Ensemble Learning with Manifold-Based Data Splitting for Noisy Label Correction

Hao-Chiang Shao, *Member, IEEE,* Hsin-Chieh Wang, Weng-Tai Su, *Student Member, IEEE*, and Chia-Wen Lin, *Fellow, IEEE*

***Abstract*—Label noise in training data can significantly degrade a model's generalization performance for supervised learning tasks. Here we focus on the problem that noisy labels are primarily caused by mislabeled confusing samples, which tend to be concentrated near decision boundaries rather than uniformly distributed, and whose features should be equivocal. To address the problem, we propose an ensemble learning method to correct noisy labels by exploiting the local structures of feature manifolds. Different from typical ensemble strategies that increase the prediction diversity among sub-models via certain loss terms, our method trains sub-models on disjoint subsets, each being a union of randomly selected seed samples' nearest-neighbors of the same class on the data manifold. As a result, only a limited number of sub-models will be affected by locally-concentrated noisy labels, and each sub-model can learn a coarse representation of the data manifold along with a corresponding graph. The constructed graphs are used to suggest a set of label correction candidates, and accordingly, our method determines label correction results by majority decisions. Our experiments on real-world noisy label datasets demonstrate the superiority of the proposed method over existing state-of-the-arts.**

***Index Terms*—Ensemble Learning, Label Noise, Data Splitting, Graph Representation, Label Correction.**

I. Introduction

Learning from noisy data is generally a vital issue in representation learning because of two reasons. First, as revealed in [1], given a classification model $\Phi(\cdot, \theta)$ learned by optimizing its parameter θ via the cross-entropy loss on the predicted label, its expected error is affected by i) the label noise, ii) the selected learning model Φ, and iii) the model parameter θ learned in the training process. Then, once the learning model Φ—suitable or not—is determined, the influence brought by Φ and θ is fixed. Hence, the only factor that may downgrade the model performance becomes the label noise within the ground-truth training *sample-label* pairs. Second, because it is expensive and time-consuming to collect data with reliable clean labels, in real-world applications people usually resort to exploiting low-cost datasets, e.g., those collected by search engines and labeled by quick annotators or crowd-sourcing, for training purposes. These low-cost datasets usually contain a certain amount of noisy labels and thus are not as reliable as professionally-labelled datasets like COCO [2] and ImageNet [3]. Consequently, how to learn from noisy data, including how to correct noisy data, has become a crucial task in real-world applications.

Label noise can be roughly divided into two types: random label noise and confusing label noise, as illustrated in Fig. 1. The former typically involves mismatched descriptions or tags usually due to the negligence of an annotator. For this type of error, not only a label error may occur randomly in the sample space, but also the erroneous label is often of another irrelevant random class. In contrast, the latter is the main cause of noisy labels in real-world applications and usually occurs when a to-be-labeled sample contains confusing contents or equivocal features. Confusing label noise often occurs on data samples lying near decision boundaries, and such a confusing noisy label should be corrected as one of the categories neighboring to the current one in the feature space.

Most existing methods focused on correcting random label noise and reported promising performance on simulated datasets with synthetic noisy labels like "noisy MNIST" [4] and "noisy CIFAR-10" [5] obtained by randomly altering data labels. Primarily, there are two sorts of strategies for solving the noisy label correction problem: transition matrix-based approaches [6]–[8] and class-prototype-based ones [9], [10]. All these methods learn from a given noisy dataset to correct noisy labels according to the distribution of labels within a local neighborhood. As a result, they cannot well cope with real-world confusing label noise robustly because confusing labels usually lie near the decision boundaries between categories and may be surrounded by other instances with incorrect confusing labels. Moreover, these methods may also give a wrong suggestion to a clean instance once it is located in the transition area between two classes akin in the feature domain.

In contrast, several methods resort to noise-robust loss functions to fight against label noise [11]–[14]. Such noise-robust loss functions work under the assumption that whether an instance is mislabeled has nothing to do with its content. That is to say, the distribution of noisy labels is still assumed to be independent of that of data instances. Because this

Manuscript received February 28, 2021; revised August 2, 2021 and September 23, 2023; accepted October 8, 2021. Date of publication Month Date, 2021; date of current version Month Date, 2021. This work was supported in part by Ministry of Science and Technology, Taiwan, under Grant 110-2634-F-007-015. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Xian-Sheng Hua. (Corresponding author: Chia-Wen Lin)

Hao-Chiang Shao is with the Department of Statistics and Information Science, Fu Jen Catholic University, New Taipei City, Taiwan. (email: shao.haochiang@gmail.com)

Hsin-Chieh Wang and Weng-Tai Su are with the Department of Electrical Engineering, Nation Tsing Hua University, Hsinchu, Taiwan. (email: lucas85062055@gmail.com; s105061805@m105.nthu.edu.tw)

