Revisiting sample weights based method for noisy-label detection and classification

Tuan Hoang, Hung Tran, Santu Rana, Sunil Gupta, and Svetha Venkatesh

Deakin University, A2I2
{tuan.h;tduy;santu.rana;sunil.gupta;svetha.venkatesh}@deakin.edu.au

Abstract. The remarkable success of Convolutional Neural Networks (CNNs) in image classification can be attributed large clean training datasets. However, real-world data is often far from noise-free, impacting the performance of resulting deep neural network (DNN) models. Existing literature focuses on noisy label detection, often drawing a clear line between noisy and clean label samples. Nevertheless, each sample contributes differently to the final model performance; some noisy-label samples may still be valuable to a certain level, while certain clean-label samples might not significantly enhance the model. In this work, assuming that a small clean-label dataset may be available, we aim to learn a sample weight for each training sample. This weight is gradually updated as the model is training to indicate the usefulness of a particular sample in minimizing loss with respect to the clean-label dataset. Consequently, our method prioritizes high-quality data samples, minimizing the impact of harmful or unhelpful ones by assigning close-to-zero weights in a weighted loss function. We empirically demonstrate that our method is not dependent on noise type and can work well for both real-world and synthetic noise. Our method can achieve state-of-the-art performance in terms of the classification accuracy on clean test sets.

Keywords: Noisy-label · Classification · Gradient-Through-Gradient

1 Introduction

Deep learning has been widely adopted across various fields, such as computer vision, natural language processing, and image/music/video generation. One of the reasons deep models excel is their ability to leverage vast quantities of data for training. However, data annotation can introduce label noise due to various reasons, for instance, due to differing degrees of expertise or measurement error, unclear criteria, e.g., medical examples. Training a model with noisy labels could be detrimental, as networks have been shown to fit the mislabeled training examples (*i.e.* memorization). Thus, mitigating the effects of noisy labels becomes a critical issue that needs careful treatment. The most straightforward solution would be to clean up the corrupted-label samples by either discarding or correcting them before training. However, this process can be very expensive and time-consuming. Therefore, there is a great amount of effort that has been spent to develop algorithms to train models that are robust to label noise.

Earlier solutions often require a priori knowledge of the noise rates. However, these solutions may be difficult to work with in practice. Usually, one needs to estimate the noise rates from data. Therefore, recent works provide solutions where no reliable estimate of the noise rates is available. Many of existing works usually do not consider the fact that different samples can have different effects on the final model. Some samples with clean labels but being outliers (*i.e.*, the sample is drawn from a distribution that is different from the test dataset distribution) can have detrimental effect to the model. Ignoring these kind of training samples would result in better performance for the final model. Additionally, corrupted-label samples are not equally harmful. For example, an image labeled as a *bird* but actually depicting a *airplane* might still be very informative and helpful to the model compared to the same image but labeled as a *cow*.

L2RW [15] and MW-Net [16] try to address this aforementioned problem. However, their approaches can lead to suboptimal solutions. Specifically, in L2RW [15], the authors propose to learn a perturbation for each sample in a training mini-batch. However, the sample weights are learnt by only considering the current training and validation mini-batches. This usually results in very high variation in the sample weights across different mini-batches. Additionally, historical information about how helpful/harmful a training sample is not considered. Shu *et al*. [16] proposed MW-Net which learns a multilayer perceptron (MLP) network to map the loss value of a training sample to its weight. However, it is possible for clean and noisy samples to have similar loss values. Therefore, this approach can result in a suboptimal solution, especially for instance-dependence noise as observed in our experiments.

In this paper, we revisit the idea of learning sample weights proposed in [15, 16]. Differently, we propose to explicitly learn a scalar weight value for each training sample and accumulate their updates as the training progresses. This allows the samples, which are more helpful in minimizing the loss values of the validation dataset and ideally the testing dataset as well, will have higher and higher weight values; and the harmful samples (e.g., noisy or outlier samples) would be assigned lower and lower weights. As the result, the model will focus more on the clean samples and minimize the impact of noisy samples on the model.

Our main contributions can be summarized as follows:

- We have reexamined the concept of learning sample weights, proposing a novel approach that explicitly learns a scalar value for each training sample. This scalar value accumulates the positive or negative impact of the sample on the model concerning a clean dataset as the training progresses. This helps to effectively enhance the influence of helpful samples while minimizing the impact of harmful ones.
- Our proposed method demonstrates robust performance across various types of noisy labels without the need for prior knowledge about the noise, such as its type and ratio. Experimental results highlight that our method is independent of distributional assumptions on the noise distribution. Specifically, with the same noise ratio, our method consistently achieves comparable perfor-

mance across different synthetic noise types, including instance-dependence noise, on both CIFAR-10 and CIFAR-100 datasets.
– In comparison with state-of-the-art approaches, our method achieves favorable performance on the real-world noisy classification benchmark Clothing1M [20].

2 Related works

Existing approaches related to making model robust to noise label can be categorized into two main types: (1) detecting corrupted labels and then cleansing potential corrupted labels, and (2) directly training noise robust models with noisy labels.

(1) Noise-cleansing-based Approaches attempt to detect corrupted labels (*i.e.*, sample selection) and then cleanse potential corrupted labels or reduce their impacts on subsequent training. Several works ultilize k-NN based technique to filter out corrupted samples when learning with noisy labels. Kong *et al.* [9] and Bahri *et al.* [1] independently demonstrate that, despite training with a noisy dataset, the neural network still produces representations that are beneficial for effective deep k-NN filtering. Li *et al.*, [11] propose DivideMix which first dynamically divide training data into a labeled set with clean samples and an unlabeled set with noisy samples, then simultaneously training two diverged models in a semi-supervised manner on both sets to avoid confirmation bias. Zhu *et al.*, [24] propose to identify corrupted labels through two algorithms: a voting-based local detection method, which checks noisy label consensuses of nearby features, and a ranking-based global detection method, which scores instances based on their likelihood of being clean and filters out a percentage of instances with low scores as corrupted. Yu *et al.*, [22] apply a classical multiple hypothesis testing algorithm, the Benjamini-Hochberg (BH) procedure, to detect corrupted samples.

