

Identifying Mislabeled Data using the Area Under the Margin Ranking

Geoff Pleiss^{1,2} Tianyi Zhang¹ Ethan R. Elenberg¹ Kilian Q. Weinberger^{1,2}

Abstract

Not all data in a typical training set help with generalization; some samples can be overly ambiguous or outrightly mislabeled. This paper introduces a new method to identify such samples and mitigate their impact when training neural networks. At the heart of our algorithm is the *Area Under the Margin* (AUM) statistic, which exploits differences in the training dynamics of clean and mislabeled samples. A simple procedure—adding an extra class populated with purposefully mislabeled *indicator samples*—learns a threshold that isolates mislabeled data based on this metric. This approach consistently improves upon prior work on synthetic and real-world datasets. On the Web-Vision50 classification task our method removes 17% of training data, yielding a 2.6% (absolute) improvement in test error. On CIFAR100 removing 13% of the data leads to a 1.2% drop in error.

1. Introduction

As deep networks become increasingly accurate at predictive tasks, the potential improvement of novel architectures in real-world applications is often inherently limited by data quality. After all, not all data sets are as reliably curated as CIFAR10/100 (Krizhevsky et al., 2009) or ImageNet (Deng et al., 2009). Many real-world datasets contain samples that are “weakly-labeled” through proxy variables or web scraping (e.g. Xiao et al., 2015; Joulin et al., 2016; Li et al., 2017a). Human annotators, especially on crowdsourced platforms, can be prone to making labeling mistakes. Moreover, even the most celebrated datasets famously contain harmful examples. See Figure 1 for suspicious examples detected by our proposed method—some are clearly mislabeled, others inherently ambiguous (e.g. a dog carrying toilet tissue).

Mislabeled training examples are problematic for overpa-

rameterized deep networks, which can achieve zero training error even on datasets with randomly-assigned labels (Zhang et al., 2017). Memorizing the mislabeled samples results in less generalizable features and worse performance. Intuitively, a single “good” filter (e.g. detecting floppy ears or a drooling tongue) can correctly classify hundreds of DOG images. If a BIRD is mislabeled as a DOG, however, the deep net must learn overly specific filters—only applicable for this very image—in order to overfit to this wrong label.

Our goal is to first identify and subsequently remove mislabeled samples from training datasets. Discarding these harmful samples reduces memorization and improves generalization on predictive tasks. Perhaps more importantly, accurate identification of mislabeled data allows practitioners to easily audit and curate their datasets. For example, a company might like to know about common labeling mistakes in order to reduce systematic error in its annotation pipeline. Large datasets may be too costly to manually inspect; therefore, an automated method should be able to isolate mislabeled samples with high precision and recall.

Prior methods deal with mislabeled data through various methods, such as validation loss on rotating hold-out sets (Chen et al., 2019) or robust loss functions (Reed et al., 2014; Zhang & Sabuncu, 2018). Recent theoretical and empirical work (Arpit et al., 2017; Allen-Zhu et al., 2019; Arora et al., 2019) suggests that the training dynamics of SGD contain salient signals about memorization and noisy data. Similar insights have been used recently to design improved training procedures for deep networks (Arazo et al., 2019; Li et al., 2020; Nguyen et al., 2020). Our work further builds off of these insights.

We propose a novel method to for identifying mislabeled samples in a training set. Consider an image of a BIRD accidentally mislabeled as a DOG. Its memorization is the outcome of a delicate tension. During training, the gradient updates of the image itself will encourage the network to (wrongly) predict the DOG label, whereas the gradient updates from other training images will encourage predicting BIRD through generalization. The opposing updates between the (incorrect) assigned label and the (hidden) true class membership is ultimately reflected in the logits during training. To this end, we introduce a new statistic, the *Area Under the Margin* (AUM), which is the average dif-

¹ASAPP Inc. ²Cornell University. Correspondence to: Geoff Pleiss <gpleiss@asapp.com>, Tianyi Zhang <tzhang@asapp.com>, Ethan R. Elenberg <eelenberg@asapp.com>.

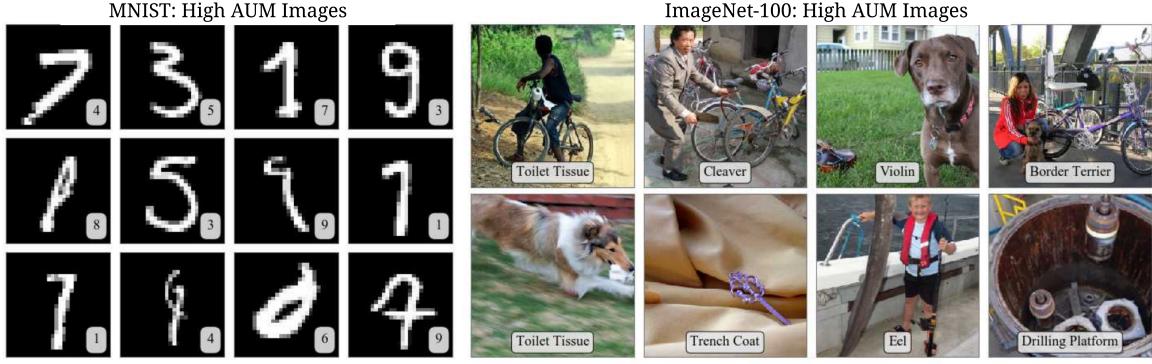


Figure 1. Images from MNIST (left) and the first 100 ImageNet classes (right) with the lowest *Area Under the Margin* (AUM) ranking. The images are mislabeled or ambiguous. MNIST AUMs are from a LeNet model; ImageNet AUMs are from a ResNet-50 model.

ference between the logit value of the assigned and highest non-assigned labels. Correctly-labeled samples do not exhibit this same tension, and consequently the AUM differs significantly between correctly- and mislabeled examples.

Naturally, the AUM score of correctly-labeled samples still varies. Some samples may look deceptively like another class or may be less canonical in appearance. In order to separate mislabeled samples from difficult (but beneficial) samples, we make a second contribution. We introduce an extra (artificial) class into the dataset and purposefully assign a small percentage of *indicator* training data to this new class. As the fake class does not actually exist in the data set, all indicator samples are definitely mislabeled. Consequently, we can use the AUM statistics of these points to obtain a natural threshold that separates correctly-labeled samples from mislabeled ones.

Equipped with the AUM statistic and indicator samples, we can confidently remove training points whose statistic falls below the threshold. Our approach exhibits three desirable properties: high detection accuracy, robustness against “hard” examples, and invariance to model architecture. On standard benchmark tasks, we improve upon the performance of existing methods simply by removing identified mislabeled samples. We are also able to clean many real-world datasets—including WebVision and Tiny ImageNet—for improved classification performance. Most surprisingly, *removing 13% of the CIFAR100 dataset results in a 1.2% reduction in test error* for a ResNet-32 model.

2. Related Work

Deep learning with noisy datasets is widely studied through a variety of angles—see [Algan & Ulusoy \(2019\)](#) for a more complete review. Several researchers have proposed novel training architectures ([Goldberger & Ben-Reuven, 2017](#)) or robust loss functions ([Reed et al., 2014](#); [Ma et al., 2018](#); [Zhang & Sabuncu, 2018](#); [Xu et al., 2019](#)). [Li et al. \(2019\)](#)

and [Hu et al. \(2019\)](#) develop theoretical guarantees in the noisy-label setting for early stopping and other regularizations. Our work aims to not only improve model robustness but also to improve the quality of training sets.

