Noise-resistant Deep Metric Learning with Ranking-based Instance Selection

Chang Liu[1], **Han Yu**[1], **Boyang Li**[1*], **Zhiqi Shen**[1*], **Zhanning Gao**[2*], **Peiran Ren**[2], **Xuansong Xie**[2], **Lizhen Cui**[3,4], **Chunyan Miao**[1*]

[1]School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore
[2]Alibaba Group, Hangzhou, China [3]School of Software, Shandong University (SDU), Jinan, China
[4]Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR), SDU, Jinan, China
*{boyang.li, zqshen}@ntu.edu.sg, zhanning.gzn@alibaba-inc.com, ascymiao@ntu.edu.sg

Abstract

The existence of noisy labels in real-world data negatively impacts the performance of deep learning models. Although much research effort has been devoted to improving robustness to noisy labels in classification tasks, the problem of noisy labels in deep metric learning (DML) remains open. In this paper, we propose a noise-resistant training technique for DML, which we name Probabilistic Ranking-based Instance Selection with Memory (PRISM). PRISM identifies noisy data in a minibatch using average similarity against image features extracted by several previous versions of the neural network. These features are stored in and retrieved from a memory bank. To alleviate the high computational cost brought by the memory bank, we introduce an acceleration method that replaces individual data points with the class centers. In extensive comparisons with 12 existing approaches under both synthetic and real-world label noise, PRISM demonstrates superior performance of up to 6.06% in Precision@1.

1. Introduction

Commonly resulting from human annotation errors or imperfect automated data collection, noisy labels in training data degrade the predictive performance of models trained on them [11, 47, 16]. Manual inspection and correction of labels are labour-intensive and hence scale poorly to large datasets. Therefore, training techniques that are robust to incorrect labels in training data play an important role in real-world applications of machine learning.

To date, most works on noise-resistant neural networks [11, 17, 31, 52, 47, 48, 16] focus on image classification. Little research effort has been devoted to noise-resistant deep metric learning (DML). The goal of DML is to learn a distance metric that maps similar pairs of data points close together and dissimilar pairs far apart, based on a predefined notion for similarity. DML finds diverse applications such as image retrieval [18, 10, 33], landmark identification [49], and self-supervised learning [25].

Pair-based loss functions encourages DML networks to distinguish a similar pair of data points from one or more dissimilar pairs. Large batch sizes often lead to improved performance [6, 46, 5], as larger batches are more likely to contain informative examples. Pushing the idea of large batches to an extreme, [46] collects all positive and negative data samples from a memory bank. However, in the presence of substantial noise, indiscriminate use of all samples could lower performance. Alternatively, [26] uses learnable class centers to replace individual data samples in order to reduce computational complexity. Nonetheless, the cluster centers can also be sensitive to outliers and label noise.

We propose a noise-resistant deep metric learning algorithm, *Probabilistic Ranking-based Instance Selection with Memory* (PRISM), which works with both the memory bank approach and the class-center approach. PRISM computes the probability that a label is clean based on the similarities between the data point and other data points using features extracted during the last several training iterations. This may be seen as modeling the posterior probability of the data label. For data points with high probability, we extract their features and insert them into the memory bank, which is used in subsequent model updates. In addition, we develop a smooth top-R (sTRM) trick to adjust the threshold for noisy data identification as well as an acceleration technique that replaces individual data points with the class centers in the probability calculation.

We perform extensive empirical evaluations on both synthetic and real datasets. Inspired by the the "noise cluster" phenomenon observed from real-world data, we introduce the Small Cluster noise model to mimic open-set noise in real data. Experimental results show that PRISM achieves superior performance compared to 12 existing DML and noise-resistant training techniques under symmetric noise, Small Cluster noise, and real noise. In addition, the acceleration trick speeds up the algorithm by a factor of 6.9 on SOP dataset. The code and data are available at https://github.com/alibaba-edu/Ranking-based-Instance-Selection.

2. Related Work

Noise-resistant Training in Classification. Training under noisy labels has been studied extensively for classification [2, 47, 48, 16, 54, 1]. A common approach is to gradually detect noisy labels and exclude them from the training set. F-correction [31] models the noise as a class transition matrix. MentorNet [17] trains a teacher network that provides weight for each sample to the student network. Co-teaching [11] trains two networks concurrently. Samples identified as having small loss by one network is used to train the other network. Co-teaching+ [52] trains on samples that have small losses and different predictions from the two networks.

Differing from the conventional classification problem, metric learning learns an effective distance metric that works well for unseen classes and is evaluated using retrieval-based criteria. In the experiments, PRISM demonstrates superior performance to several noise-resistant classification baselines.

Models of Label Noise. Though noisy labels are prevalent, it is often difficult to ascertain and control the degree of noise in natural datasets. Artificial noisy models are thus commonly used as evaluation metrics for noise-resistant algorithms. In the symmetric noise model [40], a proportion R of data points belonging to one ground-truth class are uniformly distributed to all other classes. In pairwise noise (e.g., [11]), data points from each class are transferred to a designated target class. [16] curates a natural dataset with controlled levels of noise.