(2) Noise-robust Approaches aim to design a robust classifier with noisy labels, including by utilizing explicit regularizations designing robust architectures or improving the loss functions to design a robust classifier. A common approach involves loss correction, wherein the initial step entails estimating a transition matrix [13,17,21,25]. Subsequently, correction or reweighting is executed through forward/backward propagation. Alternatively, the estimated transition matrix can be further adjusted with controllable variations [19]. Instead of decide the early stopping point by considering a DNN as a whole, Bai *et al.,* [2] propose to progressively apply early stop for different layers at different point to reduce the memorization effect of DNN on noisy samples. Rather than determining the early stopping point by evaluating the entire DNN collectively, Bai *et al.* [2] suggest applying early stopping at various points for different layers to mitigate the memorization effect of the DNN on noisy samples. Several works in the literature presume the presence of a limited set of clean data, utilizing it to train classifiers resilient to label noise. Li *et al.* [12] propose to the knowledge acquired from a small clean dataset to enhance the learning process of another model using the

entire noisy dataset. Hendrycks *et al.* [6] utilize clean data by proposing a loss correction technique that utilizes clean examples in a data-efficient manner to mitigate the effects of label noise on deep neural network classifiers. Zhang *et al.*, [23] leverage a small set of clean data to estimate the exemplar weights and labels, then train models in a supervised manner that is highly invulnerable to label noise. L2RW [15] and MW-Net [16] proposed to learn sample weights that help to minimize the loss of a validation mini-batch. Based on L2RW and MW-Net, Zhang *et al.,* [14] proposed to learn to address the dependence on the given clean dataset, which might be biased. MentorNet [7] learns time-varying weights for each training sample so that samples appear in a meaningful order that facilitate the training. Kim *et al.,* [8] utilized the concept of Negative Learning (NL) to address the noisy data classification. NL has been proven to be effective in preventing the CNN from overfitting to noisy data. Cheng *et al.,* [3] propose a confidence regularization for cross-entropy loss, which theoretically guarantee to avoid overfitting to instance-dependent label noise.

We note that our proposed method can be categorized as both detecting corrupted labels and training noise-robust models with noisy labels. This is because the sample weights not only serve to filter out noisy samples but it also contribute to mitigating the harmful affects of noisy samples.

3 Proposed method

3.1 Problem statement

We are considering a classification problem involving a set of N training samples represented as $D := (x_i, y_i)_{i=1}^N$. The samples (x_i, y_i) are drawn from random variables $Z = (X, Y) \in \mathcal{X} \times \mathcal{Y}$ following a joint distribution $\mathcal{D}$. The primary objective of the classification task is to discover a classifier $f : \mathcal{X} \to \mathcal{Y}$ that accurately maps $\mathcal{X}$ to $\mathcal{Y}$. A classifier can be obtained by minimizing the empirical risk with respect to the cross-entropy loss defined as:

$$\mathcal{L}_{\text{CE}}(f(x), y) = -\log \mathbb{P}(y|x) = -\log \frac{\exp(f_y(x))}{\sum_{i=1}^K \exp(f_i(x))}.$$

However, in many real-world applications, we often encounter scenarios where our training set is noisy. In this situation, the learner encounters noisy samples $\tilde{Z} = (X, \tilde{Y})$, where $\tilde{Y}$ represents a noisy version of Y. More specifically, the classifier works with a noisy dataset $D^n = \{(x_n, \tilde{y}_i)\}_{i=1}^N$, sampled *i.i.d.* from the noisy distribution $\tilde{\mathcal{D}}$. The label $\tilde{y}_i$ is deemed *corrupted* if $\tilde{y}_i \neq y_i$, and *clean* otherwise. We aim to learn a classifier that is robust to noisy-label samples. Following [22, 24], we focus on the closed-set label noise scenario, in which Y and $\tilde{Y}$ are assumed to be in the same label space. Additionally, it is assumed that a set of clean data $D^c = \{(\boldsymbol{x}_j^c, y_j^c)\}_{j=1}^m$ drawn from $\mathcal{D}$ is also available and $|D^c| < |D^n|$ (practically $|D^c| \ll |D^n|$).

3.2 Learning sample weights

In the typical training process, every training sample holds an equal weight in determining the loss value, which is used to update the model via gradient

computation. However, in this setup, the presence of noisy samples substantially degrades the training model quality by introducing considerable noise into the gradients. Additionally, individual samples vary in their contribution to the final model's performance, where as clean but outlier samples may sometimes have a negative impact. Conversely, noisy samples are not universally detrimental.

To tackle this challenge, we introduce a novel approach that involves learning a scalar value $\alpha_i \in [0, 1]$ for each training sample. This scalar serves as a weight for the associated sample loss. Ideally, the unhelpful samples (*e.g.*, very noisy or outlier samples) will be assigned small weights ($\alpha_i \approx 0$), whereas helpful samples (*e.g.*, clean samples) will receive higher weights ($\alpha_i \approx 1$). Consequently, the training model will prioritize useful samples, effectively minimizing the impact of harmful samples.