Identifying mislabeled samples. There are many approaches to robust training that either explicitly or implicitly identify mislabeled samples. Some methods filter data through cross-validation ([Chen et al., 2019](#)) or an auxiliary network ([Jiang et al., 2017](#)). The vast majority of approaches use training loss as a proxy for label quality, as mislabeled samples tend to have a higher loss at early epochs ([Zhang et al., 2017](#); [Shen & Sanghavi, 2019](#)). [Han et al. \(2018\)](#) discard a fixed percentage of data based on training loss, training another network on the remaining samples. [Arazo et al. \(2019\)](#) and [Li et al. \(2020\)](#) fit the losses of all samples with two-component mixture models, using the weights to differentiate clean from mislabeled data. Our method similarly uses training dynamics; however, we replace loss with a metric (AUM) that does not confuse difficult samples for mislabeled examples. Moreover, we use a non-parametric method (indicator samples) rather than a parametric mixture to separate correctly- and mislabeled data.

Some research has focused on a relaxed setting where a small set of training data is assumed to be “trusted”—i.e. free of mislabeled examples. The trusted data can be used to estimate the noise-transition matrix for mislabeled data ([Sukhbaatar et al., 2015](#); [Hendrycks et al., 2018](#)), or can be used to perform meta-learning ([Ren et al., 2018](#)), distillation ([Li et al., 2017b](#)), or pseudo-labeling ([Zhang et al., 2019](#)). In this paper, we consider the more restrictive setting where no subset of the training data can be trusted; all accessible data is potentially mislabeled.

Relation to semi-supervised learning/data augmentation. There has been recent interest in combining noisy-dataset learning with semi-supervised learning and data augmentation techniques. These approaches identify a small

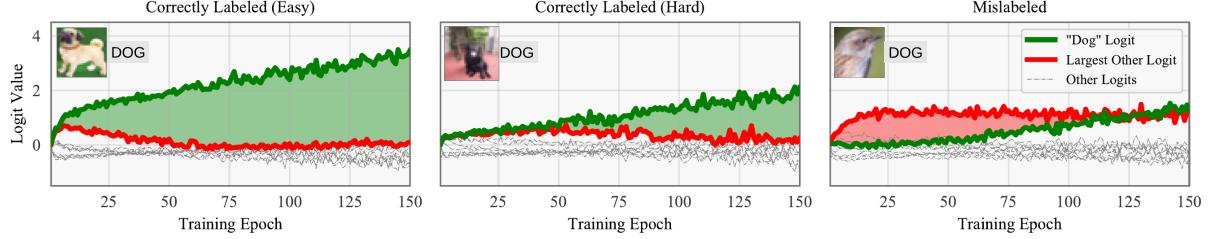


Figure 2. Illustration of the *Area Under the Margin* (AUM) metric. The graphs display logit trajectories for easy-to-learn dogs (left), hard-to-learn dogs (middle), and samples mislabeled as DOGS (right). The AUM is the shaded region between the DOG logit and the largest other logit. Green regions represent positive AUM, red regions represent negative AUM. Both the easy and hard correctly-labeled samples have a larger AUM than the mislabeled samples. (Dataset: CIFAR10 with 40% mislabeled samples.)

set of correctly-labeled data and then use the remaining untrusted data in conjunction with semi-supervised learning (Luo et al., 2018; Li et al., 2020; Ortego et al., 2019; Nguyen et al., 2020), pseudo-labeling (Zhang et al., 2019), or MixUp augmentation (Arazo et al., 2019; Zhang et al., 2018). Our paper is primarily concerned with the identification of correctly-labeled data rather than how to best use mislabeled data. For simplicity, we choose to discard data identified as mislabeled. However, any of the above methods could be used if we only discarded their labels.

3. Identifying Mislabeled Data

We assume our training dataset $\mathcal{D}_{\text{train}} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ consists of two data types. A **mislabeled sample** is one where the assigned label does not match the input. For example, \mathbf{x} might be a picture of a bird and its assigned label y might be DOG. A **correctly-labeled sample** has an assigned label that matches the ground-truth of the input. Some correctly-labeled examples might be “easy-to-learn” if they are common (e.g. $y = \text{DOG}$, \mathbf{x} is a golden retriever catching a frisbee). Others might be “hard-to-learn” if they are rare-occurrences (e.g. $y = \text{DOG}$, \mathbf{x} is an uncommon breed). In general, we assume that adding both easy and hard correctly-labeled samples to $\mathcal{D}_{\text{train}}$ improves model generalization, whereas mislabeled examples hurts generalization.

We wish to identify the mislabeled data in $\mathcal{D}_{\text{train}}$ simply by observing differences in training dynamics among samples. Our proposed method consists of a novel statistic—*area under the margin* (AUM)—as well as a mechanism to simulate the training dynamics of mislabeled data—*indicator samples*. AUM distills training differences between correctly- and mislabeled samples into a single scalar. Indicator samples have similar AUM values as mislabeled data, thereby identifying samples that should be removed from $\mathcal{D}_{\text{train}}$.

3.1. The Area Under the Margin (AUM) Ranking

Let $(\mathbf{x}, y) \in \mathcal{D}_{\text{train}}$ be a sample and let $\mathbf{z}^{(t)}(\mathbf{x}) \in \mathbb{R}^c$ be its logit vector (pre-softmax output) at epoch t . The entry

$z_i^{(t)}(\mathbf{x})$ is the logit corresponding to class i . The *margin* at epoch t captures how much larger the (potentially incorrect) assigned logit is than all other logits:

$$M^{(t)}(\mathbf{x}, y) = \frac{\text{assigned logit}}{z_y^{(t)}(\mathbf{x})} - \max_{i \neq y} z_i^{(t)}(\mathbf{x}). \quad (1)$$

A negative margin corresponds to an incorrect prediction, while a positive margin corresponds to a confident correct prediction. A sample will have a very negative margin if gradient updates from similar samples oppose the sample’s (potentially incorrect) assigned label.

We hypothesize that, at any given epoch during training, a mislabeled sample will *in expectation* have a smaller margin than a correctly-labeled sample. We capture this by averaging a sample’s margin measured at each training epoch—a metric we refer to as *area under the margin* (AUM):

$$\text{AUM}_{\mathbf{x}, y} = \frac{1}{T} \sum_{t=1}^T M^{(t)}(\mathbf{x}, y), \quad (2)$$

where T is the total number of training epochs. This metric is illustrated by Figure 2, which plots the logits for various CIFAR10 examples over the course of training a ResNet-32. Each of the 10 lines represents the logit for a particular class (averaged over 50 samples). The left and middle graphs display correctly-labeled DOG examples. We sub-divide these into “easy-to-learn” and “hard-to-learn” dogs.¹ For both sets of correctly-labeled samples the DOG logit grows larger than all other logits. The green shaded region measures the AUM, which is positive for both sets of examples and especially large for the easy-to-learn examples. Conversely, the right plot displays logits for samples that are mislabeled as DOGS.² Though the DOG logit eventually becomes the largest, it is much smaller than another logit (red line) for most of training. This large logit corresponds to BIRD—the ground-truth of these mislabeled images—and is likely due

¹ We base learning difficulty off the final training loss.

² We purposefully add these mislabelings to CIFAR10.

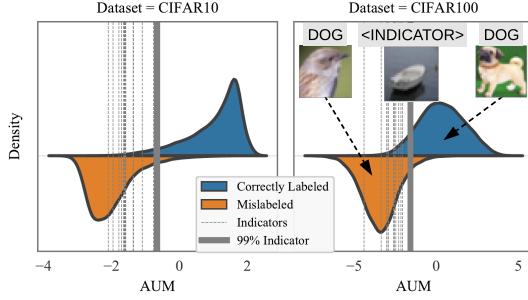


Figure 3. Illustrating the role of *indicator samples* on CIFAR10/100 with 40% mislabeled samples. The graphs are histograms of AUMs for correctly-labeled (blue) and mislabeled samples (orange). Dashed lines represent the AUM values of indicator samples. The 99th percentile of indicator AUMs (solid gray line) separates correctly- and mislabeled samples.

to gradient updates from similar-looking correctly-labeled birds. Consequentially, these mislabeled samples have a very negative AUM, signified by the red area on the graph.