Open-set noise refers to the presence of data points that do not belong to any classes recognized by the dataset. Under open-set noise, it is futile to model noise as a class transition matrix, which adds to the difficulty. In classification, [47] simulates open-set noisy labels by adding data from other datasets. In this paper, we propose the Small Cluster noise model for open-set label noise in metric learning.

Deep Metric Learning. We broadly categorize deep metric learning into 1) pair-based methods and 2) proxy-based methods. Pair-based methods [34, 44, 43, 37] calculate loss based on the contrast between positive pairs and negative pairs, which is often calculated using contrastive loss [7], triplet losses [14], or softmax loss [9]. In this process, identifying informative positive and negative pairs becomes an important consideration [35, 12, 36, 6, 43, 37, 50]. Proxy-based methods [26, 32, 6, 19, 38, 55, 8, 53, 4] represent each class as one or more proxy vectors, and use the similarities between the input data and the proxies to calculate the loss. Proxies are learned from data during model training, which could deviate from the class center under heavy noise and cause performance degradation.

Noisy Labels in Metric Learning: To our knowledge, the only method which explicitly handles noise in neural metric learning is [42]. The technique estimates the posterior label distribution using a shallow Bayesian neural network with only one layer. Due to its computational complexity, the approach may not scale well to deeper network architectures.

A few works attempt to handle outliers in normal training data in DML, but do not explicitly deal with substantial label noise. Wang *et al.* [45] uses the pair-based loss and trains a proxy for each class simultaneously to adjust the weights of the outliers, but substantial label noise may cause the learned proxies to be inaccurate. Ozaki *et al.* [30] handles noisy data in DML by first performing label cleaning using a model trained on a clean dataset, which may not be available in real-world applications. In this paper, we do not rely on the existence of a clean dataset.

3. The PRISM Approach

The detection of noisy data labels usually attempts to catch data points that stand out from others in the same class. However, distinguishing such data samples in DML requires a good similarity metric. The learning of a good similarity metric in turn depends on the availability of clean training data, thereby creating a chicken-and-egg problem.

To cope with this challenge, PRISM adopts an online data filtering approach. At every training iteration, we use the features extracted during the past several iterations to filter out a portion of training data. The rest are considered clean, added to the memory bank, and used in updating the metric.

3.1. Identifying Noisy Labels

Let the training set be $\mathcal{X} = \{(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)\}$, where x_i is an image and y_i is the corresponding label. N is the size of training set. The aim of DML is to learn a convolutional neural network, $f(\cdot)$, which extracts a feature vector $f(x_i)$ for image x_i, such that the cosine similarity $S(f(x_i), f(x_j))$ between $f(x_i)$ and $f(x_j)$ is high if $y_i = y_j$ and low if $y_i \neq y_j$:

$$S(f(x_i), f(x_j)) = \frac{f(x_i)^T f(x_j)}{\|f(x_i)\| \|f(x_j)\|}. \tag{1}$$

In addition, we denote the current mini-batch as $\mathcal{B} = \{(x_0, y_0), (x_1, y_1), ..., (x_B, y_B)\}$ where B is the batch size. A pair of feature $(f(x_i), f(x_j))$ is called positive pair if $y_i = y_j$, negative pair if $y_i \neq y_j$.

To identify noisy labels, we maintain a first-in first-out memory bank $\mathcal{M}$, $\mathcal{M} = \{(v_0, y_0), (v_1, y_1), ..., (v_M, y_M)\}$, to store historic features of data samples. M is the size of memory bank. In every step of stochastic gradient descent, we separate the clean data from the noisy data, and append the current feature v_i of clean data x_i to the memory bank. If the maximum bank capacity is exceeded, the oldest features are dequeued from the memory bank, so that we always keep track of the more recent features.

We compare the features of x_i with the content of the memory bank to determine if its label y_i is noisy. If y_i is a clean label, then the similarity between x_i and other samples with the same class label in the memory should be large compared to its similarity with samples from other classes. Based on this intuition, we define the probability of (x_i, y_i) being a clean data point, $P_{\text{clean}}(i)$, as follows

$$P_{\text{clean}}(i) = \frac{\exp\left(T(x_i, y_i)\right)}{\sum_{k \in C} \exp\left(T(x_i, k)\right)} \tag{2}$$

$$T(x_i, k) = \frac{1}{M_k} \sum_{(v_j, y_j) \in \mathcal{M}, y_j = k} S(f(x_i), v_j) \tag{3}$$

M_k is the number of samples in class k in the memory bank. $T(x_i, k)$ is the average similarity between x_i and all the stored features v_j in class k. $T(x_i, k)$ may be seen as $\log P(X = x_i | Y = k)$ up to a constant. Equation 2 can be understood as the probability $P(Y = k | X = x_i)$, assuming a uniform prior for $P(Y = k)$ and identical normalization constants for every class. Although similar math forms can be found in applications such as metric learning [9, 32], data visualization [39], and uncertainty estimation [24, 51], we note that its use in noise-resistant DML is novel.

When $P_{\text{clean}}(i)$ falls below a threshold m, we treat (x_i, y_i) as a noisy data sample. We propose two methods to determine the value of threshold m: the top-R method (TRM) and the smooth top-R method (sTRM). Under TRM, we define a filtering rate (*i.e.*, estimated noise rate) R. In each minibatch, we treat (x_i, y_i) as noisy if $P_{\text{clean}}(i)$ falls in the smallest $R\%$ of all samples in the current minibatch $\mathcal{B}$.