The scalar value is trained in such a way that the weighted loss helps to reduce the loss with regard to the clean dataset D^c. More specifically, given the model parameters Θ and a mini-batch of training data $B^n \subset D^n$, the updated model parameter is a function of sample weights α_i^t at t-th time step as follows:

$$\Theta'(\alpha) = \Theta - \eta \nabla_\Theta \frac{1}{\sum_{i \in B^n} \alpha_i} \sum_{i \in B^n} \alpha_i \mathcal{L}\big(f_\Theta(\boldsymbol{x}_i), y_i\big), \tag{1}$$

where η is a learning rate. Then, given a mini-batch of the clean dataset $B^c \subset D^c$, we can update the training sample weights α_i such that they minimize the loss on the clean dataset as follows:

$$\alpha_i' = \alpha_i - \eta_\alpha \nabla_{\alpha_i} \sum_{j \in B^c} \mathcal{L}\big(f_{\Theta'(\alpha)}(\boldsymbol{x}_j^c), y_j^c\big). \tag{2}$$

Ideally, we want to update α_i such that they minimize the loss with regarding to the entire clean dataset. However, this nested optimization loop can be computationally expensive. We found that we can achieve good performance by using a moderate sized mini-batch. In our experiments, we opt for $|B^c| = 250$.

The process in Eq. 2 involves back-propagating the gradients of a clean mini-batch up to the sample weights α, by unrolling all model parameter gradients $\nabla_\Theta \sum_{i \in B^n} \alpha_i \mathcal{L}\big(f_\Theta(\boldsymbol{x}_i), y_i\big)$. Essentially, this operation involves a gradient through a gradient. From a computational perspective, it requires an additional backward pass through $\mathcal{L}\big(f_{\Theta'(\alpha)}(\boldsymbol{x}_j^c), y_j^c\big)$ to compute Hessian-vector products, which is fortunately supported by standard numerical computation libraries such as TensorFlow and PyTorch. We use sigmoid function to normalize the sample weights within the range of $[0, 1]$.

Additionally, during the initial epochs, when the model lacks sufficient discriminative capability, noisy samples might assist in reducing the loss concerning the clean dataset D^c as the model is still learning low-level features in lower layers [2]. Consequently, there is a risk of incorrectly updating the sample weight α_i during this phase. Furthermore, if a clean sample is erroneously assigned a small weight at the early stages, correcting this by assigning a higher weight later on can be challenging as the sample is nearly disregarded in the loss. Therefore, commencing the learning of sample weights too early could lead to sub-optimal

Algorithm 1: Learn Sample Weights (LSW).

Requires : Learning rate η and η_α, clean D^c and noisy D^n training datasets.

Parameters: Model parameters Θ, training sample weights $\{\alpha_i^{t=1} = 0.5\}_{i=1}^{N}$

for $t = 1$ **to** *max_iter* **do**
  Sample a mini batch of training samples $B^n = \{\boldsymbol{x}_i, y_i\}$ and their corresponding weights $\{\alpha_i^t\}$ from D^n.
  $\Theta^{t\prime}(\alpha) := \Theta^t - \eta \nabla_{\Theta^t} \frac{1}{\sum_{i \in B^n} \alpha_i^t} \sum_{i \in B^n} \alpha_i^t \mathcal{L}\big(f_{\Theta^t}(\boldsymbol{x}_i), y_i\big)$
  if $t \geq$ *start_iter* **then**
    Sample a mini batch of clean samples B^c from D^c.
    for $i \in B$ **do**
      $\alpha_i^{t+1} := \alpha_i^t - \eta_\alpha \nabla_{\alpha_i} \sum_{j \in B^c} \mathcal{L}\big(f_{\Theta^{t\prime}(\alpha)}(\boldsymbol{x}_j^c), y_j^c\big)$
      $\alpha_i^{t+1} := \text{sigmoid}(\alpha_i^{t+1} - 0.5)$
  $\Theta^{t+1} := \Theta^{t\prime}(\alpha)$

performance. In this work, we delay the updating of the sample weights for two epochs, i.e., $start\_iter = 2|D^n|/|B^n|$.

The sample weights α_i are jointly trained with the model parameters Θ. We summarize the training procedure in Algorithm 1. It is noteworthy that while a jointly trained model can achieve reasonably good classification performance, retraining the model from scratch with the learned sample weights could lead to better results. This is attributed to the negative impact of noisy samples on the model during the early stages when their weights are still significant. We further investigate this aspect in Ablation studies (Section 4.4).

Computation cost: As our method requires the gradient-through-gradient, which requires more GPU memory and computation. From a computational standpoint, as the weights well reflect their corresponding sample contribution, our method only needs one extra data sampling, forward/backward pass. In other words, our method does not require an additional forward pass for the newly-updated sample weights as in [15] and [16]. Hence, theoretically, given that $|B^c| = |B^n|$, our method needs $2\times$ the regular training time, which is computationally more efficienct than [15] and [16] as they need $3\times$ the regular training time. Regarding memory usage, we observe approximately 35% increase in memory requirements for storing intermediate representations.

4 Experiments

4.1 Experiment setting

Datasets and Baselines: To evaluate the effectiveness of our proposed method, following the convention evaluation [3], we conduct experiments on three benchmark datasets: CIFAR-10, CIFAR-100 [10] and Clothing1M. We use ResNet-34 [5] for CIFAR-10 and CIFAR-100, and ResNet-34/50 [5] for Clothing1M [20].

Table 1: Comparison of Noisy Label Detection metrics: FDR, Recall, F1 on CIFAR-10 with three types of synthetic noise. The top-2 results are highlighted in **bold**. The symbols ↓ and ↑ respectively denote that lower and higher values are preferable.