Chia-Wen Lin is with the Department of Electrical Engineering and the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, Taiwan, and with Electronic and Optoelectronic System Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan. (email: cwlin@ee.nthu.edu.tw)

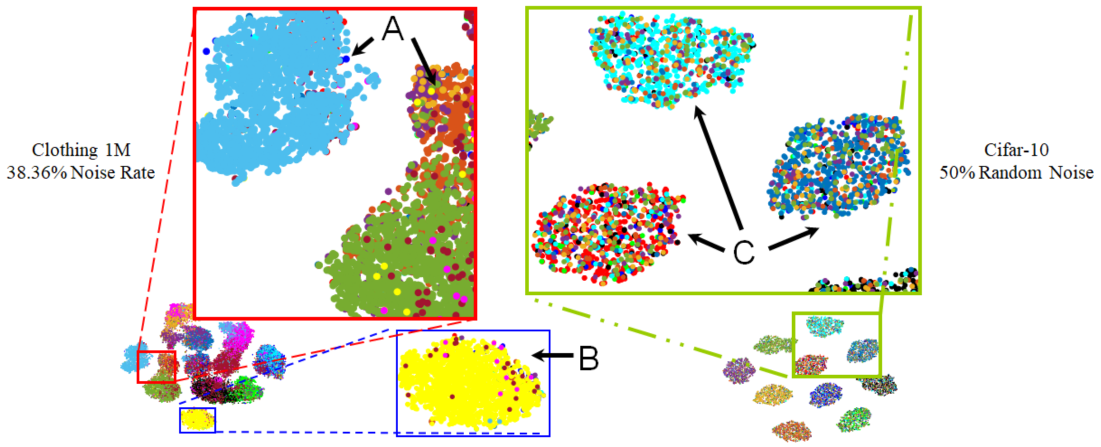


Fig. 1. Illustration of confusing label noise versus random label noise. Left: the t-SNE map of instance features of Clothing1M. Right: the t-SNE map of randomly perturbed CIFAR-10. Note that the dots in different colors denote samples of different categories. In real-world applications, noisy labels (i.e., those dots with different colors in a category) are locally concentrated near the decision boundaries, as indicated by the arrows in (A) and (B), and the synthetic random noise shown in CIFAR-10 simulations are actually rare in real-world data, as contrasted by (B) and (C).

assumption is only valid in the case of random label noise, these methods usually cannot perform well on real-world data involving confusing label noise.

Still, a few label correction methods are ensemble learning-based approaches [15]–[17]. By taking a majority decision on predictions of multiple sub-models, ensemble learning approaches can improve the robustness of predictions. Some ensemble learning approaches are based on semi-supervised learning strategies, aiming to increase the freedom of predictions via suitable data augmentation schemes [18], [19]. Nevertheless, common data augmentation schemes cannot synthesize data distributed far more differently from that of the original dataset, especially for those with confusing noise labels. Therefore, these approaches usually need to collaborate with additional loss functions to guarantee the correctness of soft-labels predicted by different sub-networks. Although these methods were designed to increase the uncertainty of sub-networks' predicted labels on noisy samples, they may also decrease the confidence of sub-networks' predictions on clean instances. Hence, corrections suggested by these methods are usually considered as soft-labels for training another student model. A primary drawback of these methods is that they do not take the within-neighborhood label consistency into account, and thus the obtained soft-labels may still be noisy.

To address the above problems with confusing label noises, we propose an ensemble-based label correction algorithm by exploiting the local structures of the data manifold. As illustrated in Fig. 2, our noisy label correction scheme involves three iterative phases: i) k-NN based data splitting, ii) multi-graph label propagation, and iii) confidence-guided label correction. Because confusing label noise tend to be located densely near decision boundaries, they may destroy the label smoothness within some local "inter-class transition bands" on the graph. By partitioning the source noisy dataset into disjoint subsets using our k-NN splitting scheme, each noisy label, along with its k-nearest-neighbors, will usually affect only a minority of the ensemble branches. As a result, each ensemble branch generates a graph holding its own unique noisy local manifold structures that will be treated as outliers during the majority decision process. To this end, we train the ensemble branches on the corresponding disjoint subsets independently. Through this design, each sub-network can learn not only a coarse global representation of a data manifold, but also different local manifold structures. We then derive label correction suggestions for each sample by propagating labels within each disjoint subset through graphs constructed for individual ensemble branches. Consequently, our method suggests final label corrections by ruling out inconsistent suggestions. Extensive experiments show the superiority of our method over existing state-of-the-arts.

The novelty of the proposed method is threefold.