In contrast, sTRM keeps track of the average of the R^{th} percentile of $P_{\text{clean}}(i)$ values over the last τ batches. Formally, let Q_j be the R^{th} percentile $P_{\text{clean}}(i)$ value in j-th mini-batch, the threshold m is defined as:

$$m = \frac{1}{\tau} \sum_{j=t-\tau}^{t} Q_j \tag{4}$$

Compared to TRM, the sliding window approach of sTRM reduces the influence of a single mini-batch and creates a smooth and more accurate estimate of the R^{th} percentile.

To create balanced minibatches, we first sample P unique classes and sample K images for each selected class, yielding PK images in every minibatch.

3.2. Accelerating PRISM

The above method requires computing the similarities between all pairs of data samples, which has high time complexity. We propose a simple technique for improving the efficiency. For the k^{th} cluster, we replace its M_k data samples with the mean feature vector w_k of the class,

$$\begin{aligned}\sum_{\substack{(v_j, y_j) \in \mathcal{M} \\ y_j = k}} \frac{S(f(x_i), v_j)}{M_k} &= \left(\frac{1}{M_k} \sum_{\substack{(v_j, y_j) \in \mathcal{M} \\ y_j = k}} \frac{v_j}{\|v_j\|} \right) \frac{f(x_i)}{\|f(x_i)\|} \\ &= w_k \frac{f(x_i)}{\|f(x_i)\|},\end{aligned} \tag{5}$$

$$w_k = \frac{1}{M_k} \sum_{(v_j, y_j) \in \mathcal{M}, y_j = k} \frac{v_j}{\|v_j\|}. \tag{6}$$

Plugging Eq. (5) into Eq. (2), $P_{\text{clean}}(i)$ can be expressed as:

$$P_{\text{clean}}(i) = \exp\left(w_{y_i} \frac{f(x_i)}{\|f(x_i)\|} \right) / \sum_{k \in C} \exp\left(w_k \frac{f(x_i)}{\|f(x_i)\|} \right). \tag{7}$$

The time complexity of Eq. (2) is $O(PKN)$ for a mini-batch of size PK and a training set of N samples. Adopting Eq. (7) reduces the time complexity to $O(PK|C|)$, where $|C|$ is the total number of classes which is much smaller than N. This yields an acceleration factor of $\frac{N}{|C|}$.

3.3. The PRISM Algorithm

We now describe the PRISM algorithm, shown as Algorithm 1. In each iteration of training, PRISM first calculates $P_{\text{clean}}(i)$ for each (x_i, y_i) tuple. In the first iteration, when the class center vector w_{y_i} has not been updated and is the zero vector, all data samples in class y_i are considered clean (Line 3). At this moment, the memory bank does not contain any data points of class k, so we cannot compute the mean vector w_k. After the first iteration, we will update the center vector to a non-zero value and perform sample selection based on $P_{\text{clean}}(i)$.

After that, we compute the threshold m and add data points with $P_{\text{clean}}(i) > m$ to $\mathcal{B}_{clean}$ (Line 12). To reduce computational cost, we update only the center vectors of classes in $\mathcal{B}_{clean}$ (Line 17). Finally, the loss is calculated to update parameters of model $f(\cdot)$ (Line 19). PRISM computes the loss with clean labels and perform stochastic gradient descent. The loss functions are described in the following section.

3.4. Loss Functions

The traditional pair-based contrastive loss function [7] computes similarities between all pairs of data samples within the mini-batch $\mathcal{B}$. The loss function encourages $f(\cdot)$ to assign small distances between samples in the same class and large distances between samples from different classes.

Algorithm 1: A training iteration of PRISM

Input : $\mathcal{B} = \{(x_0, y_0), (x_1, y_1), ..., (x_B, y_B)\}$: a given minibatch of data with size B; $f(\cdot)$: a given deep metric model;

Parameter: $\{w_k | k \in C\}$: the set of mean feature vectors of all classes, all initialized to zero before training commences;

```
for each (x_i, y_i) ∈ B do
    if w_{y_i} = 0⃗ then
        P_clean(i) = 1;
    else
        Calculate P_clean(i) according to Eq. (7);
    end
end
Calculate the threshold m using TRM or sTRM;
Initialize B_clean as an empty set;
for each (x_i, y_i) ∈ B do
    if P_clean(i) > m then
        Add (f(x_i), y_i) to B_clean;
    end
end
Enqueue B_clean into the Memory Bank
for each (v_i, y_i) ∈ B_clean do
    Update w_{y_i} according to Eq. (6);
end
Calculate loss L(B_clean) and update the parameters
  of f(·)
```