Noise Type	*Symmetric*				*Asymmetric*			*Instance*			
Noise Rate ε	0.3	0.4	0.5	0.6	0.3	0.4	0.5	0.3	0.4	0.5	0.6
	FDR ↓										
CORES^2	**8.87**	6.93	6.66	5.79	53.02	56.28	48.80	**10.25**	28.14	15.51	12.11
NLNL	**9.06**	**7.21**	**6.36**	**5.53**	**10.73**	**9.83**	35.35	**9.73**	17.55	13.30	9.67
SimiFeat-R	15.24	11.07	8.34	7.57	21.92	29.91	51.40	17.75	17.14	21.91	28.84
SimiFeat-R^c	34.26	26.05	26.55	22.13	40.79	40.19	50.79	34.67	28.04	23.61	18.49
BHN	17.30	13.86	13.36	10.26	16.82	13.84	**12.16**	17.84	**14.22**	**12.02**	10.11
Ours	9.57	**6.78**	**5.68**	**4.52**	**9.12**	**4.92**	**4.34**	10.56	**8.42**	**9.61**	**6.25**
	Recall ↑										
CORES^2	87.41	87.46	86.95	86.45	5.25	4.31	5.12	78.10	19.69	92.27	92.32
NLNL	**98.10**	**97.55**	**97.25**	**96.91**	**90.95**	**90.11**	**86.75**	**98.77**	92.89	93.93	**94.59**
SimiFeat-R	**98.46**	**98.42**	**98.14**	**97.69**	90.81	77.36	44.01	**96.35**	92.46	81.21	67.40
SimiFeat-R^c	95.66	96.03	97.19	97.80	89.09	74.90	51.89	95.07	**94.31**	**94.07**	90.08
BHN	59.05	62.53	77.30	79.71	52.54	62.54	74.61	55.65	58.42	64.69	69.80
Ours	94.31	92.47	92.53	90.02	**91.31**	**91.39**	**90.24**	95.80	**95.57**	**95.21**	**96.12**
	F1 ↑										
CORES^2	89.23	90.18	90.03	90.16	9.44	7.85	9.30	83.52	30.91	88.21	90.05
NLNL	**94.38**	**95.11**	**95.41**	**95.67**	**90.10**	**90.13**	74.08	**94.33**	87.35	**90.17**	**92.41**
SimiFeat-R	91.10	**93.44**	**94.79**	**94.99**	83.96	73.55	46.19	88.75	**87.39**	79.62	69.22
SimiFeat-R^c	77.93	83.56	83.66	86.70	71.14	66.51	50.51	77.44	81.63	84.31	85.58
BHN	68.90	72.46	81.70	84.42	64.40	72.47	**80.68**	66.35	69.50	74.56	78.58
Ours	**92.32**	92.84	93.41	92.66	**91.09**	**93.19**	**92.87**	**92.51**	**93.53**	**92.73**	**94.92**

As CIFAR-10 and CIFAR-100 datasets are clean, we divide the training set into two different sets, D^n (90%) and D^c (10%), and add synthetic noise into D^n. Three different types of synthetic label noise are generated: (i) Symmetric Noise flips labels uniformly to all other classes [8]. (ii) Asymmetric Noise is generated by pair-wise flipping, *i.e.*, randomly flipping the true label i to the next class $(i \mod K) + 1$. (iii) Instance-dependent Noise flips labels according to the probability that an example is mislabeled. This probability is computed based on the corresponding feature of the example [18].

The Clothing1M dataset [20] is a large real-world dataset, which is composed of clothing images crawled from online shopping websites. The dataset consists of one million noisy training samples (used as D^n), $47,570$ clean training samples (used as D^c), $14,131$ clean validation set, and $15,000$ clean testing set.

We compare our proposed method with recent state-of-the-art methods: MW-Net [16], CORES^2 [3], NLNL [8], SimiFeat-R [24], BHN [22]. As SimiFeat-R requires a model that is pretrained on ImageNet (ILSVRC2012) dataset [4]. For fair comparison with other methods, we also report SimiFeat-R results with the pretrained model which is trained on the clean dataset D^c only, denoted as SimiFeat-R^c.

Table 2: Comparison of Noisy Label Detection metrics: FDR, Recall, F1 on CIFAR-100 with three types of synthetic noise.

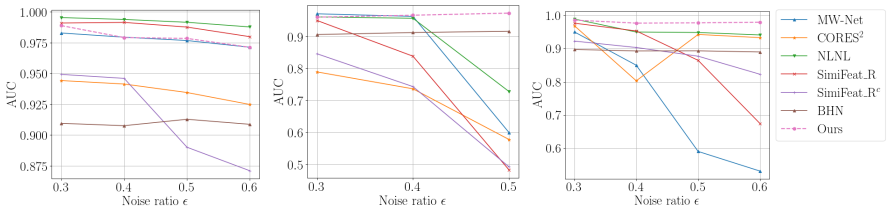
Noise Type	*Symmetric*				*Asymmetric*			*Instance*			
Rate ε	0.3	0.4	0.5	0.6	0.3	0.4	0.5	0.3	0.4	0.5	0.6
	FDR ↓										
CORES^2	**3.90**	**3.38**	**2.87**	**2.89**	58.62	56.67	51.16	**6.39**	**5.54**	**6.18**	6.15
NLNL	**11.12**	10.07	9.31	14.19	**22.69**	36.05	42.78	30.22	27.69	27.17	37.08
SimiFeat-R	38.98	28.94	22.64	17.66	47.12	46.64	49.22	41.70	35.56	32.66	32.39
SimiFeat-R^c	58.26	48.02	39.37	35.71	62.83	55.62	50.03	63.10	53.94	44.19	33.92
BHN	12.89	**8.75**	**4.10**	**4.38**	**9.97**	**5.66**	**5.70**	**5.79**	6.62	**6.00**	**3.76**
Ours	25.56	13.16	7.34	4.74	37.26	**16.10**	**7.91**	25.96	14.37	7.43	**3.95**
	Recall ↑										
CORES^2	16.27	22.22	30.11	36.66	0.09	0.07	0.09	9.95	13.75	18.43	21.85
NLNL	93.21	93.20	95.59	**97.94**	55.78	41.21	26.72	94.91	**93.78**	**96.32**	**95.11**
SimiFeat-R	**99.59**	**99.47**	**99.35**	**99.41**	**89.24**	**76.71**	61.35	**96.00**	91.58	84.72	73.94
SimiFeat-R^c	**98.06**	**97.76**	**97.83**	96.87	**87.16**	**82.00**	**74.95**	87.57	84.37	84.71	87.09
BHN	14.68	14.72	10.10	15.38	12.49	9.05	14.93	4.29	9.86	13.23	13.88
Ours	81.99	78.07	82.61	75.12	58.42	62.14	**65.11**	**96.34**	**96.77**	**97.15**	**97.24**
	F1 ↑										
CORES^2	27.83	36.14	45.97	53.22	0.17	0.14	0.18	17.98	24.00	30.81	35.44
NLNL	**90.99**	**91.54**	**93.08**	**91.47**	**64.80**	50.12	36.43	80.43	**81.65**	**82.94**	**75.73**
SimiFeat-R	75.67	**82.89**	86.98	**90.07**	**66.41**	**62.94**	55.56	**72.55**	75.65	75.03	70.63
SimiFeat-R^c	58.55	67.87	74.86	77.28	52.11	57.59	**59.97**	51.92	59.59	67.29	75.14
BHN	25.13	25.35	18.27	26.50	21.93	16.51	25.77	8.21	17.82	23.19	24.26
Ours	**78.04**	82.22	**87.34**	84.00	60.51	**71.40**	**76.28**	**83.73**	**90.86**	**94.80**	**96.64**