- We propose a novel iterative data splitting method to split training samples into disjoint subsets, each preserving some local manifold structure of source data while representing a coarse global approximation. This design can confine the negative influence of mislabeled instances to a minority of ensemble branches so that such noisy labels can be corrected through majority decisions.
- Our design contains a novel noisy-label branch that can stably provide a correct suggestion for within-class clean labels. Hence, this branch can boost the accuracy of label correction results, especially for datasets primarily containing confusing label noise.
- We adopt multi-graph label propagation, instead of a simple nearest-neighbor strategy, to derive label correction suggestions via different graph representations characterizing the manifold of the same noisy data. Hence, our method can take advantage of both nearest neighbors and manifold structures with the aid of graphs.

The rest of this paper is organized as follows. Some most relevant works are surveyed in Sec. II. Sec. III presents the proposed schemes for data splitting, multi-graph label propagation, and confidence-guided label correction. In Sec. IV, experimental results on public datasets with noise labels are demonstrated. Finally, conclusions are drawn in Sec. V.

II. Related Work

A. Loss Functions

To solve noisy label correction problem, several methods were developed by re-designing loss terms so that the learning system itself can re-weight the importance of training instances. For example, Wang et al. [20] employed an importance re-weighting strategy that enables the training on noisy data to reflect the results of noise free environment. Arazo et al. [21] evaluated their entropy-based per-sample loss for label correction by controlling the confidence of training sample via two weights. Still, some other losses, e.g., mean absolute error (MAE), generalized cross entropy (GCE), and reverse cross entropy (RCE), were proposed to avoid a biased learning system, the gradient saturation problem, or the overfitting/under-learning problem in the presence of noisy labels [12], [22], [23]. Nevertheless, although these methods prevent a learning system from fitting noisy samples, they may still be impracticable to complicated datasets due to the ignorance of hard samples.

B. Label Correction

Most recent label correction methods work based on i) a noise transition matrix, or ii) class prototypes. As for the former sort of methods, they are based on an instance-independent assumption to derive the transition probability, which depicts how possible a clean label flipping into a noisy one, independent to any type of data structure. However, this assumption cannot prevent the generation of incorrect labels, nor can it prevent the transition matrix from learning noisy labels. Therefore, Xia et al. [8] proposed a transition revision method to address this issue, but their method is still limited by the fairness of the estimated initial transition matrix. Moreover, "class prototype" methods learn to represent each class via some class prototypes, based on which data samples can receive corrected label information from their neighborhood [9], [10]. Nevertheless, these methods learn prototypes from a noisy dataset, so the quality and the accuracy of prototypes are still affected by the original noise distribution. Both these two types of methods learn from noisy data. Because there is no effective mechanism to verify the corrected labels suggested by the transition matrix or the class prototypes, the resulting pseudo-label may be incorrect and generate a worse correction suggestion if the class prototypes or the transition matrix overfits noisy training labels.

C. Ensemble learning

Some other methods exploit i) ensemble learning that feeds stochastically-augmented inputs to a number of parallel sub-models to derive the corresponding prediction jointly with ii) an averaging method to derive label correction suggestions [16], [19]. These methods are based on the inconsistency among information learned by different sub-networks. For example, MLNT [15] randomly changes the labels of each sub-network's training data to enhance the difference learned by each subnet, and DivideMix [16] uses an additional regularization term to increase the prediction diversity among different sub-models. However, making sub-models learn to be different from each other cannot ensure the correctness of any sub-model itself, and they may be worse than they are supposed to be due to an unmatched noise distribution. For example, Yan et al.'s method works only on uniformly distributed label noises [24]. In real-world applications, label noise usually distributes in some specific way rather than uniformly. Consequently, it is still intractable for simple ensemble learning approaches to filter out noisy labels.

D. Label Propagation

Label propagation is a graph-based semi-supervised method for generating pseudo-labels of unlabeled data based on given anchors [25]–[27]. Compared with the nearest-neighbor strategy that does not take the data manifold into account, label propagation methods can propagate labels from labeled anchors to unlabeled samples through a graph characterizing the manifold of the noisy dataset. For example, the method in [28] uses an affinity matrix to derive an approximate k-NN graph, through which it performs label propagation for a large-scale diffusion task; and, Iscen et al. [26] proposed a transductive label propagation method to make label predictions by using i) a nearest-neighbor graph, ii) conjugate gradient based label propagation, and iii) a loss weighted by label certainty. Moreover, Bontonou et al. [29] proposed a graph smoothness loss, maximizing the distance between samples of different classes, to derive better features and manifold representations for classification purpose. Because this loss function takes only distances between instances in different classes into account, the performance of Bontonou et al.'s method is similar to those of cross-entropy based methods. However, the graph smoothness loss increases the robustness of a learning model against noisy training inputs and is thus worthy to be integrated into a noisy label correction scheme.