3.2. Indicator Samples

Given that AUM captures differences between clean and mislabeled data, our goal is to identify mislabelings using this statistic. As seen in Figure 2 mislabeled samples tend to have a smaller AUM than correctly labeled samples. However, it is unclear what exact AUM values are indicative of mislabelings. Figure 3 displays a violin plot with the histograms of AUM values for CIFAR10/100 with 40% label noise.³ CIFAR10 samples have AUMs between -4 and 2, and most samples with negative AUMs tend to be mislabeled. However, the values on CIFAR100 tend to be more extreme (between -7 and 5), and up to 40% of clean samples have a negative AUM. The optimal threshold dividing correctly- and mislabeled samples is dataset dependent.

This introduces a challenging question: how do we choose an AUM threshold without prior information about the label quality? Our strategy is to insert fake data—which we refer to as *indicator samples*—that mimic the training dynamics of mislabeled data. Samples with similar or worse AUM values than indicator samples are assumed to be mislabeled.

Indicator samples are training samples assigned to a non-existent class. We construct indicator samples in a simple way: *take a subset of training data and re-assign their label to a brand new class*—i.e. a class that doesn’t really exist. In particular, assume that our training set has N samples that belong to c classes. We randomly select $N/(c+1)$ samples and re-assign their labels to $c+1$.⁴

³ AUM values come from a ResNet-32 trained for 150 epochs.

⁴ We add an additional neuron to the network’s output layer so that it is able to predict the $c+1$ class.

Choosing $N/(c+1)$ indicator examples ensures the extra class is as likely as other classes on average.

Using an extra class $c+1$ for indicator samples has a subtle but important property: all indicator samples are *guaranteed* to mimic mislabeled data. (Since indicator samples are constructed from potentially-mislabeled training data, assigning these samples a random label in $[1, c]$ might accidentally “correct” some mislabeled examples.) Consequentially, the network can only raise their assigned $c+1$ logit through memorization. This should result in a small and likely negative margin, just as with mislabeled examples. Figure 3 displays the AUMs of indicator samples (dashed gray lines), which are indeed smaller than correctly-labeled AUMs (blue histogram) on noisy CIFAR10/100.

Choosing an AUM threshold from indicator samples.

Since indicator samples mimic mislabeled data, we assume that no clean sample should have a lower AUM than that of an indicator sample. We therefore identify mislabeled samples as any sample with a lower AUM than the 99th percentile indicator sample. Figure 3 demonstrates the efficacy of this strategy. The 99th percentile indicator AUM (thick gray line) separates correctly- and mislabeled samples on noisy CIFAR10/100 with high precision and recall.

3.3. Our Method

Putting this all together, we propose the following procedure for identifying mislabeled data:

1. Choose a subset of data \mathcal{D}_{IND} to be indicator samples. Create a modified training set:
$$\mathcal{D}'_{\text{train}} = \{(\mathbf{x}, c+1) : \mathbf{x} \in \mathcal{D}_{\text{IND}}\} \cup (\mathcal{D}_{\text{train}} \setminus \mathcal{D}_{\text{IND}}).$$
2. Train a network on $\mathcal{D}'_{\text{train}}$ until the first learning rate drop. Compute the AUM of all samples in the process.
3. Compute α : the 99th percentile indicator sample AUM.
4. Identify mislabeled samples using α as a threshold:
$$\{(\mathbf{x}, y) \in (\mathcal{D}_{\text{train}} \setminus \mathcal{D}_{\text{IND}}) : \text{AUM}_{\mathbf{x}, y} \leq \alpha\}.$$

Details. By stopping the training before the first learning rate drop, we prevent the network from converging and therefore memorizing difficult/mislabeled examples. We find this consistently results in a more salient signal from AUM. In practice, this procedure only allows us to determine which samples in $\mathcal{D}_{\text{train}} \setminus \mathcal{D}_{\text{IND}}$ are mislabeled. We therefore must repeat this procedure using a different set of indicator samples to identify the remaining mislabeled samples. In total the whole procedure takes roughly the same amount of computation as training a normal network: two networks are trained up until the first learning rate drop (roughly halfway through the training of most networks).

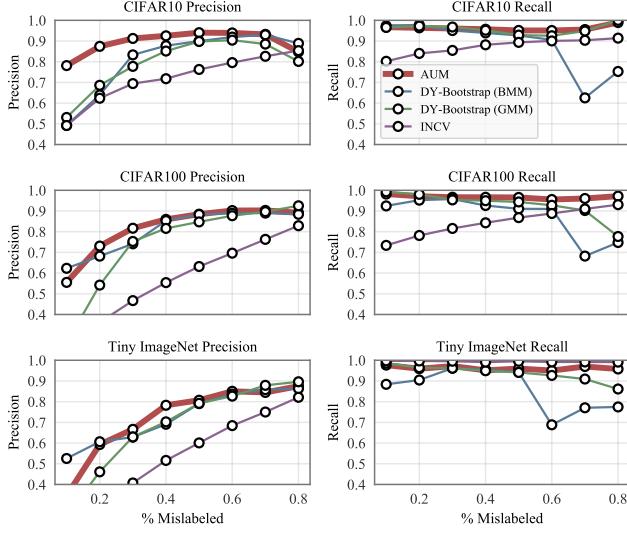


Figure 4. Precision/recall for identifying mislabeled data under uniform noise models. Results are from ResNet-32 models.

4. Experiments

We test the efficacy of AUM and indicator samples in two ways. First, we directly measure the precision and recall of our mislabeled data identification procedure. We construct synthetic noisy datasets with various noise models to have ground-truth knowledge about which samples are mislabeled. Second, we train models on noisy datasets after removing the identified data. We use test-set error as a proxy for identification performance—removing mislabeled samples should improve accuracy, whereas removing correctly-labeled samples should hurt accuracy. This allows us to test our method on real-world datasets where the quality of any particular label is unknown. In all experiments we do not assume the presence of any trusted data for training or validation. (See Appendix A for all experimental details.)

4.1. Mislabeled Sample Identification

Datasets. CIFAR10 and CIFAR100 (Krizhevsky et al., 2009) contain 32×32 images from 10 and 100 classes respectively. In all experiments, we use a subset of 45,000 images for training. We also consider Tiny ImageNet, a 200-class subset of the ImageNet (Deng et al., 2009) dataset with 95,000 images resized to be 64×64 pixels. We corrupt these datasets following a uniform noise model (the labels for mislabeled samples are assigned uniformly at random). In other words, a mislabeled samples’ (incorrect) assigned class is independent of its (hidden) ground-truth label.

Baselines. We compare against several methods from the existing literature. Arazo et al. (2019) fit the training losses of all samples mixture of two beta distributions as part of their larger method (**DY-Bootstrap BMM**). Samples that

Table 1. Test-error performance on CIFAR10 (ResNet-32) with synthetic mislabeled samples (uniform noise model).