Evaluation Metrics: We first compare performance of all methods in term of Noisy Label Detection: (1) the False Discovery Rate (FDR) of detected corrupted examples; (2) the recall of detected corrupted examples (Recall); (3) the F1 of the detected corrupted examples; and (4) the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curves. For our method, we empirically select samples with $\alpha_i \geq 0.3$ as clean samples. While the first three metrics provide the performance at a specific threshold, AUC provides an aggregate measure of performance across all possible thresholds[1].

Additionally, we report Image Classification accuracy of the final classifier model. Specifically, we train a ResNet-34 model on the combination of the filtered training dataset D_f^n ($\alpha_i \geq 0.3$ for our method) [2] and the clean dataset D^c with simple augmentations, which only includes random cropping and horizontal flipping. We use the SGD optimizer with starting learning of 0.1 together with

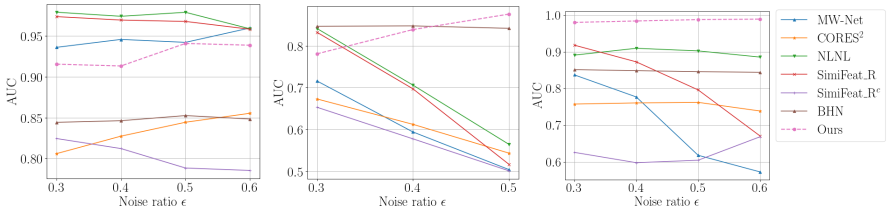
[1] We note that as MW-Net [16] aims to minimized the impact of noisy samples, and the authors do not provide a way to separate clean and noisy samples. Hence, we only report AUC for MW-Net.

[2] As authors of MW-Net [16] does not provide a way to separate clean and noisy samples, we assume that the noise rate is available to estimate the number of noisy samples (lowest weights) and eliminate them.



(a) Symmetric **(b)** Asymmetric **(c)** Instance

Fig. 1: Comparison in term of AUC using CIFAR-10 dataset for various synthetic noise types and ratios.



(a) Symmetric **(b)** Asymmetric **(c)** Instance

Fig. 2: Comparison in term of AUC using CIFAR-100 dataset for various synthetic noise types and ratios.

a Cosine Annealing learning rate scheduler to reduce learning rate to 10^{-8} after 100 epochs and the mini-batch size of 125.

4.2 Noisy Label Detection

We report the noisy label detection results (FDR, Recall and F1) of three types of synthetic noises at various noise levels for CIFAR-10 in Table 1 and for CIFAR-100 in Table 2. We report the AUC for CIFAR-10 in Figure 1 and CIFAR-100 in Figure 2. First, we can observe that SimiFeat-R [24] slightly outperforms our method and achieve the best noise label detection performance among comparing method for Symmetric noise for both CIFAR-10 and CIFAR-100 datasets. However, the performance of SimiFeat-R on asymmetric and instance noises is considerable lower, especially at higher noise levels. SimiFeat-R^c achieves significantly lower performance for both CIFAR-10 and CIFAR-100 Similar to SimiFeat-R, NLNL achieves good performance on Symmetric noise for CIFAR-10 dataset and noticable lower performance for asymmetric and instance noise. CORES^2 achieves high performance on symmetric and instance noise, but much lower performance on asymmetric noise. We found that BHN [22] requires a relative large number of clean training samples[3] to work properly. When the number of clean samples are limited (*e.g.*, 10% as in our experiment setting), BHN's performance degrades considerably. It is worth noting that, even though our AUC is lower than some baselines in symmetric noise scenarios, we achieves consistently

[3] They use 40% of CIFAR-10 and CIFAR-100 datasets as clean data.

Fig. 3: Examples of CIFAR-10 noisy samples with high sample weight value ($\alpha > 0.9$) (*i.e.* the samples can still be helpful for the model). The true-label/noisy-label of these images (left to right) are as follows: airplane/bird (the airplane is colorful with a green background, which is more commonly seen in birds than in airplanes), airplane/bird (the combination of an object with wings and a green background is common seen in birds), cat/dog (the body structures and fur texture of dog and cat are similar), horse/deer (the body structures of horse and deer are similar), and truck/airplane (the round shape of the truck makes it resemble to an airplane).