III. Proposed Method

A. Overview

Based on the assumption that real-world noisy labels (e.g., annotated via crowd-sourcing) tend to be locally concentrated near decision boundaries, our key idea is to confine the influence of such locally-concentrated label noise to only a minority of sub-networks via our local-patch-based data slitting strategy. As a result, most ensemble branches can learn their own relatively correct approximations of the data manifold around the noisy local patch, and a suitable label correction result can be yielded by majority decision accordingly. Also, no matter whether label noise is locally-concentrated or uniformly distributed in the data space, it can be effectively mitigated by the proposed ensemble learning method. Fig. 2 illustrates the framework of the proposed noisy label correction scheme, where each training epoch of the proposed method can be divided into three phases, as will be elaborated in Sec. III-B–III-D.

Let (x_i, y_i) denote the i-th *sample-label* pair in a source noisy dataset $(\mathcal{X}, \mathcal{Y})$, where $\mathcal{X} = \{x_i\}$ is the sample set and $\mathcal{Y} = \{y_i\}$ is the corresponding noisy label set. Our ensemble learning scheme aims to correct noisy labels by training

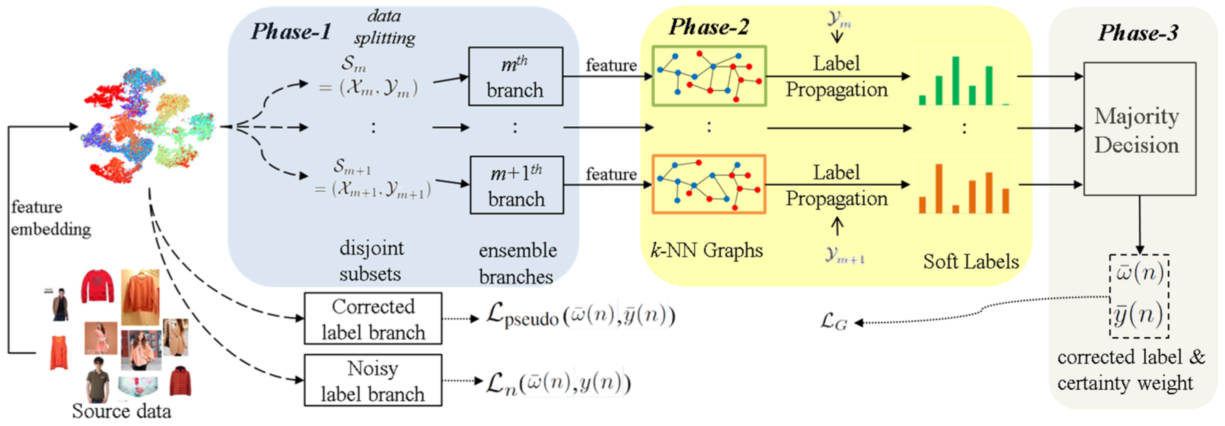


Fig. 2. Framework of our ensemble-based label correction scheme. It contains primarily three branches, namely i) noisy label branch controlled by $\mathcal{L}_n$, ii) corrected label branch controlled by $\mathcal{L}_{pseudo}$, and iii) ensemble branches controlled by $\mathcal{L}_G$. Our design focuses on the ensemble branches implemented in three phases. The noisy label branch is used to perform *feature embedding* and also used to regulate certainty weight jointly with the corrected label branch. Only the corrected label branch is used while deploying. The three phases of our ensemble branches are detailed in Sec. III-B–III-D.

M ensemble branches on M disjoint subsets $\{\mathcal{S}_m\}, m = 1, 2, ..., M$, derived by splitting $\mathcal{X}$. After performing the data splitting procedure described in Sec. III-B, our method first trains the m-th classifiers $\Phi_m(\cdot;\theta_m)$ on the m-th disjoint subset $\mathcal{S}_m$ through an ensemble branch. Second, by feeding all training samples $x_i \in \mathcal{X}$ into each classifier, we extract totally M different feature vector sets, each being used to construct one graph representation depicting the data manifold of $\mathcal{X}$. Third, based on the M different graph representations, we then take the m-th *partial label* set (x_j^m, y_j^m), where $x_j^m \in \mathcal{S}_m$, as an initial reference to evaluate the label correction suggestions for sample $x \in (\mathcal{X} \setminus \mathcal{S}_m)$ through label propagation. This step results in totally $M \times M$ label correction suggestions. Fourth, taking the corrected labels $\hat{y}_j^{m,l}$ of $x_j^m \in \mathcal{S}_m$ obtained in the l-th training epoch as additional references, we generate another $M \times M$ label correction suggestions through the M graphs. These four steps forms our second phase and will be elaborated in Sec. III-C. Finally, we derive the most likely labels $\bar{\mathcal{Y}} = \{\bar{y}_i\}$ from the total $2 \times M^2$ label correction suggestions via majority decision, as elaborated in Sec. III-D.

Specifically, because real-world noisy labels tend to be locally concentrated near decision boundaries, we devise a data splitting strategy to partition the source dataset $\mathcal{X}$ into M disjoint subsets $\mathcal{S}_m$, each of which can be expected to retain the local shapes of k randomly-selected local neighborhoods, no matter whether it is noisy or not, as well as to ensemble a random subset of $\mathcal{X}$. As a result, by making a majority decision on the approximate manifolds learned by the M sub-networks, the local influences brought by noisy samples can be effectively mitigated.