More formally, the loss for mini-batch $\mathcal{B}$ is

$$\begin{aligned} L_{\text{batch}}(\mathcal{B}) = & \sum_{i,j \in B, y_i \neq y_j} \max(S(f(x_i), f(x_j)) - \lambda, 0) \\ & - \sum_{i,j \in B, y_i = y_j} S(f(x_i), f(x_j)) \end{aligned} \tag{8}$$

where $\lambda \in [0, 1]$ is a hyperparameter for the margin. With a memory bank $\mathcal{M}$ that stores the features of data samples in previous minibatches in a first-in-first-out manner [46], we can employ many more positive and negative pairs in the loss, which may reduce the variance in the gradient estimates. The memory bank loss can be written as:

$$\begin{aligned} L_{\text{bank}}(\mathcal{M}, \mathcal{B}) = & \sum_{\substack{(x_i, y_i) \in \mathcal{B}, (v_j, y_j) \in \mathcal{M} \\ y_i \neq y_j}} \max(S(f(x_i), v_j) - \lambda, 0) \\ & - \sum_{\substack{(x_i, y_i) \in \mathcal{B}, (v_j, y_j) \in \mathcal{M} \\ y_i = y_j}} S(f(x_i), v_j). \end{aligned} \tag{9}$$

The total loss is the sum of the batch loss $L_{\text{batch}}(\mathcal{B})$ and the memory bank loss $L_{\text{bank}}(\mathcal{M}, \mathcal{B})$, referred to as the *memory-based contrastive loss* [46]. As we adopt the memory bank setup to identify data with noisy labels, PRISM works well under the memory-based contrastive loss.

Another loss we employ with PRISM is the Soft Triple loss [32], a type of proxy-based loss function. This loss maintains H learnable proxies per class. A proxy is a vector that has the same size with the feature of an image. The similarity between an image to a given class of images is represented as a weighted similarity to each proxy in the class. The loss is computed as the similarities between the minibatch data and all classes:

$$L_{\text{SoftTriple}} = -\log \frac{\exp(\lambda(S'_{i,y_i} - \delta))}{\exp(\lambda(S'_{i,y_i} - \delta)) + \exp(\lambda S'_{i,j})}, \tag{10}$$

$$S'_{i,j} = \frac{\sum_{h=1}^{H} \exp\left(\gamma f(x_i)^\top p_j^h\right) f(x_i)^\top p_j^h}{\sum_{h=1}^{H} \exp\left(\gamma f(x_i)^\top p_j^h\right)}. \tag{11}$$

λ and γ are predefined scaling factors. δ is a predefined margin. p_j^h is the h-th proxy for class j, which is a learnable vector updated during model training.

4. Experimental Evaluation

4.1. Datasets

We compare the algorithms on five datasets, including:

- **CARS** [20], which contains 16,185 images of 196 different car models. Following [20], we use the first 98 models for training and the rest for testing, and incorporate synthetic label noise into the training set.

- **CUB** [41], which contains 11,788 images of 200 different bird species. Following [41], we use the first 100 species for training and the rest for testing, and incorporate synthetic label noise into the training set.

- **Stanford Online Products (SOP)** [29], which contains 59,551 images of 11,318 furniture items on eBay. We use 59,551 images in all classes for training and the rest for testing, and incorporate synthetic label noise into the training set.

- **Food-101N** [21], which contains 310,009 images of food recipes in 101 classes. The test set is the Food-101 [3] dataset, which contains the same 101 classes as Food-101N. Images in Food-101N are obtained using search results from Google, Bing, Yelp, and TripAdvisor with an estimated noise rate of 20% [21]. In the same evaluation setup with CARS, CUB, and SOP, we use 144,086 images in the first 50 classes (in alphabetical order) as the training set, and the remaining 51 classes in Food-101 as the test set which contains 51,000 images.

Figure 1: Example images in CARS-98N. Images in the first row have clean labels. The second row shows some images with noisy labels, including car interiors and car parts.

- **CARS-98N**. We build a new noisy label dataset named CARS-98N by crawling 9,558 images for 98 car models from Pinterest. We used the Pinterest image search engine to retrieve images using the 98 labels from the CARS training set as the query terms. The CARS-98N is *only used for training*, and the test set of CARS is used for performance evaluation. Figure 1 shows example images in this dataset. The noisy images often contain the interior of the car, car parts, or images of other car models.

4.2. Synthesizing Label Noise

In the experiments, we adopt two models for noisy label synthesis: 1) symmetric noise and 2) Small Cluster noise. Symmetric noise [40] has been widely used to evaluate the robustness of classification models (e.g., [11, 52, 31, 22]). Given a clean dataset, the symmetric noise model assigns a predefined portion of data from every ground-truth class to all other classes with equal probability, without regard to the similarity between data samples. After the noise synthesis, the number of classes remains unchanged.

We contend that noisy labels that occur naturally differ from the symmetric noise model. Observing Food-101N and CARS-98N, the two datasets with naturally occurring noisy labels, we notice that some noisy data points are close to each other and can form small clusters. This is evident in Figure 1, where the car interior and car part images can form their own clusters. Further, the number of classes in metric learning may not be fixed. For example, in person or vehicle reidentification, two people or vehicles with similar looks may be inadvertently merged into one cluster. Conversely, images of the same person with different outfits may be separated into different clusters.

To mimic these traits of naturally occurring label noise, we propose a new noise synthesis model — Small Cluster. In this method, we first cluster images from a randomly selected ground-truth class into a large number of small clusters, using features extracted from a pretrained neural network. Here we use L2-normalized features from ResNet-18 pretrained on ImageNet. The number of clusters is set to one half of the number of images in the class. Each cluster is then merged into a randomly selected ground-truth class. After every iteration, the number of classes decreases by one. In this way, the Small Cluster model creates an open-set label noise scenario [47] as the ground-truth classes are eliminated in the corrupted dataset.