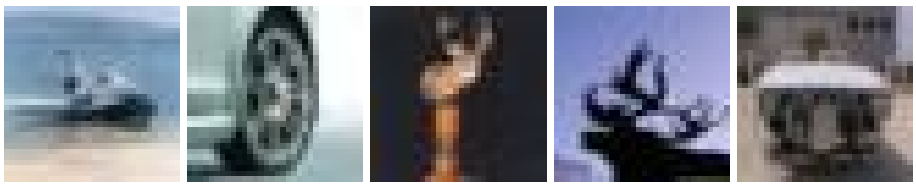


Fig. 4: Examples of CIFAR-10 clean samples with low sample weight value ($\alpha < 0.05$). From left to right: an image labeled as "airplane" portrays an airplane on water, an uncommon scenario; an image labeled as "car" displays only the car tire; an image labeled as "cat" is not easily recognizable as a cat; an image labeled "deer" displays only the silhouette; and an image labeled as "truck" presents the truck's rear rather than the front or sides as commonly seen in test images.

high AUC for all three types of synthetic noise at various noise levels. This is in contrast to other baseline methods (except BHN), which can degrade significantly at higher noise rates. This observation suggests that our method does not depend on the distributional assumptions on the noisy distribution.

4.3 Classification Performance on Clean Test Sets

Results on CIFAR-10 and CIFAR-100: We further verify whether our method improves the classification accuracy of the final classifier by selecting a better subset of training samples. In this experiment, we remove the detected corrupted samples and use the remaining samples to train the ResNet-34 classifier. We report the experimental results in Table 3 for CIFAR-10 and Table 4 for CIFAR-100. From Table 3, we can observe that our method can achieve very competitive performance compared to other baseline methods. Our method is slight under-perform SimiFeat-R on Symmetric noise at $\epsilon = 0.5$ and $\epsilon = 0.6$, while outperform SimiFeat-R by large margins on asymmetric and instance noise. It is worth noting that it is not totally fair as SimiFeat-R requires the model which

Table 3: Comparison of classification accuracy on CIFAR10 dataset with three types of synthetic noise.

Noise Type	*Symmetric*				*Asymmetric*			*Instance*			
Rate ε	0.3	0.4	0.5	0.6	0.3	0.4	0.5	0.3	0.4	0.5	0.6
MW-Net	92.36	90.64	88.65	84.26	92.30	90.26	79.25	88.72	80.68	67.34	49.55
CORES^2	90.30	88.49	85.99	83.22	90.60	89.10	85.47	87.17	86.63	81.38	80.64
NLNL	92.26	90.74	88.09	87.42	91.76	90.34	51.12	90.12	82.06	80.53	79.93
SimiFeat-R	92.55	90.65	**90.45**	**87.48**	90.37	83.99	48.48	90.79	87.99	76.08	53.11
SimiFeat-R^c	88.06	84.22	75.64	72.54	83.93	75.86	47.76	86.24	83.93	77.97	72.16
BHN	89.21	86.22	81.90	80.26	91.75	87.31	78.45	85.81	81.63	68.34	51.55
Ours	**92.68**	**91.14**	88.37	84.18	**92.45**	**91.52**	**90.62**	**92.65**	**91.50**	**90.83**	**89.10**

Table 4: Comparisons of classification accuracy on CIFAR100 dataset with three types of synthetic noise.

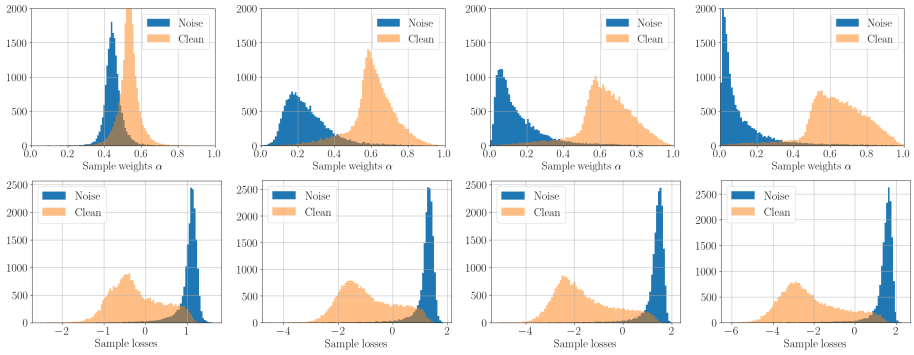
Noise Type	*Symmetric*				*Asymmetric.*			*Instance*			
Rate ε	0.3	0.4	0.5	0.6	0.3	0.4	0.5	0.3	0.4	0.5	0.6
MW-Net	69.69	66.62	65.47	**60.74**	61.32	59.32	35.10	64.91	58.70	40.49	31.37
CORES^2	55.70	49.30	42.15	34.80	56.15	49.90	40.30	59.23	50.14	46.72	38.86
NLNL	69.21	66.51	63.00	56.63	59.71	46.40	36.72	56.00	52.48	36.93	32.56
SimiFeat-R	68.21	65.18	61.01	56.61	63.96	48.41	29.03	65.21	60.61	50.05	31.99
SimiFeat-R^c	47.79	41.71	38.14	36.09	41.67	35.34	26.57	39.64	35.86	31.00	29.86
BHN	64.78	55.30	51.12	50.02	65.61	58.94	46.51	65.48	60.64	50.21	37.86
Ours	**70.81**	**68.73**	**66.53**	59.89	**69.45**	**67.98**	**64.31**	**74.58**	**72.00**	**71.35**	**67.47**

is pretrained in ImageNet dataset. Our method also achieves significantly improvement over other baselines, e.g., BHN. For CIFAR-100 dataset, our method consistently outperform all other baselines in all cases. We note that even though our method generally achieves slightly lower Recall and higher FDR for Symmetric noise compared to other baselines, our method still achieves the best performance in term of the classification accuracy of the final classifier model in most cases. These results demonstrate that some clean samples do not have much contribution and removing these samples does not affect the classification performance. On the contrary, some noisy samples can still be not totally harmful. Some CIFAR-10 examples of noisy samples with high weights and clean samples with low weights are presented in Figure 3 and 4 respectively. Furthermore, we note that our method outperforms MW-Net for many of synthetic noise types and noise rates. Especially for asymmetric and instance noise at higher noise rate $\epsilon \geq 0.5$, our method can achieve significant higher classification accuracy. These results confirms the advantages of our approach compared to MW-Net in learning the sample weights.