Consequently, since the *noisy label branch* in our design is trained on the whole $\mathcal{X}$ with original noisy labels $\mathcal{Y}$, i.e., (x_i, y_i), it can embed all $x_i \in \mathcal{X}$ into features as common classifiers do. Also, the *corrected label branch* is trained on corrected-label pairs $(x_i, \bar{y}_i)$, where $\bar{y}_i$ can be obtained after every training epoch. The corrected label branch is the only branch used for deployment.

Overall, each training epoch of our method can be divided into three phases, as will be elaborated in Sec. III-B–III-D.

TABLE I
LIST OF SYMBOLS

M	the number of ensemble branches
B	the number of data packages per class per branch
(x_i, y_i)	the i-th image sample and its label
$\mathcal{X}, \mathcal{Y}$	the sets of x_i and y_i, respectively
$\hat{y}_j^{m,l} \in \hat{\mathcal{Y}}_m$	the label correction suggestion of x_j given by the m-th ensemble branch after the l-th epoch
$\mathcal{S}_m$	the m-th disjoint subset after data splitting
$\Phi_m(\cdot)$	the classifier learned by the m-th ensemble branch
θ_m^l	the model parameter of Φ_m (at the l-th epoch)
$\bar{y}_i \in \bar{\mathcal{Y}}$	the most likely label correction result of x_i based on per-epoch $2 \times M^2$ label suggestions
$f_i^{m,l}$	the feature of x_i extracted by the m-th model at the l-th epoch
W_m	normalized weighted undirectional adjacency matrix representing the graph constructed based on f_i^m
(x_j^m, y_j^m)	sample-label pair for deriving label correction suggestions via label propagation
$(x_j^m, \hat{y}_j^{m,l})$	sample-correction pair for deriving label correction suggestions via label propagation
$Z_{m,n}$	matrix recording soft-labels derived from the m-th graph and reference labels for samples in $\mathcal{S}_n$
$\omega_{m,n}(i,:)$	the confidence level of $Z_{m,n}$
$\bar{\omega}_i$	the normalized confidence level of $\bar{y}_i$ defined in (6)
$\hat{\omega}_i$	the average confidence level defined in (7)
Y_n	$N \times C \times 2$ matrix recording two reference partial labels, y_j^n and $\hat{y}_j^{n,l}$, of data samples belonging to $\mathcal{S}_n$
$\hat{Y}_{m,n}(i,:)$	soft-label suggested by the m-th ensemble branch

In addition, for the sake of clarity, all notations used in this paper are summarized in Table I.

B. Phase-1: Data Splitting/Re-splitting

The first phase of each training epoch is to randomly scatter per local neighborhood of data points on the source data manifold, including those with noisy labels, into disjoint subsets. Specifically, as illustrated in Fig. 3, for each selected

x_i, we pick only its k nearest-neighbors of the same class to form a *local patch* around x_i with $k = \frac{N_c}{(M \cdot B)}$, where N_c is the number of samples within the class that x_i belongs to, and B and M are two predefined hyper-parameters: the number of local manifolds and the number of ensemble branches, respectively. Therefore, because each disjoint subset $\mathcal{S}_m$ holds different local feature relationships of each class, we can confine the negative influence of each locally-concentrated noisy neighborhood to primarily one $\mathcal{S}_m$ and one ensemble branch by packing every local k-nearest neighbors. This is the main advantage of the proposed method. As a result, when a noisy local neighborhood only contaminates a minority of ensemble branches, our method can vote down the negative influence of noisy labels by a majority decision. This case leads to a performance leap with our method: the best case. Meanwhile, in the extreme case that all noisy labels are uniformly distributed globally, our data splitting strategy will not alter the noise distribution so that our ensemble model leads to a as good performance as typical random-selection schemes: the baseline. In sum, our method can thus achieve a performance in between the best case and the baseline. Since real-word confusion label noise distribution tends to be concentrated around decision boundaries, our method can usually lead to a performance improvement.

The proposed data splitting strategy is composed of the following three steps:

Initialization: This step determines the hyper-parameters, including the number of ensemble branches M and the number of data packages per class per branch B. First, we estimate a rough yet acceptable number of clusters within the c-th class by using conventional k-means clustering and Silhouette analysis. Then, the number $M \cdot B$ should be chosen so that the number of clusters be an integer multiple of $M \cdot B$. Finally, we select a suitable number of ensemble branches M so that the number of local manifold B can fall within its suitable range, e.g. $B \in [4, 16]$, as will be discussed in Sec. IV.