4.3. Baseline Techniques

We compare PRISM against 12 baselines, including four baselines designed for noise-resistant classification and eight baselines for deep metric learning:

- Co-teaching [11], which trains two CNNs jointly. Data samples assigned small losses by one model are used to train the other model.
- Co-teaching+ [52]. Similar to Co-teaching, a model is trained using data samples that (1) the two models disagree on, and (2) receive small losses from the other model.
- Co-teaching [11] with Temperature [53], which adds a temperature hyperparameter to the cross-entropy loss.
- F-correction [31], which multiplies the class transition matrix to the loss function. Since Small Cluster produces open-set label noises, for which the class transition matrix is not properly defined, this baseline is only used under symmetric noise settings.
- Eight DML algorithms. Four of them use proxy-based losses (including Soft Triple [32], FastAp [6], nSoftmax [53] and proxyNCA [26]), and the other four use pair-based losses (including MS loss [43], circle loss [37], contrastive loss [7] and memory contrastive loss (MCL) [46]). They treat all data samples as clean.

We train the classification baselines using cross-entropy. When retrieving images during inference, we use the L2-normalized features from the layer before the final linear classifier.

For Co-teaching, Co-teaching+ and Memory Contrastive Loss [46], we use the official implementation. The other DML algorithms are implemented by Pytorch Metric Learning [28]. We follow the hyperparameter settings given in the respective papers or code repositories. For Co-teaching and Co-teaching+, we use the learning rate (LR) scheduler given in their code, while for others (including our algorithm), cosine LR decay [23] is used. We set the batch size to 64 for experiments on all datasets and all models. The input images are first resized to 256x256, then randomly cropped to 224x224. A horizontal flip is performed on the training data with a possibility of 0.5.

Table 1: Precision@1 (%) on CARS, SOP, and CUB dataset with symmetric label noise.

	CARS			SOP			CUB		
Noisy Label Rate	10%	20%	50%	10%	20%	50%	10%	20%	50%
Algorithms for image classification under label noise									
Co-teaching [11]	73.47	70.39	59.55	62.60	60.26	52.18	53.74	51.12	45.01
Co-teaching+ [52]	71.49	69.62	62.35	63.44	67.93	58.29	53.31	51.04	45.16
Co-teaching [11] w/ Temperature [53]	77.51	76.30	66.87	73.71	71.97	64.07	55.25	54.18	50.65
F-correction [31]	71.00	69.47	59.54	51.18	46.34	48.92	53.41	52.60	48.84
DML with Proxy-based Losses									
FastAP [6]	66.74	66.39	58.87	69.20	67.94	65.83	54.10	53.70	51.18
nSoftmax [53]	72.72	70.10	54.80	70.10	68.90	57.32	51.99	49.66	42.81
ProxyNCA [26]	69.79	70.31	61.75	71.10	69.50	61.49	47.13	46.64	41.63
Soft Triple [32]	76.18	71.82	52.53	68.60	55.21	38.45	51.94	49.14	41.46
DML with Pair-based Losses									
MS [43]	66.31	67.14	38.24	69.90	67.60	59.58	57.44	54.52	40.70
Circle [37]	71.00	56.24	15.24	72.80	70.50	41.17	47.48	45.32	12.98
Contrastive Loss [7]	72.34	70.93	22.91	68.70	68.80	61.16	51.77	51.50	38.59
Memory Contrastive Loss (MCL) [46]	74.22	69.17	46.88	79.00	76.60	67.21	56.72	50.74	31.18
MCL + PRISM (Ours)	**80.06**	**78.03**	**72.93**	**80.11**	**79.47**	**72.85**	**58.78**	**58.73**	**56.03**

For Soft Triple and Soft Triple with PRISM, we set the number of proxies per class $H = 10$ for CARS-98N and Food-101N. Other hyperparameters follow [32]. The size of the memory bank is set to the size of the training dataset as in [46].

Again following [46], when comparing performance on CARS, CUB and CARS-98N, we use BN-inception [15] as the backbone CNN model for all algorithms. The dimension of the output feature is set as 512, the same as in [46]. No other tricks (e.g., freezing BN layers) are used during the experiments. For SOP and Food-101N, we use ResNet-50 [13] with a 128-dimensional output.

Testing is based on the ranked list of the nearest neighbors for the test images. Specifically, we use Precision@1 (P@1) and Mean Average Precision@R (MAP@R) [27] as the evaluation metrics. The test sets are noise-free, as the purpose of the experiments is to evaluate the algorithms in the presence of noisy labels in the training data.

Table 2: Precision@1 (%) on CARS, SOP, and CUB with Small Cluster label noise.