Noise	0.20	0.40	0.60	0.80
Standard	25.0 ± 0.6	43.3 ± 0.8	63.3 ± 0.8	83.4 ± 0.4
Random	16.5 ± 0.7	22.8 ± 1.6	35.8 ± 2.8	55.6 ± 5.5
Bootstrap	22.4 ± 0.4	37.4 ± 0.9	52.0 ± 0.5	68.8 ± 0.9
MentorNet	13.3 ± 0.2	18.1 ± 0.4	—	66.0 ± 4.9
Co-Teaching	11.2 ± 0.2	13.5 ± 0.2	19.3 ± 0.2	80.7 ± 1.2
D2L	12.3 ± 0.5	15.6 ± 0.6	27.3 ± 1.2	Diverged
L_{DMI}	14.1 ± 0.1	20.4 ± 1.6	34.9 ± 2.5	67.2 ± 2.9
DY-Bootstrap	20.6 ± 0.1	31.2 ± 2.8	43.6 ± 3.4	Diverged
INCV	10.5 ± 0.2	13.2 ± 0.2	18.9 ± 0.5	46.7 ± 3.8
AUM	9.8 ± 0.1	12.5 ± 0.1	17.9 ± 0.1	45.6 ± 3.3
Oracle	9.0 ± 0.1	9.7 ± 0.8	10.8 ± 1.8	12.6 ± 3.5

are assigned to the high-loss beta distribution are considered to be mislabeled. The authors also propose using a mixture of two Gaussian distributions (**DY-Bootstrap GMM**)—an approach also used by (Li et al., 2020). INCV (Chen et al., 2019) is an iterative approach that first filters training data through iterative cross-validation.⁵ The remaining samples are shared between two networks that inform each other about which training samples are most likely to be clean based on loss. These filtered samples, as well as the unshared examples, are identified as mislabeled. We perform all methods using ResNet-32 models. For our method (**AUM**), as well as the BMM and GMM methods, we train networks for 150 epochs with no learning rate drops. We fit the BMM and GMM models to losses from the last epoch. For INCV we use the publicly available implementation.

Results. Figure 4 displays the precision and recall of the various identification methods at different noise levels. The most challenging settings for all methods are CIFAR100 and Tiny ImageNet with low noise. We hypothesize that this may be due to mislabeled examples in the uncorrupted versions of these datasets (as evidenced by the results in Table 4). The AUM ranking achieves a recall of $> 95\%$ and a precision of $> 80\%$ in most settings. In fact, AUM tends to achieve the highest precision *and* recall in most noise settings. It is worth emphasizing that the AUM model achieves this high performance without any supervision or prior knowledge about the noise model.

4.2. Robust Training on Synthetic Noisy Datasets.

To further evaluate our proposed method, we train ResNet-32 models on the noisy datasets after discarding the identified mislabeled samples. We compare the performance of this approach (referred to as **AUM** in the tables) to several baselines. As a lower bound for performance, we train a

⁵ While this approach wasn’t explicitly designed to identify mislabeled data, it flags and discards potentially mislabeled samples. Therefore we consider it a baseline for comparison.

Table 2. Test-error performance on CIFAR100 (ResNet-32) with synthetic mislabeled samples (uniform noise model).

Noise	0.20	0.40	0.60	0.80
Standard	50.4 \pm 0.3	62.5 \pm 0.3	76.2 \pm 0.8	91.8 \pm 0.5
Random	43.0 \pm 0.1	51.2 \pm 0.5	63.5 \pm 0.9	83.7 \pm 1.8
Bootstrap	48.6 \pm 0.3	58.9 \pm 0.5	70.3 \pm 0.4	89.8 \pm 1.6
MentorNet	35.8 \pm 0.6	42.5 \pm 0.4	—	75.7 \pm 1.4
Co-Teaching	35.9 \pm 0.2	39.8 \pm 0.4	52.0 \pm 0.5	89.1 \pm 1.4
D2L	46.0 \pm 2.0	70.3 \pm 3.6	Diverged	Diverged
DY-Bootstrap	47.0 \pm 0.8	57.0 \pm 0.7	63.4 \pm 0.9	87.2 \pm 0.9
INCV	41.4 \pm 0.9	44.6 \pm 0.4	56.3 \pm 0.6	76.3 \pm 1.1
AUM	34.5 \pm 0.3	38.7 \pm 0.2	47.0 \pm 0.9	68.3 \pm 1.3
Oracle	35.5 \pm 0.1	39.0 \pm 0.4	44.8 \pm 1.0	55.1 \pm 0.7

Table 3. Test-error performance on Tiny ImageNet (ResNet-32) with synthetic mislabeled samples (uniform noise model).

Noise	0.20	0.40	0.60	0.80
Standard	58.6 \pm 0.3	66.3 \pm 0.4	77.6 \pm 0.2	92.4 \pm 0.1
Random	54.3 \pm 0.2	60.8 \pm 0.3	70.7 \pm 0.5	87.1 \pm 2.0
DY-Bootstrap	58.2 \pm 0.1	63.7 \pm 0.5	76.8 \pm 0.0	93.2 \pm 0.8
INCV	54.8 \pm 0.3	57.4 \pm 0.2	63.8 \pm 0.5	82.6 \pm 1.3
AUM	51.1 \pm 0.4	55.3 \pm 0.3	62.7 \pm 0.6	79.2 \pm 0.4
Oracle	52.7 \pm 0.6	56.5 \pm 0.4	63.0 \pm 0.6	71.2 \pm 0.3

ResNet-32 that follows a **Standard** training procedure on the full dataset. As an upper bound, we train an **Oracle** ResNet-32 only on the correctly-labeled portion of data. We do not perform early stopping since we do not assume the presence of a clean validation set. These three methods use the ResNet training procedure described in (He et al., 2016).

Baselines. We train models using the **INCV** and **DY-Bootstrap**⁶ methods described in Section 4.1. **Bootstrap** (Reed et al., 2014) and **D2L** (Ma et al., 2018) interpolate the one-hot target label with the predicted label. **MentorNet** (Jiang et al., 2017) learns a weighting scheme for training examples using an LSTM.⁷ We also compare to a **Random Weighting** (Ren et al., 2018) baseline, where each sample is assigned a random weight. The weights are drawn from a rectified normal distribution and are re-drawn at every epoch. All methods use ResNet-32 models and the hyperparameters defined in their publicly available implementations. We note that a number of these methods can be used in conjunction with semi-supervised learning (Li et al., 2020; Nguyen et al., 2020) or MixUp data augmentation (Zhang et al., 2018; Arazo et al., 2019). Given that our focus is on identifying mislabeled examples, we consider these to be complimentary orthogonal approaches. Therefore, we do not utilize these additional training procedures with our

⁶ We compare to the DY-Bootstrap variant proposed by (Arazo et al., 2019) that does not use MixUp to disentangle the performance benefits of mislabel identification and data augmentation.

⁷ Due to incomplete release, we cannot train MentorNet (Jiang et al., 2017) in some noise settings.

method or any of the baselines in Tables 1-3.

Results. Table 1 and Table 2 display the test set performance of the various methods on corrupted versions of CIFAR10/100. We observe several trends. First, there is a large performance gap between the Oracle and Standard baselines (up to 62%), suggesting that the standard networks overfit to mislabeled examples. Most of the methods reduce this performance gap significantly. Our identification scheme (AUM) achieves the lowest error in most settings. On the 20% and 40% CIFAR10 variants, AUM recovers the oracle performance, suggesting that our cleaned dataset is nearly as good as the oracle dataset. In fact, our method *surpasses* oracle performance on CIFAR100 with 20% and 40% label noise—simply by removing samples from the training set. We hypothesize that our identification procedure identifies mislabeled or highly-ambiguous samples that are in the standard (uncorrupted) training set.

In addition, we test our method’s performance on Tiny ImageNet (Table 3). We compare against DY-Bootstrap and INCV, which are two of the most recent methods that use identification. This dataset is notably more challenging as it contains 200 classes; a baseline model on the uncorrupted dataset achieves only 50% accuracy. Nevertheless, our identification method cleans datasets that match (and surpass) the oracle performance in most noise settings.

4.3. Real-World Datasets

In real-world scenarios, data quality may be unknown a priori. Some datasets may have many mislabelings whereas others might be mostly correct. Ideally a method should work in both extremes: finding mislabeled samples in noisy datasets while leaving clean datasets relatively unchanged.

