optim(\theta, \mathcal{B}, \mathcal{L})$

C Dataset Preprocessing & Statistics

We used already preprocessed versions of text datasets in the Wrench framework [52]. For CheXpert, where each class corresponds to one pathology that might be present in a training sample, we needed to additionally adapt the labels so that they can be used for our experiments. In the original dataset, the label assigned to each class equals either -1 (uncertain), blank (not mentioned), 0 (negative), or 1 (positive). We binarized the label vector by mapping -1 to 1 (potentially increasing the noise ratio, but keeping more positive training samples), and blank to 0. The assignment of the value 0 to the blank labels is somewhat intuitive: not every radiographic report will mention all 12 pathologies, and if an observation is not even mentioned in the study report, there likely are no indications for it in the corresponding images [16, 24]. Among the possible ways of handling the uncertainty labels (turning them to either 0 or 1, ignoring them, or keeping them as a separate class), we decided to choose the reassignment to label 1 as it was the best performing method for most of the classes in [24]. We focus on predicting the 12 pathologies in the CheXpert dataset, omitting the additional "Support Devices" and "No Finding" labels.

Table 4 provides information about the number of samples belonging to each class for all datasets used in the experiments. Note that in the multi-class multi-label CheXpert dataset, one sample can belong to multiple classes.

D Data Encoding

For the text datasets, we used feature-based TF-IDF vector representations. The samples were encoded with the `TfidfVectorizer` tool available in Sklearn library[11].

For the CheXpert dataset, we used a fine-tuned convolutional neural network (CNN) as a feature extractor, following Giacomello et al. [18]. In our experiments, the data samples were encoded using a fine-tuned EfficientNet-B0 [46]. The `torchvision` implementation [23] of EfficientNet-B0 used for our experiments was pre-trained on ImageNet. Despite the fact that the images contained in ImageNet greatly differ from chest radiographs, the pre-training still gives good intialization, as was shown by Ke et al. [26]. The chest radiographs were rescaled to a size of 224x224 and normalized with the mean and standard deviation from ImageNet. Then, the EfficientNet was fine-tuned on the training data for two epochs using the Adam optimizer with learning rate 0.0001 and batch size 16. Once the fine-tuning was completed, 1280-dimensional feature vectors for each image were extracted from the penultimate layer of the network and used as the data representations.

For CIFAR-10, we fine-tune a pre-trained ResNet-50 for 100 epochs with learning rate 0.001, batch size 64 and no weight decay and retrieve emeddings from the penultimate layer.

[11] https://scikit-learn.org/stable/

Dataset	Class	%Samples
YouTube	SPAM	52.3%
	HAM	47.7%
SMS	SPAM	63.8%
	HAM	36.2%
TREC	DESC	56.6%
	ENTY	8.0%
	HUM	12.8%
	ABBR	1.0%
	LOC	8.0%
	NUM	13.6%
Yorùbá	Africa	6.1%
	Entertainment	25.2%
	Health	6.5%
	Nigeria	10.3%
	Politics	26.2%
	Sport	6.8%
	World	18.9%
Hausa	Africa	19.6%
	Health	10.1%
	Nigeria	8.6%
	Politics	7.9%
	World	53.7%
CheXpert	Enlarged Cardiomediastinum	10.4%
	Cardiomegaly	15.8%
	Lung Opacity	49.8%
	Lung Lesion	4.8%
	Edema	29.1%
	Consolidation	19.1%
	Pneumonia	11.1%
	Atelectasis	30.1%
	Pneumothorax	10.1%
	Pleural Effusion	43.7%
	Pleural Other	2, 8%
	Fracture	4.3%

Table 4: Percentage of samples belonging to each class in all data sets used for the experiments. Note that CheXpert dataset in designed for a multi-label classification, meaning more than one class can be assigned to one sample.

E The Hyper-Parameters and Search Space

The hyper-parameters for our algorithm and baselines were retrieved with a grid search on the validation sets; each trial was performed 3 times with different initialization. The search space is provided in Table 6, and the exact selected values for our method AGRA can be found in Table 5. All AGRA results presented in the paper are reproducible with the seed value 0.

Dataset	Learning Rate η	Batch Size M	Weight Decay λ
YouTube	$1e-2$	32	$1e-3$
SMS	$1e-3$	128	$1e-3$
TREC	$1e-1$	32	$1e-4$
Yorùbá	$1e-1$	512	$1e-4$
Hausa	$1e-1$	512	$1e-4$
CIFAR-10	$1e-3$	512	$1e-4$
CheXpert	$1e-3$	128	$1e-3$

Table 5: Selected hyper-parameters with grid search for the best AGRA configurations, which are:

- for YouTube: $F1$ comparison loss, no weighted sampling, no alternative label,
- for SMS: CE comparison loss, weighted sampling, no alternative label,
- for TREC: $F1$ comparison loss, weighted sampling,
- for Yorùbá: $F1$ comparison loss, weighted sampling,
- for Hausa: $F1$ comparison loss, weighted sampling,
- for CIFAR-10: $F1$ comparison loss, weighted sampling,
- for CheXpert: $F1$ comparison loss, weighted sampling not implemented.

For CheXpert, the hyperparameters were fixed to learning rate $1e-3$, weight decay $1e-3$ and batch size 128 without grid search for AGRA and Cleanlab due to energy considerations and resource constraints. The Cleanlab hyperparameter n_folds was set to 2 for CheXpert. For CORES^2, we choose $\beta = 0.2 \cdot K$, where K is the number of classes, as suggested by [11].

Method	Hyper-parameter	Search Space
AGRA	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 128, 512
	weight decay λ	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
No denoising	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 64, 128
	weight decay λ	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
DP	DP learning rate	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	DP batch size	32, 64, 128
	DP num epochs	5, 10, 50, 100, 200
	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 64, 128
	weight decay λ	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
MeTal	MeTal learning rate	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	MeTal weight decay	32, 64, 128
	MeTal num epochs	5, 10, 50, 100, 200
	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 64, 128
	weight decay λ	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
FlyingSquid	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 64, 128
	weight decay λ	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
Cleanlab	CL n_folds	2, 5, 7, 9, 12
	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 128, 512
	weight decay λ	$1e-05$, $1e-04$, $1e-03$
CORES^2	learning rate η	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$
	batch size M	32, 128, 512
	weight decay λ	$1e-05$, $1e-04$, $1e-03$, $1e-02$, $1e-01$

Table 6: Hyper-parameters and search space used for a grid search. The search spaces for DP, MeTaL, and FS methods were inherited from Wrench [52]. The search spaces for Cleanlab and AGRA were selected empirically.