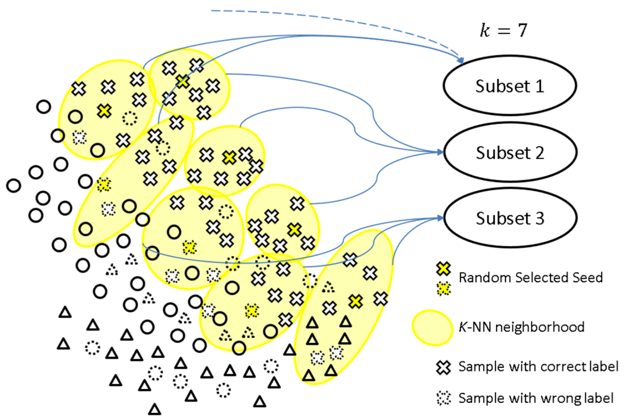
k-NN Packing: As illustrated in Fig. 3, for each class, we separate samples within the class into $M \cdot B$ packages, each containing a randomly selected seed sample and its k-NN in the feature domain with $k = \frac{N_c}{M \cdot B}$, where N_c denotes the number of samples in the c-th class. Note that "random sampling without replacement" is used to guarantee that each sample and its k-NN can be assigned into only one package.

Splitting: For each class, we randomly assign its $M \cdot B$ packages into M disjoint subsets. Each subset $\mathcal{S}_m$ therefore represents a coarse global approximation of the source data manifold but also holds fine local manifold structures of different places independently.

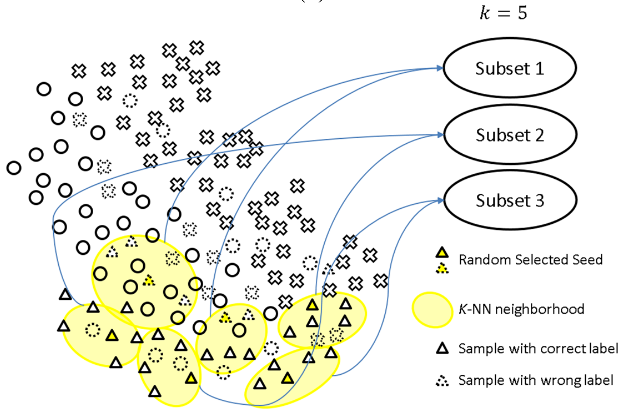
Moreover, we initialize each training epoch with the data splitting process, and we name this strategy **re-splitting**. Re-splitting enables each resemble branch to learn a different coarse approximation of the data manifold to prevent itself from overfitting and from being biased to the same training data. Because $\mathcal{S}_m$ is changed at the beginning of each epoch, we need to initialize the model parameter θ_m^l for the m-th ensemble branch to guarantee the fast convergence of the l-th training epoch by

$$\theta_m^l = (1 - \alpha_m)\theta_{\text{pseudo}}^{l-1} + \alpha_m \theta_{\text{noisy}}^{l-1}, \tag{1}$$

where $\alpha_m^l \in [0, 1]$ is a random value, and $\theta_{\text{pseudo}}^{l-1}$ and $\theta_{\text{noisy}}^{l-1}$ are the model parameters of the *corrected-label branch* and the *noisy-label branch*, respectively, learned in the previous training epoch.



(a)



(b)

Fig. 3. Conceptual illustration of our data splitting strategy. For each randomly selected seed (highlighted in yellow), we pick up only its **k-NN of the same class** (highlighted in white) to form a package, where $k = \frac{N_c}{MB}$, where N_c denoting the number of instances of class-c. This figure illustrates that each subset can be regarded as a random subset of the whole dataset if $B = 1$. (a) An example of splitting "crosses" with $k = 7$. (b) An example of splitting "triangles" with $k = 5$. Note that the *dashed* samples are mis-annotated ones.

C. Phase-2: Multi-Graph Label Propagation

We then construct the m-th graph representing the data manifold based on features $f_i^{m,l} = f(x_i, \theta_m^l)$ extracted by the m-th model. Then, for each graph, we can derive $M + M$ label correction suggestions from two different starting partial-labels, including i) the original sample-label pair (x_j^m, y_j^m) for $x_j^m \in \mathcal{S}_m$, and ii) the sample-correction pair $(x_j^m, \hat{y}_j^{m,l})$ for $x_j^m \in \mathcal{S}_m$ obtained in the l-th training epoch, where $\hat{y}_j^{m,l}$ denotes the j-th sample's label correction suggestion given by the m-th ensemble branch in the l-th training epoch. As a result, our method can hence offer totally $M \cdot (M + M) = 2M^2$ sets of label correction suggestions.