Weakly-Labeled Datasets. We test the performance of our method on two datasets where the label-noise is unknown. WebVision (Li et al., 2017a) contains 2 million images scraped from Flickr and Google Image Search. It contains no human annotation (labels come from the scraping search queries), and therefore we expect many mislabeled examples. Similar to prior work (e.g. Chen et al., 2019) we train on a subset, **WebVision50**, that contains the first 50 classes (\approx 100,000 images). Clothing1M (Xiao et al., 2015) contains 1 million images of clothes from 14 categories. Similarly to Webvision, most images are scraped, though the dataset also contains “trusted” (i.e. hand-labeled) images. We use a 100,000 sample subset which we refer to as **Clothing100k**. In order to be consistent with the other datasets, we don’t use any trusted-set images for training or validation. We train ResNet-50 models on these datasets. As with our Tiny ImageNet experiments, we compare against the recent methods of DY-Bootstrap and INCV.

Table 4. Test-error performance on real-world datasets.

	WebVision50		Clothing100K		CIFAR10		CIFAR100		Tiny ImageNet	
	Error	(% Removed)	Error	(% Removed)	Error	(% Removed)	Error	(% Removed)	Error	(% Removed)
Standard	21.4	(0.0)	35.8	(0.0)	8.1 ± 0.2	(0.0)	33.0 ± 0.6	(0.0)	49.3 ± 0.5	(0.0)
DY-Bootstrap	25.8	(4.6)	38.4	(12.1)	10.0 ± 0.1	(15.5 ± 0.6)	34.9 ± 0.3	(7.3 ± 0.3)	51.6 ± 0.1	(12.5 ± 0.2)
INCV	22.1	(26.2)	33.3	(25.2)	9.1 ± 0.2	(8.4 ± 0.4)	38.2 ± 0.3	(27.4 ± 0.2)	56.1 ± 0.3	(27.6 ± 4.8)
AUM	19.8	(17.8)	33.5	(16.7)	7.9 ± 0.0	(3.0 ± 0.1)	31.8 ± 0.2	(13.0 ± 1.8)	48.1 ± 0.1	(24.1 ± 0.7)

We use AUM/indicator samples to remove mislabeled training samples and re-train on the cleaned dataset. In Table 4, we compare this approach (AUM) to a model trained on the full dataset (Standard). Our identification procedure significantly reduces the error on both datasets. On WebVision50 we flag 17.8% of the data as mislabeled (see Appendix C for examples). Removing these samples reduces error from 21.4% to 19.8%. Similarly, we identify 16.7% mislabeled samples on Clothing100K for a similar error reduction. We offer further analysis of these datasets in Appendix C.

In comparison, we find that the DY-Bootstrap method tends to estimate less label noise than our method. Moreover, it is unable to reduce the error over the standard training procedure. DY-Bootstrap mixes the assigned and predicted labels during training; therefore, we hypothesize that it is overconfident with its identifications. INCV tends to remove more examples than our method. It performs roughly the same as our method on Clothing100k. However, on WebVision50 it achieves higher error than the standard model, suggesting that it is pruning samples too aggressively on this dataset.

“Mostly-Clean” Datasets. We also test the *uncorrupted* versions of CIFAR10, CIFAR100, and Tiny ImageNet. The images in these datasets are small, so it is possible that some are ambiguous or mislabeled. However, we expect that most images are correctly-labeled. We train ResNet-32 models following the same procedures in Section 4.2. In Table 4 we notice several trends. First, using DY-Bootstrap or INCV on these datasets results in worse performance than a standard training procedure. INCV flags over a quarter of CIFAR100 and Tiny ImageNet as mislabeled, and therefore likely throws away too much data. DY-Bootstrap tends to remove fewer examples than INCV; however, it is again likely that the bootstrap loss is overconfident. In contrast, our cleaning procedure *reduces the error* on these datasets. (See Appendix C for examples of removed images.) Simply by removing high AUM data, we reduce the error on CIFAR100 from 33.0% to 31.8%.

The number of samples removed by our method differs among the datasets: 3% on CIFAR10, 13% on CIFAR100, and 24% on Tiny ImageNet. We hypothesize that the finer granularity of classes in CIFAR100 and Tiny ImageNet make these datasets more prone to mislabeled/ambiguous samples, especially given the image size. We run our method

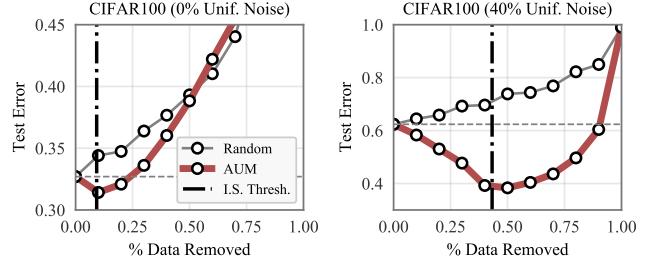


Figure 5. Removing data according to the AUM ranking (red line). A ResNet-32 achieves best test error when the amount of removed data corresponds to the indicator sample threshold (black line). Removing data randomly results in strictly worse error (gray line).

on the full ImageNet dataset and find that only 2% of samples are flagged. Removing these samples does not significantly change top-1 error (from 24.2 to 22.4). Given the rigorous annotation process of ImageNet (Deng et al., 2009) it is not surprising that we find few mislabeled samples.

4.4. Ablation Studies

We perform several ablation studies here and in Appendix B.

Removing data according to the AUM ranking. Our method removes data that have a lower AUM than the indicator sample threshold. First, we study the effect of removing fewer samples (lower threshold), more samples (higher threshold), or random samples. Specifically, we discard varying amounts of the CIFAR100 dataset and compare models trained on the resulting subsets. We examine two settings: discarding data in order of their AUM ranking and according to a random permutation. Figure 5 displays test error performance as a function of dataset size. Discarding data at random strictly increases test error regardless of threshold. On the other hand, discarding data according to AUM ranking (red line) results in a distinct optimum. This optimum corresponds with the indicator sample threshold (black dotted line), suggesting that our proposed method identifies samples harmful to generalization and keeps the data required for good performance.

Consistency across architectures. We find that our method is robust to architecture choice. The AUM ranking achieves $> 98\%$ Spearman’s correlation across networks of

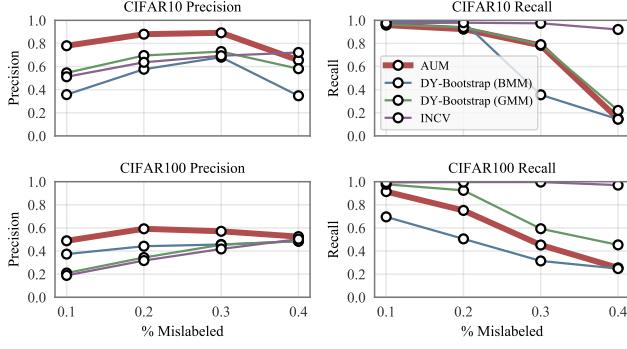


Figure 6. Precision/recall for identifying mislabeled data under asymmetric noise models. Results are from ResNet-32 models.

various depth and architecture (see Appendix B for details). This suggests that the AUM statistic captures properties that are dataset-dependent rather than model-dependent.

4.5. Limitations

Our proposed method is able to identify mislabeled examples and reduce error in many settings—including real-world datasets. Nevertheless, we can construct settings that are challenging for AUM/indicator samples. Consider a setting where mislabelings are extremely systematic: for example, all images with BIRDS either are assigned the (correct) label BIRD or the (incorrect) label DOG (i.e. they are never mislabeled as any other class). To test such a setting in practice, we corrupt labels using an *asymmetric* noise model (e.g. Chen et al., 2019). If we assume some ordering of the classes $[1, c]$, then—with probability p —we alter a sample’s assigned label y from its ground-truth class \tilde{y} to the next class adjacent class $\tilde{y} + 1$. Formally,

$$p_{\text{Asymmetric}}(y \mid \tilde{y}) = \begin{cases} (1 - p) & y = \tilde{y} \\ p & y = (\tilde{y} + 1). \end{cases}$$

Note that under this noise model, a dataset with 50% noise cannot afford performance better than random guessing. We expect that the indicator samples might return a threshold that is too low in this setting. Assume that BIRDS are mislabeled as DOGS with probability $p = 0.4$. For the average image with a bird, the model will likely predict BIRD with $\approx 60\%$ confidence and DOG with $\approx 40\%$ confidence. In other words, the BIRD and DOG logits will be rather similar. Compared with the uniform noise setting, correctly-labeled birds will have a smaller margin and mislabeled birds will have a larger margin. For indicator samples however, the model confidence for the assigned class will be $\approx 1/(c + 1)$ —representing the frequency of these samples. Indicator samples margins will therefore be smaller than mislabeled margins, resulting in low identification recall.