	CARS		SOP		CUB	
Noisy Label Rate	25%	50%	25%	50%	25%	50%
Algorithms for image classification under label noise						
Co-teaching	70.57	62.91	61.97	58.08	51.75	48.85
Co-teaching+	70.05	61.58	62.57	59.27	51.55	47.62
Co-teaching w/ Temperature	75.26	66.19	70.19	68.50	54.59	48.32
DML with Proxy-based Losses						
FastAP	62.49	53.07	70.66	67.55	52.18	48.46
nSoftmax	71.61	62.29	70.00	61.92	49.61	41.78
ProxyNCA	69.50	58.34	67.95	62.25	42.07	36.48
Soft Triple	73.26	66.66	**73.63**	64.14	56.18	50.35
Soft Triple + PRISM (Ours)	**77.60**	**70.45**	70.99	**69.38**	**57.61**	**54.27**
DML with Pair-based Losses						
MS	63.92	43.73	67.32	62.17	53.60	41.66
Circle	53.03	19.95	70.33	40.48	44.07	22.96
Contrastive Loss	65.60	26.45	68.25	64.27	47.27	39.43
Memory Contrastive Loss (MCL)	69.46	36.43	75.61	68.71	52.25	41.58
MCL + PRISM (Ours)	**77.08**	**68.26**	**78.56**	**73.84**	**55.77**	**53.46**

4.4. Results and Discussions

Symmetric Label Noise. Table 1 shows the evaluation results on CARS, SOP, and CUB under symmetric label noise. PRISM with Memory Contrastive Loss achieves the highest performance among all the compared algorithms. As the noisy label rate increases, the Precision@1 scores decrease for all approaches, but PRISM exhibits the least performance drop among all methods. In CUB, the performance of PRISM decreases by 0.05% when the noise level increases from 10% to 20%, and by less than 3% when noise increases to 50%.

The robust classification methods, including Co-teaching, Co-teaching+, and F-correction, show a certain level of robustness to noisy labels. However, their performance is generally lower compared to DML approaches and PRISM. Applying temperature normalization to cross-entropy [53] boosts the performance of Co-teaching, especially on the SOP dataset.

Approaches with proxy-based loss achieve generally high scores under CUB and CARS, even at 50% noise. However, they generally perform worse than approaches with pair-based loss under SOP. The average number of images per class in the SOP training set is only 5.26, in contrast to 82.18 in CARS, causing difficulties in accurately estimating the proxy centers in SOP.

In comparison, pair-based losses are subject to severe performance drops under high noise rates. The pair-based loss takes a pair of samples (x_i, x_j) as unit. At 50% noise, the correct pairs only account for 25% of all pairs, which bears on the performance heavily. In addition, both MS loss

Table 3: Precision@1 (%) and Mean Average Precision@R (%) on CARS-98N and Food-101N [21].

	CARS-98N		Food-101N	
	P@1	MAP@R	P@1	MAP@R
Algorithms for image classification under label noise				
Co-teaching	58.74	9.10	59.08	14.66
Co-teaching+	56.66	8.40	57.59	14.72
Co-teaching w/ Temperature	60.72	9.61	63.18	17.38
DML with Proxy-based Losses				
ProxyNCA	53.55	8.75	48.41	9.30
Soft Triple	63.36	10.88	63.61	16.23
Soft Triple + PRISM (Ours)	**64.81**	**11.21**	**64.46**	**17.53**
DML with Pair-based Losses				
MS	49.00	5.92	52.53	9.82
Contrastive	44.91	4.76	50.04	9.42
Memory Contrastive Loss (MCL)	38.73	3.34	**52.58**	**9.88**
MCL + PRISM (Ours)	**57.95**	**8.04**	52.47	9.64

[43] and circle loss [37] assign higher weights for learning the hard examples which could be the noisy data. This causes the performance to drop significantly. The use of the memory bank in MCL increases the performance. However, the performance is still lower than that of proxy-based loss approaches.

Small Cluster Open-set Label Noise. Table 2 reports the Precision@1 scores achieved by various approaches on datasets with Small Cluster label noise. Incorporating PRISM into MCL improves the performance of the resulting model. The advantage of PRISM is especially pronounced under 50% noise rate. A similar trend of performance can be observed for SoftTriple + PRISM.

Real-world Noisy Datasets. Table 3 displays the performance on two datasets with real-world label noise, CARS-98N and Food-101N [21]. In both datasets, using Soft Triple with PRISM achieves the best performance. In CARS-98N, MCL performs worse than the naïve contrastive loss [7] because the presence of noise in the memory bank hurts the performance of MCL. However, after adding PRISM to MCL, we improved Precision@1 by as much as 19%. However, we do not observe performance improvement on Food-101N from incorporating PRISM into MCL. We attribute this to an interesting characteristic of Food-101N, that many small clusters of open-set noise are often unique to one ground-truth class. For example, images of oatmeals often appear in the class `apple-pie`, but not in similar classes like `crab-cakes` or `chocolate-cake`. As a result, the model may have learned to treat oatmeals to be special apple pies. The multi-center SoftTriple loss is less susceptible to this phenomenon.

Training Time. Table 4 reports the time required for 5,000 training iterations. PRISM without centers (Eq. (2)) requires the longest training time because it needs to calculate average similarities of all classes for each minibatch

Table 4: Training time required with and without PRISM for 5,000 iterations on the SOP dataset and 10% symmetric label noise. The time recording starts when the memory bank is completely filled at iteration 3,000.