Our multi-graph label propagation strategy works based on the features extracted by classification models $\Phi_m(\cdot;\theta_m^l)$ of the M ensemble branches. The first step is to build M normalized weighted undirectional adjacency matrices $\mathcal{W}_m$, representing the m-th graph constructed based on f_i^m, for the label propagation process described in (4) [25] as follows:

$$\mathcal{W}_m = D_m^{-\frac{1}{2}}(A_m + A_m^T)D_m^{-\frac{1}{2}}, \tag{2}$$

where D_m is a diagonal matrix used to normalize $(A_m + A_m^T)$, and A_m is a directional adjacency matrix whose (s,t)-th entry is defined as follows:

$$A_m(s,t) = \begin{cases} \mathtt{sim}(f_s^m, f_t^m)^\gamma & \text{, for } f_s^m \in \mathcal{N}(f_t^m), s \neq t \\ 0 & \text{, otherwise} \end{cases} \tag{3}$$

where f_s^m is the feature of the s-th sample extracted by the model of the m-th branch, $\mathcal{N}(f_t^m)$ denotes the collection of the k-NN of f_t^m, $\mathtt{sim}(\cdot,\cdot) \in [0,1]$ denotes the cosine similarity metric, and γ is a predefined hyper-parameter. Note that A_m is asymmetric, so as $\mathcal{W}_m$.

Next, we propagate the label information of i) the sample-label pairs (x_j^m, y_j^m) in the m-th disjoint subset $\mathcal{S}_m$ and ii) the sample-correction pairs $(x_j^m, \hat{y}_j^{m,l})$ in turn through each of the M graphs to obtain totally $2M^2$ label correction suggestions, i.e., partial-labels, for each data sample. We adopt such a strategy because of the following two concerns. First, when using only original noisy labels y_j^m to derive the label propagation result, the obtained labels are probably contaminated, making the accuracy of obtained labels limited. Second, when using only corrected labels $\hat{y}_j^{m,l}$, the propagation result may overfit the corrected labels, thereby misleading the label correction process. Therefore, we exploit both the original noisy labels y_j^m and corrected labels $\hat{y}_j^{m,l}$ in our label propagation process.

Let $Z_{m,j}$ denote the label correction suggestion. The label propagation process [25] is expressed by

$$Z_{m,n} = (\mathbf{I} - \alpha\mathcal{W}_m)^{-1}Y_n, \tag{4}$$

where Y_n is an $N \times C \times 2$ matrix that records the associated reference labels with the data samples belonging to the n-th disjoint subset $\mathcal{S}_n$, $\mathcal{W}_m$ is the normalized weighted undirectional adjacency matrix defined in (2), and N and C respectively denote the cardinality of the original dataset $\mathcal{X}$ and the number of total classes in $\mathcal{Y}$. Note that the third dimension of Y_n contains two partial-labels: the original noisy one y_j^n and the corrected suggestion $\hat{y}_j^{n,l}$.

Finally, the soft-label suggested by the m-th ensemble branch based on the partial-label information of $\mathcal{S}_n$ becomes

$$\hat{Y}_{m,n}(i,:) = \underset{c}{\operatorname{argmax}}\, Z_{m,n}(i,c,:), \tag{5}$$

where $Z_{m,n}$ is an $N \times C \times 2$ matrix, where $Z_{m,n}(i,c,:)$ indicates how possible the i-th sample belongs to the c-th class. After obtaining $2M^2$ different label suggestions $\hat{Y}_{m,n}$, it turns to Phase-3 to derive the final correction result.

D. Phase-3: Confidence-Guided Label Correction

Although it is straightforward to select the most frequent category among all label suggestions $\hat{Y}_{m,n}(i,:)$ via a majority decision as the final label correction result $\bar{y}_i$ at the end of the first training epoch, we still need a measure to describe the confidence of the correction result and then to guide the training process of the next epoch. Hence, we devise this Phase-3 process to evaluate the confidence of $\bar{y}_i$ at the end of each training epoch.

To guide the label correction, we propose a normalized average confidence level to devise our loss function. The normalized average confidence level $\bar{\omega}_i$ of the final label correction result $\bar{y}_i$ is defined as

$$\bar{\omega}_i = \frac{\hat{\omega}_i - \min\{\hat{\omega}_i\}}{\max\{\hat{\omega}_i\} - \min\{\hat{\omega}_i\}}, \tag{6}$$

where

$$\hat{\omega}_i = \frac{1}{2M^2} \sum_{(m,n,q)\in\Omega_i} \omega_{m,n}(i,q). \tag{7}$$

where $\omega_{m,n}(i,q)$, defined later in (8), is the confidence level assessed based on the certainty weight described in [26], and $\Omega_i = \{(m,n,q) \mid \bar{y}_i = \hat{Y}_{m,n}(i,q)\}$ means that only those samples, whose label correction suggestions $\hat{Y}_{m,n}(i,q)$ are identical to the final corrected label $\bar{y}_i$, are used to calculate the normalized average confidence level. Therefore, $\bar{\omega}_i$ can be maximized when i) the predictions have a high confidence level, which implies a relatively low entropy, or ii) the i-th corrected labels suggested by all graphs indicate the same class.