In Figure 6 we see that the recall of our proposed method

Table 5. Test-error (ResNet-32) on datasets w asymmetric noise.

Dataset Noise	CIFAR10		CIFAR100	
	0.20	0.40	0.20	0.40
Standard	23.7 ± 0.2	43.7 ± 0.0	47.1 ± 0.4	61.8 ± 0.0
Random	16.1 ± 0.5	34.8 ± 2.7	42.3 ± 0.5	57.1 ± 0.4
Bootstrap	23.8 ± 0.4	45.0 ± 1.2	46.6 ± 0.6	61.3 ± 0.6
D2L	11.4 ± 0.5	23.6 ± 3.0	56.4 ± 1.4	83.1 ± 2.3
L_{DMI}	13.3 ± 1.7	16.0 ± 4.2	Diverged	Diverged
DY-Bootstrap	22.1 ± 0.1	40.6 ± 1.1	46.8 ± 0.1	62.1 ± 0.1
INCV	11.7 ± 0.2	20.2 ± 0.7	43.2 ± 0.2	55.6 ± 1.3
AUM	10.3 ± 0.2	41.3 ± 0.5	40.3 ± 0.4	59.8 ± 0.1
Oracle	9.2 ± 0.3	10.8 ± 0.2	35.3 ± 0.4	38.8 ± 0.5

struggles in the 30% and 40% noise settings. We would note that 40% noise is a nearly maximal amount of noise under this noise model, and that the BMM and GMM approaches have a similarly low recall. Our model achieves higher precision than all other methods, and also achieves comparable recall in the 10%-20% settings. Table 5 displays test error after discarding data flagged by our method.⁸ In the 40% noise setting our method is not competitive with the best method (INCV).⁹ A different indicator sample construction—one specifically designed for this noise model—might result in a better AUM threshold that makes our method more competitive. However, given that our approach achieves significant error reductions on real-world datasets (Table 4), we hypothesize that this particular high-noise setting is not too common in practice. In addition, our method outperforms all other methods in the 20% noise setting.

5. Discussion and Conclusion

In this paper we introduce the AUM statistic and the method of training with indicator samples. Together, these contributions reveal differences in training loss dynamics that reliably identify noisy labels with high precision and recall. This method is efficient to implement, resulting in low error on large-scale image classification datasets. We also observe that the training procedure is far more consistent across architectures than metrics such as validation loss. Our approach can be combined with other robust training methods that use data augmentation and/or semi-supervised learning. While our approach is highly effective at identifying mislabeled samples and improving dataset quality, we believe that its ability to detect suspect samples may open applications beyond the scope of this paper. As future work, we plan to investigate the use of AUM to estimate sample “hardness” for use in curriculum learning (Bengio et al., 2009) and to detect mis-predictions in the context of semi-supervised learning (Chapelle et al., 2009).

⁸ For baselines, we compare against methods that—like our approach—have no prior knowledge of the noise model. This excludes co-teaching, which requires a noise estimate.

⁹ No method is significantly better than random on CIFAR100.

References

- Algan, G. and Ulusoy, I. Image classification with deep learning in the presence of noisy labels: A survey. *arXiv preprint arXiv:1912.05170*, 2019.
- Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *NeurIPS*, 2019.
- Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., and McGuinness, K. Unsupervised label noise modeling and loss correction. In *ICML*, 2019.
- Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *ICML*, 2019.
- Arpit, D., Jastrz̄bski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In *ICML*, pp. 233–242, 2017.
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.
- Chapelle, O., Scholkopf, B., and Zien, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.
- Chen, P., Liao, B., Chen, G., and Zhang, S. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*, 2019.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. *CVPR*, pp. 248–255, 2009.
- Goldberger, J. and Ben-Reuven, E. Training deep neural networks using a noise adaptation layer. In *ICLR*, 2017.
- Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. W.-H., and Sugiyama, M. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 2018.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.
- Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018.
- Hu, W., Li, Z., and Yu, D. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *NeurIPS Workshop on Machine Learning with Guarantees*, 2019.
- Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L., and Weinberger, K. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. doi: 10.1109/TPAMI.2019.2918284.
- Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mennet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2017.
- Joulin, A., van der Maaten, L., Jabri, A., and Vasilache, N. Learning visual features from large weakly supervised data. In *ECCV*, pp. 67–84, 2016.
- Krizhevsky, A. et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- Li, J., Socher, R., and Hoi, S. C. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2020.
- Li, M., Soltanolkotabi, M., and Oymak, S. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks, 2019. URL <https://arxiv.org/abs/1903.11680>.
- Li, W., Wang, L., Li, W., Agustsson, E., and Gool, L. V. Webvision database: Visual learning and understanding from web data, 2017a. URL <https://arxiv.org/abs/1708.02862>.
- Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, J. Learning from noisy labels with distillation. *ICCV*, pp. 1928–1936, 2017b.
- Luo, Y., Zhu, J., Li, M., Ren, Y., and Zhang, B. Smooth neighbors on teacher graphs for semi-supervised learning. In *CVPR*, pp. 8896–8905, 2018.
- Ma, X., Wang, Y., Houle, M. E., Zhou, S., Erfani, S. M., Xia, S.-T., Wijewickrema, S. N. R., and Bailey, J. Dimensionality-driven learning with noisy labels. *ICML*, 2018.
- Nguyen, D. T., Mummadli, C. K., Ngo, T. P. N., Nguyen, T. H. P., Beggel, L., and Brox, T. Self: Learning to filter noisy labels with self-ensembling. In *ICLR*, 2020.
- Ortego, D., Arazo, E., Albert, P., O'Connor, N. E., and McGuinness, K. Towards robust learning with different label noise distributions. *arXiv preprint arXiv:1912.08741*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pp. 8024–8035, 2019.

Reed, S. E., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. *CoRR*, abs/1412.6596, 2014.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

Shen, Y. and Sanghavi, S. Learning with bad training data via iterative trimmed loss minimization. In *ICML*, 2019.

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L. D., and Fergus, R. Training convolutional networks with noisy labels. In *ICLR Workshop*, 2015.

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. Learning from massive noisy labeled data for image classification. In *CVPR*, pp. 2691–2699, 2015.

Xu, Y., Cao, P., Kong, Y., and Wang, Y. L_dmi: An information-theoretic noise-robust loss function. *arXiv preprint arXiv:1909.03388*, 2019.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhang, Z. and Sabuncu, M. R. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

Zhang, Z., Zhang, H., Arik, S. O., Lee, H., and Pfister, T. Ieg: Robust neural network training to tackle severe label noise. *arXiv preprint arXiv:1910.00701*, 2019.

Supplementary Materials for: Identifying Mislabeled Data using the Area Under the Margin Ranking

A. Experiment Details

All experiments are implemented in PyTorch (Paszke et al., 2019). Since we don't assume the presence of trusted validation data, we do not perform early stopping. All test errors are recorded on the model from the final epoch of training.