Algorithm	Training Time (Seconds)
Memory Contrastive Loss (MCL)	1,679.22
MCL + PRISM without centers	12,294.76
MCL + PRISM with centers	1,777.38
Soft Triple	1,685.47
Soft Triple + PRISM with centers	1,767.97

data. However, by maintaining center vectors to identify noise (Eq. (7)), the required training time decreases significantly. PRISM only incurs 6% more training time than that of Memory Contrastive Loss on SOP Similar observation can be found when we change the loss function to Soft Triple [32]. On CARS and CUB dataset, PRISM incurs up to 10.4% more training time. Due to space limitation, we show detailed results in the supplementary material. The results show that PRISM is an efficient method to handle noisy labels.

4.5. Ablation Study

Design of $P_{\text{clean}}(i)$. We compare PRISM against slightly different designs of the clean-data identification function $P_{\text{clean}}(i)$. As baselines, we adopt the following designs of $P_{\text{clean}}(i)$, which are compared with Eq. (2). The first baseline, Batch-positive, uses the average similarity between x_i and same-class samples within the minibatch. This approach neither uses the memory bank nor considers the negative samples.

$$P_{\text{clean}}(i) \propto \frac{1}{N_{y_i}} \sum_{(x_j,y_j)\in\mathcal{B}, y_j=y_i} S(f(x_i), f(x_j)) \tag{12}$$

The second baseline, Memory-positive, uses the average similarity between x_i and other same-class samples retrieved from the memory bank, but still does not consider the negative samples.

$$P_{\text{clean}}(i) \propto \frac{1}{N_{y_i}} \sum_{(v_j,y_j)\in\mathcal{M},\, y_j=y_i} S(f(x_i), v_j) \tag{13}$$

Table 5 reports the performance on CARS with 25% Small Cluster noise using different noise-filtering strategies with the threshold m determined by TRM. We use Soft Triple [32] as the loss function. All filtering strategies improve performance relative to the scenario of no noise filtering. The Memory-positive strategy works better than Batch-positive, showing the importance of using the entire memory bank to identify noise. PRISM, in the form of Equation (2), achieves the best P@1 and MAP@R scores.

Table 5: Precision@1 (%) and MAP@R (%) under different designs of the noise identification function. We use TRM to determine the threshold m for all methods.

	P@1	MAP@R
No noise filtering	73.40	15.21
Batch-positive filtering, Eq. (12)	74.66	16.44
Memory-positive filtering, Eq. (13)	75.12	17.11
PRISM, Eq. (2)	**75.40**	**17.40**

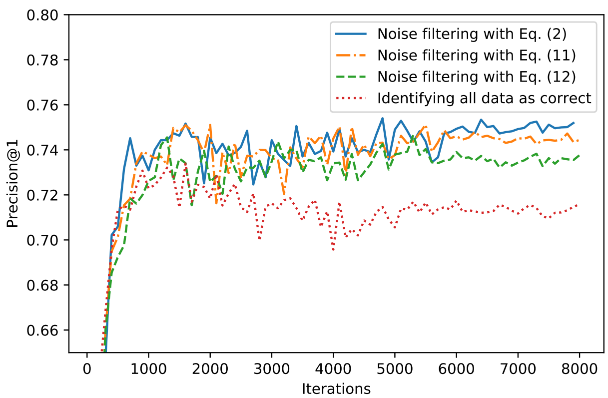

Figure 2: The Precision@1 (%) vs number of iterations on CARS with 25% Small Cluster Noise.

Figure 2 shows that the Precision@1 changes as training iterations increase. Ignoring label noise makes the model overfit to the noisy data, which is shown as good initial performance followed by a rapid fall. The batch-positive strategy, which does not use the memory bank, can improve performance, but also experiences a performance drop after 5,000 training iterations. Under the the PRISM strategy of Eq. (2), the model converges to the best performance.

TRM vs sTRM. Figure 3 illustrates the model performance for different choices of the sliding window size in sTRM. Note that TRM is a special case of sTRM, where the window size τ is set to 1. Across all different choices of τ, sTRM consistently outperforms TRM.

Noise rate R for CARS-98N. In PRISM, we use the R^{th} percentile of $P_{\text{clean}}(i)$ values to determine the threshold m for identifying noisy labels. Table 6 shows the performance under different R values. We use memory-based contrastive loss [46] as the loss function. The model achieves the best performance when $R = 50\%$. We can thus estimate that the noisy label rate of CARS-98N is approximately 50%. We also use $R = 50\%$ for Co-teaching and its extensions.

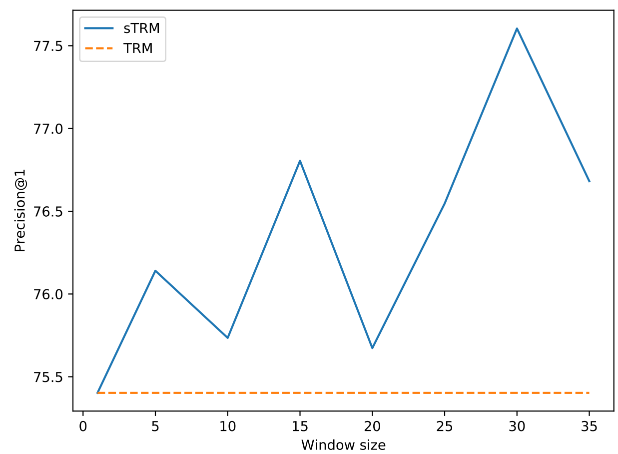

Figure 3: The Precision@1 (%) vs. window size τ. The dataset used is CARS with 25% Small Cluster noise.