Table 5: Comparison of classification accuracy on Clothing1M dataset using ResNet-34 and ResNet-50.

Method	CE	PTD-R-V	MW-Net	CORES^2	SimiFeat-R	CAL	BHN	Ours
ResNet-34	70.28	71.67*		73.24*				**74.95**
ResNet-50	70.79		73.72*		73.41*	74.17*	75.50*	**75.91**

* The results are cited from the corresponding papers.



(a) Epoch 25 **(b)** Epoch 50 **(c)** Epoch 75 **(d)** Epoch 100

Fig. 5: The distribution of sample weights α (first row) and sample losses (bottom row) at different epochs along the process of training ResNet-34 on CIFAR-10 dataset with symmetric noise and $\epsilon = 0.4$.

Results on Clothing1M: Apart from presenting results on synthetic datasets, we also provide the classification accuracy for the real-world noisy label dataset Clothing1M in Table 5. Our method achieves the best accuracy and outperforms BHN for ResNet-50 models with the accuracy gains by 0.41% respectively. We note that our method can outperform MW-Net [16] by a large margin of 2.19%. This again indicates the advantages of our approach compared to MW-Net in learning the sample weights.

4.4 Ablation studies

Visualization of the sample weights α: Figure 5 illustrates the histograms of sample weights and sample loss distributions at different epochs during the training of ResNet-34 on CIFAR-10 with symmetric noise at $\epsilon = 0.4$. Notably, at the early training stage (Epoch 25), there is only a slight distinction between the weights of clean and noisy samples. However, as training progresses, the weights of noisy samples gradually decrease, while the weights of clean samples gradually increase. By Epoch 100, a significant portion of noisy samples exhibit very small weights ($\alpha \approx 0$), effectively minimizing their impact. This observation aligns with the distribution of sample loss values, where, over the course of training, the losses for clean samples decrease, indicating a better fit, while the losses for noisy samples mostly remain unchanged, indicating that the model doesn't attempt to fit those samples.

We also present a visualization of the sample weights learned by MW-Net [16] in Figure 6. It is noticeable that MW-Net can effectively assign small weights to the majority of noisy samples. However, MW-Net still inaccurately assigns high weight values for numerous corrupted samples. As a result, the filtered dataset still contains a lot of noisy samples with high weights. In contrast, our method (Figure 5d) ensures that noisy samples, even when selected, have small weights compared to clean samples. This observation clarifies why our method achieves superior performance compared to MW-Net in term of classification performance for almost every settings.

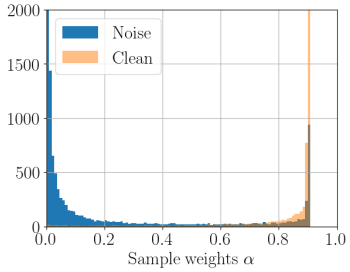


Fig. 6: Sample weights learned by MW-Net using ResNet-34 and CIFAR-10 dataset with symmetric noise at $\epsilon = 0.4$.

Effectiveness of the sample weights on classification model: We conduct an experiment to train classification models under different setting to understand the effectiveness of the sample weights in minimizing the impact of harmful samples. In this experiment, we only use the noisy dataset D^n to train the classifier: (1) Full D^n without sample weights (we use the D^c as validation dataset to apply early stop), (2) Full D^n with sample weights α when jointly train Θ and α, (3) Full D^n with sample weights α, (4) Filtered D^n (*i.e.* D^n_f) at a predefined threshold ($\alpha \geq 0.3$) *without* sample weights, (5) Filtered D^n (*i.e.* D^n_f at $\alpha \geq 0.3$) *with* sample weights, (6) clean-label samples of D^n only (this works as the upper bound). The experimental results are reported in Table 6. Comparing between Setting (1) and (2), we can see that the sample weights can help to minimize the impact of the noisy samples on model significantly and help to reduce the gap between a classifier trained on D^n with vanilla cross entropy loss and a classifier trained on only clean samples by 38.39%, 53.42%, and 67.82% for symmetric, asymmetric and instance noise respectively. Additionally, removing low weight samples ($\alpha_i \leq 0.3$) (*i.e.*, Setting (4) and (5)) will further improve classification accuracy and reduce performance gaps with the upper bounds.

Effects of clean dataset size: As our method relies on a small clean dataset, we conducted experiments to evaluate how the size of the clean dataset D^c impacts overall performance. In this investigation, we utilized the CIFAR-10 and CIFAR-100 dataset, with 80% of the training dataset (40,000 samples) designated as the noisy set. The remaining 20%, comprising 10,000 samples, served as the pool from which we randomly sampled the clean dataset at varying sizes. The experimental results are presented in Table 7. We can observe that for CIFAR-10 dataset, there is only small drops in AUC and accuracy when the clean dataset size as small as 10 samples per class. For CIFAR-100, we observe larger drops in AUC and accuracy as $|D^c|$ getting smaller. This is understandable, as the model requires more data points to effectively distinguish between

Table 6: Comparison of classification accuracy on CIFAR-10 with three types of synthetic noise at $\epsilon = 0.4$ using ResNet-34 for various settings to analyse the effectiveness of the sample weight α.