CIFAR10, CIFAR100, and Tiny ImageNet. All models unless otherwise specified are ResNet-32 models that follow the training procedure of He et al. (2016). We train the models for 300 epochs using 10^{-4} weight decay, SGD with Nesterov momentum, a learning rate of 0.1, and a batch size of 256. The learning rate is dropped by a factor of 10 at epochs 150 and 225. The ResNet-32 model is designed for 32×32 images. For Tiny ImageNet (which is 64×64), we add a stride of 2 to the initial convolution layer.

When computing the AUM to identify mislabeled data, we train these models up until the first learning rate drop (150 epochs). We additionally drop the batch size to 64 to increase the amount of variance in SGD. We find that this variance decreases the amount of memorization, which makes the AUM metric more salient. All other hyperparameters are consistent with the original training scheme.

After removing samples identified by AUM/indicator samples, we modify the batch size so that the network keeps the same number of iterations as with the full dataset. For example, if we remove 25% of the data, we would modify the batch size to be 192 (down from 256).

WebVision50 and Clothing100k. We train ResNet-50 models on these two datasets from scratch. Almost all training details are consistent with He et al. (2016)— 10^{-4} weight decay, SGD with Nesterov momentum, initial learning rate of 0.1, and a batch size of 256. The only difference is the length of training. Because the datasets are smaller than ImageNet, we train the models for 180 epochs. We drop the learning rate at epochs 60 and 120 by a factor of 10.

When computing the AUM, we train these models up until the first learning rate drop (60 epochs) with a batch size of 256. As with the smaller datasets, we keep the number of training iterations constant after removing high AUM examples.

ImageNet. The ImageNet procedure exactly matches the procedure for WebVision and Clothing100K, except that we only train for 90 epochs, with learning rate drops at 30 and 60. The AUM is computed up until epoch 30.

B. Additional Ablation Studies

Consistency across architectures. Since AUM functions like a ranking statistic, we compute the Spearman's correlation coefficient between the different networks. In Figure S1 far left we compare CIFAR10 (40% noise) AUM values computed from ResNet and DenseNet (Huang et al., 2019) models of various depths. We find that the AUM ranking is *essentially the same* across these networks, with $> 98\%$ correlation between all pairs of networks. It is worth noting that AUM achieves this consistency in part because it is a running average across all epochs. Without this running average, the margin of samples only achieves roughly 75% correlation (middle left plot). Finally, AUM is more consistent than other metrics used to identify mislabeled samples. The training loss (middle right plot), used by Arazo et al. (2019), achieves 75% correlation. Validation loss (far right plot), used by INCV (Chen et al., 2019), achieves 40% correlation. These metrics are more susceptible to network variance, which in part explains why AUM achieves higher identification performance.

AUM vs. margin. We observe that integrating margin values over time is necessary for the consistency of the metric. Figure S1 plots the inter-network correlation of the margin at epoch 150 for training samples (the last epoch before the

Identifying Mislabeled Data using the Area Under the Margin Ranking

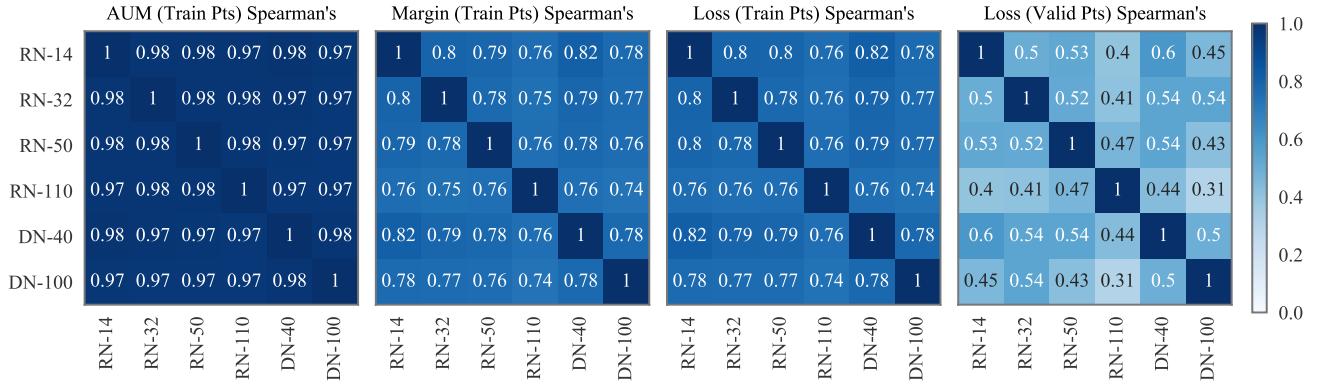


Figure S1. Spearman's correlation of AUM and other metrics across various network architectures (RN=ResNet, DN=DenseNet). AUM (left) produces a very consistent ranking of the training data. The margin itself (middle left) produces less consistent rankings. Training loss (middle right) and validation loss (far right) are also much less correlated across networks.

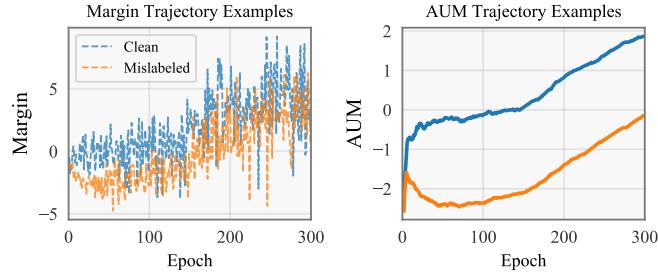


Figure S2. Left: Margin trajectories over 300 training epochs for one clean sample and one mislabeled CIFAR example. The trajectories are difficult to separate consistently. **Right:** AUM trajectories (running average of margin) are separable and less noisy.

learning rate is dropped). Most networks achieve a correlation between 0.79 and 0.85—roughly 10 percentage points less than training AUM. **Figure S2** shows a similar qualitative effect for a fixed architecture across training epochs. On the left, stochastic training dynamics result in margin trajectories that are both noisy and non-monotonic. Note that margins for clean and mislabeled examples occupy a similar range of values, have similar variation, and intersect several times throughout training. On the right, we see that AUM produces a consistent separation after the first few epochs.

C. More Results for Real-World Datasets

AUM values. **Figure S3** displays the empirical AUM densities on the real-world datasets. Unlike the synthetic mislabeled datasets (**Figure 3**, main text) these datasets do not exhibit bimodal behavior. The indicator sample threshold—represented by a gray line—differs for all datasets.

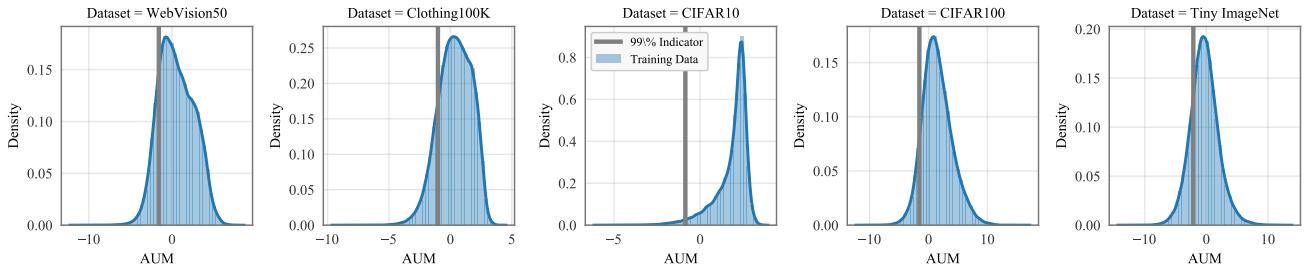


Figure S3. AUM distributions on real-world datasets. Gray lines represent the threshold learned by indicator samples.

Example removed images. **Figure S4** displays high AUM images for CIFAR10, CIFAR100, and Tiny ImageNet. **Figure S5** and **Figure S6** display high AUM images for WebVision50, and Clothing100K, respectively.