Table 6: Precision@1 (%) and Mean Average Precision@R (%) when using different filtering rate R for identifying noisy label. Models trained with filtering rate $R = 50\%$ obtained the best performance.

R	0.0	0.3	0.4	0.5	0.6	0.7
P@1	38.73	49.43	52.43	**57.95**	54.91	51.27
MAP@R	3.34	5.47	6.29	**8.04**	7.43	6.29

5. Conclusions

In this paper, we propose a simple, efficient, and effective approach, Probabilistic Ranking-based Instance Selection with Memory (PRISM), to enhance the performance of deep metric learning in the presence of training label noise. Through extensive experiments with both synthetic and real-world datasets, we demonstrate that PRISM outperforms 12 existing approaches.

6. Acknowledgments

We gratefully acknowledge the support by the National Research Foundation, Singapore through the AI Singapore Programme (AISG2-RP-2020-019), NRF Investigatorship (NRF-NRFI05-2019-0002), and NRF Fellowship (NRF-NRFF13-2021-0006); Alibaba Group through Alibaba Innovative Research and Alibaba-NTU Singapore Joint Research Institute (Alibaba-NTU-AIR2019B1); the Nanyang Assistant/Associate Professorships; NTU-SDU-CFAIR (NSC-2019-011); NSFC No.91846205; the Innovation Method Fund of China No.2018IM020200; the RIE 2020 Advanced Manufacturing and Engineering Programmatic Fund (No. A20G8b0102), Singapore.

References

[1] Görkem Algan and Ilkay Ulusoy. Meta soft label generation for noisy labels. In *ICPR*, 2020.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

[4] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In *ECCV*, pages 548–564, 2020.

[5] Andrew Brown, Weidi Xie, Vicky Kalogeiton, and Andrew Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *ECCV*, 2020.

[6] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *CVPR*, pages 1861–1870, 2019.

[7] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005.

[8] Ismail Elezi, Sebastiano Vascon, Alessandro Torcinovich, Marcello Pelillo, and Laura Leal-Taixé. The group loss for deep metric learning. In *ECCV*, pages 277–294, 2020.

[9] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. *Advances in neural information processing systems*, 17:513–520, 2004.

[10] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254, 2017.

[11] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, pages 8527–8537, 2018.

[12] Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *ICCV*, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[16] Lu Jiang, Mason Liu Di Huang, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *ICML*, 2020.

[17] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, pages 2304–2313, 2018.

[18] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[19] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *CVPR*, pages 3238–3247, 2020.

[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV workshops*, pages 554–561, 2013.

[21] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, 2018.

[22] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, pages 5051–5059, 2019.

[23] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2016.

[24] Amit Mandelbaum and Daphna Weinshall. Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*, 2017.

[25] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*, 2019.

[26] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *ICCV*, pages 360–368, 2017.

[27] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *ECCV*, 2020.

[28] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Pytorch metric learning. *arXiv preprint arXiv:2008.09164*, 2020.

[29] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012, 2016.

[30] Kohei Ozaki and Shuhei Yokoo. Large-scale landmark retrieval/recognition under a noisy and diverse dataset. *arXiv preprint arXiv:1906.04087*, 2019.

[31] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, pages 1944–1952, 2017.

[32] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *ICCV*, pages 6450–6458, 2019.

[33] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, pages 5107–5116, 2019.

[34] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[35] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016.

[36] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *CVPR*, 2019.

[37] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *CVPR*, pages 6398–6407, 2020.

[38] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *ECCV*, 2020.

[39] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[40] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *NeurIPS*, pages 10–18, 2015.

[41] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[42] Dong Wang and Xiaoyang Tan. Robust distance metric learning via Bayesian inference. *IEEE Transactions on Image Processing*, 27(3):1542–1553, 2017.

[43] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *CVPR*, pages 5022–5030, 2019.

[44] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. Ranked list loss for deep metric learning. In *CVPR*, pages 5207–5216, 2019.

[45] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, and Neil M Robertson. Deep metric learning by online soft mining and class-aware attention. In *AAAI*, volume 33, pages 5361–5368, 2019.

[46] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. In *CVPR*, pages 6388–6397, 2020.

[47] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *CVPR*, pages 8688–8696, 2018.

[48] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, pages 13726–13735, 2020.

[49] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2 – a large-scale benchmark for instance-level recognition and retrieval. In *CVPR*, 2020.

[50] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *ICCV*, pages 2840–2848, 2017.

[51] Chen Xing, Sercan Arik, Zizhao Zhang, and Tomas Pfister. Distance-based learning from errors for confidence calibration. In *ICLR*, 2020.

[52] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *ICML*, pages 7164–7173, 2019.

[53] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. In *BMVC*, 2019.

[54] Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. Meta label correction for noisy label learning. In *AAAI*, volume 35, 2021.

[55] Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. Fewer is more: A deep graph metric learning perspective using fewer proxies. In *NeurIPS*, 2020.