Noise Type	Symmetric	Asymmetric	Instance
(1) - D^n *without* α (baseline)	83.01	84.68	79.63
(2) - D^n with α (joint training)	86.98	89.02	89.07
(3) - D^n with α (retraining)	87.21	89.29	89.27
(4) - D_f^n *without* α	90.05	90.28	90.60
(5) - D_f^n with α	90.22	90.72	91.30
(6) - Upper bound	93.95	93.31	93.55

Table 7: Analyse the noise detection and classification performance at various clean dataset $|D^c|$ sizes per class.

	CIFAR-10				CIFAR-100		
$\|D^c\|$/class	10	20	50	100	10	20	50
AUC	95.56	95.98	96.09	96.65	89.40	90.65	92.59
Accuracy	87.86	89.74	89.83	89.94	61.10	63.32	65.06

a greater number of classes. Furthermore, our approach involves learning sample weights to minimize the loss function on the clean data. As the size of the clean dataset shrinks, it might become less representative of the actual distribution present in the test set. This mismatch between clean training data and testing data contributes to the performance decline observed with smaller clean datasets, particularly for CIFAR-100.

5 Conclusion

In conclusion, this paper has introduced a simple yet novel approach to learn sample weights by explicitly learning a scalar weight value for each training sample. This sample weight dynamically accumulates the positive or negative impacts contributed by the corresponding sample to the model, effectively enhancing the influence of helpful samples and minimizing the impact of harmful ones. The experimental results demonstrate the efficacy of our method, showcasing favorable performance in both noisy label detection and classification metrics when compared to recent methods across various synthetic noise types and levels as well as the real-world noisy dataset Clothing1M. Moreover, our approach to learning sample weights outperforms the method proposed in MW-Net [16], highlighting its potential for achieving superior performance in various applications.

Acknowledgement

This research was partially supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP210102798). The views expressed herein are those of the authors and are not necessarily those of the Australian Government or Australian Research Council.

References

1. Bahri, D., Jiang, H., Gupta, M.: Deep k-NN for noisy labels. In: ICML. vol. 119, pp. 540–550 (13–18 Jul 2020)
2. Bai, Y., Yang, E., Han, B., Yang, Y., Li, J., Mao, Y., Niu, G., Liu, T.: Understanding and improving early stopping for learning with noisy labels. In: NeurIPS (2021)
3. Cheng, H., Zhu, Z., Li, X., Gong, Y., Sun, X., Liu, Y.: Learning with instance-dependent label noise: A sample sieve approach. In: ICLR (2021)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: CVPR (2016)
6. Hendrycks, D., Mazeika, M., Wilson, D., Gimpel, K.: Using trusted data to train deep networks on labels corrupted by severe noise. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) NIPS. vol. 31 (2018)
7. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: ICML. pp. 2304–2313 (10–15 Jul 2018)
8. Kim, Y., Yim, J., Yun, J., Kim., J.: Nlnl: Negative learning for noisy labels. In: ICCV (2019)
9. Kong, S., Li, Y., Wang, J., Rezaei, A., Zhou, H.: Knn-enhanced deep learning against noisy labels. CoRR **abs/2012.04224** (2020)
10. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
11. Li, J., Socher, R., Hoi, S.C.: Dividemix: Learning with noisy labels as semi-supervised learning. In: ICLR (2020)
12. Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., Li, L.J.: Learning from noisy labels with distillation. In: ICCV. pp. 1928–1936 (2017)
13. Patrini, G., Rozza, A., Menon, A.K., Nock, R., Qu, L.: Making deep neural networks robust to label noise: A loss correction approach. In: CVPR. pp. 2233–2241 (2017)
14. Pfister, T., Zhang, Z.: Learning fast sample re-weighting without reward data. In: International Conference on Machine Learning (2021)
15. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. In: ICML (2018)
16. Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., Meng, D.: Meta-weight-net: learning an explicit mapping for sample weighting (2019)
17. Vahdat, A.: Toward robustness against label noise in training deep discriminative neural networks. In: NIPS. p. 5601–5610 (2017)
18. Xia, X., Liu, T., Han, B., Wang, N., Gong, M., Liu, H., Niu, G., Tao, D., Sugiyama, M.: Part-dependent label noise: Towards instance-dependent label noise. In: NeurIPS (2020)
19. Xia, X., Liu, T., Wang, N., Han, B., Gong, C., Niu, G., Sugiyama, M.: Are anchor points really indispensable in label-noise learning? In: NeurIPS. vol. 32 (2019)
20. Xiao, T., Xia, T., Yang, Y., Huang, C., Wang, X.: Learning from massive noisy labeled data for image classification. In: CVPR (2015)
21. Yao, Y., Liu, T., Han, B., Gong, M., Deng, J., Niu, G., Sugiyama, M.: Dual t: reducing estimation error for transition matrix in label-noise learning. In: NeurIPS (2020)

22. Yu, C., Ma, X., Liu, W.: Delving into noisy label detection with clean data. In: ICML (2023)
23. Zhang, Z., Zhang, H., Arik, S.O., Lee, H., Pfister, T.: Distilling effective supervision from severe label noise. In: CVPR. pp. 9291–9300 (2020)
24. Zhu, Z., Dong, Z., , Liu, Y.: Detecting corrupted labels without training a model to predict. In: ICML (2022)
25. Zhu, Z., Song, Y., Liu, Y.: Clusterability as an alternative to anchor points when learning with noisy labels. In: ICML. pp. 12912–12923 (18–24 Jul 2021)