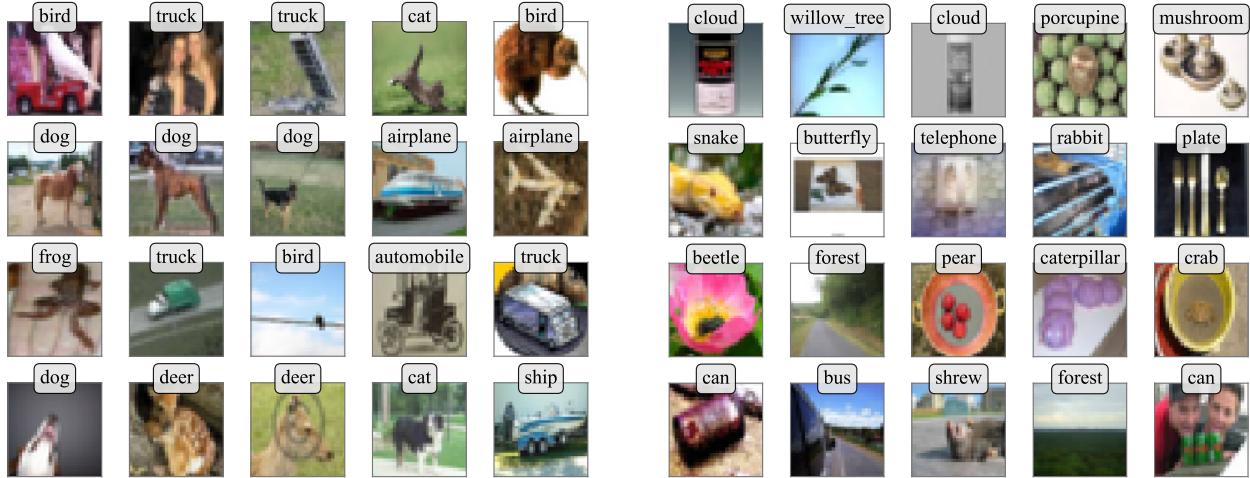


Figure S4. Images from CIFAR10 (left) and CIFAR100 (right) with the worst AUM ranking.

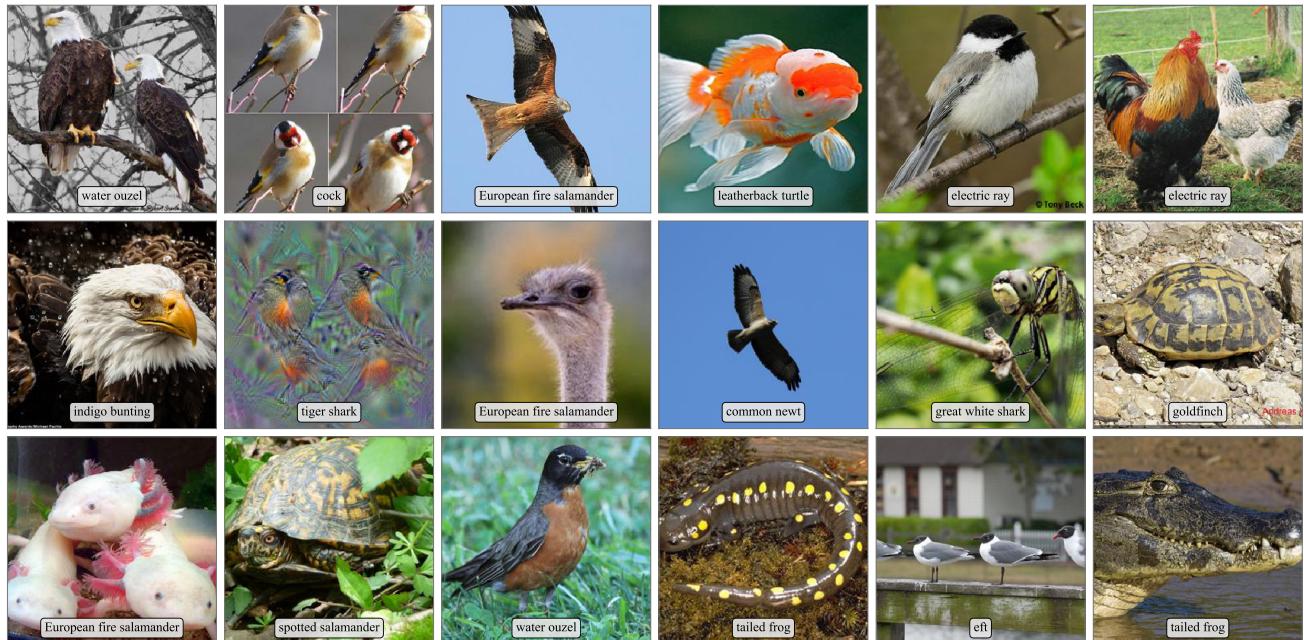


Figure S5. Images from WebVision50 with the worst AUM ranking.

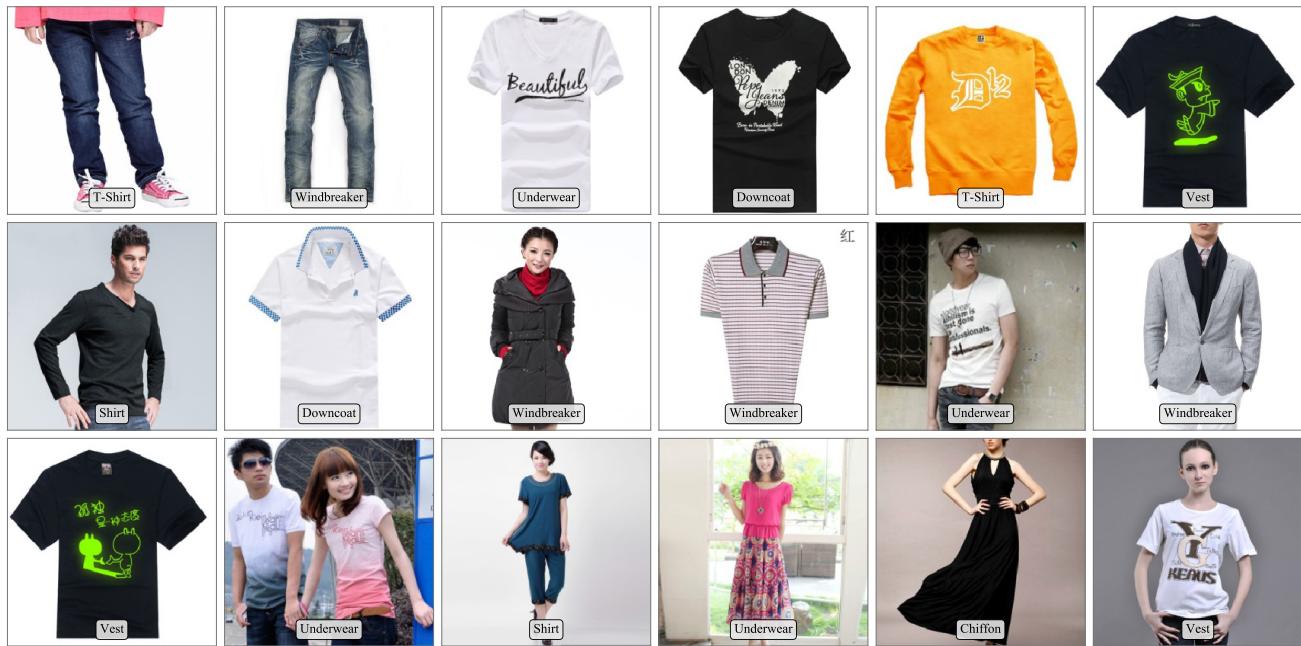


Figure S6. Images from Clothing100K with the worst AUM ranking.