The confidence level $\omega_{m,n}(i)$ of the label propagation result $Z_{m,n}$ is assessed based on the certainty weight described in [26]. That is,

$$\omega_{m,n}(i,:) = 1 - \frac{H(\bar{Z}_{m,n}(i,c,:))}{\log(C)}, \tag{8}$$

where

$$\bar{Z}_{m,n}(i,c,:) = \frac{Z_{m,n}(i,c,:)}{\sum_k Z_{m,n}(i,k,:)},$$

and

$$H(\bar{Z}_{m,n}(i,c,:)) = -\sum_{c=1}^{C} \bar{Z}_{m,n}(i,c,:)\log(\bar{Z}_{m,n}(i,c,:)),$$

where $\bar{Z}_{m,n}$ records the normalized occurrence frequency of each label in the label propagation result matrix $Z_{m,n}$, and $H(\cdot)$ denotes the entropy. Therefore, a large $\omega_{m,n}(i,q)$ implies a label $\hat{Y}_{m,n}(i,q)$ with high confidence (i.e., lower uncertainty).

After normalizing the final weights described in (6), we use i) the corrected labels and ii) the weighted cross entropy loss to train the next-epoch ensemble branches and the next-epoch corrected label branch. The loss functions used to train our ensemble model are detailed below.

E. Loss Functions

We adopt the following weighted cross-entropy to measure the loss of corrected (pseudo) labels branch:

$$\mathcal{L}_{\text{pseudo}} = -\sum_i \bar{\omega}_i\, \bar{y}_i \log(p_{\text{pseudo}}(i,\bar{y}_i))), \tag{9}$$

where $\bar{y}_i$ is the pseudo (corrected) label of x_i derived from our label correction process in the current training epoch, and $p_{\text{pseudo}}(i, \bar{y}_i)$ denotes the probability, predicted by the softmax layer of the corrected (pseudo) label branch, of x_i belonging to class $\bar{y}_i$.

Then, the noisy label branch is trained via the weighted cross-entropy loss. That is,

$$\mathcal{L}_{\text{noisy}} = -\sum_i \ell_{\text{noisy}}(i), \tag{10}$$

where

$$\ell_{\text{noisy}}(i) \quad = \quad \begin{cases} \bar{\omega}_i \, y_i \log(p_{\text{noisy}}(i,c)) & \text{, if } c = \bar{y}_i \\ (1-\bar{\omega}_i) \, y_i \log(p_{\text{noisy}}(i,c)) & \text{, otherwise.} \end{cases}$$

where p_{noisy} denotes the probability predicted by the softmax layer of the noisy label branch. This loss means that if the label of the i-th sample x_i is corrected, we give it a complementary confidence level $1 - \bar{\omega}_i$ to highlight its low confidence.

Moreover, because the M ensemble branches are mainly designed to construct the M graphs rather than predicting the classification probability of each sample, we exploit a graph smoothness loss modified from [29] to derive a set of graph embedding so as to maximize the inter-class distances, similar to the concept described in [30]. This loss uses radial basis functions to measure the weighted distance between the features of two different-class samples. Because two samples may come from different $\mathcal{S}_m$, this loss encourages a large enough between-class margin on all graphs. That is,

$$\mathcal{L}_G = \sum_{(s,t)} \sqrt{\bar{\omega}_s \bar{\omega}_t} \exp\left\{ -\alpha \| \mathbf{p}_s^{\mathcal{I}(s)} - \mathbf{p}_t^{\mathcal{I}(t)} \|_2 \right\}, \tag{11}$$

where $(s,t) \in \{(s,t) \mid \bar{y}(s) \neq \bar{y}(t)\}$, $\mathcal{I}(s)$ is an indicator function that returns the index of the disjoint subset containing the s-th sample, and $\mathbf{p}_s^{\mathcal{I}(s)}$ denotes the s-th sample's probability vector generated by the softmax layer of the subnetwork trained on the $\mathcal{I}(s)$-th subset.

In (11), $\mathcal{L}_G$ has an additional weighting factor $\sqrt{\bar{w}_s \bar{w}_t}$ that gives a lower weight to an instance pair with a lower confidence level to alleviate possible negative effects brought by the instance pair. Note, $\mathcal{L}_G$ is different from typical cross-entropy loss and triplet loss [31] designs because $\mathcal{L}_G$ aims to map instances of different classes into feature clusters far away from each other. On the contrary, both cross-entropy loss and triplet loss encourage the network to map instances of the same class into the similar features. Consequently, $\mathcal{L}_G$ can assist our model to produce distinguishable features for label correction.

Finally, the total loss of the proposed model is

$$\mathcal{L}_{\text{total}} = \xi_1 \mathcal{L}_G + \xi_2 \mathcal{L}_{\text{pseudo}} + \xi_3 \mathcal{L}_n, \tag{12}$$

where we set $\xi_1 = \xi_2 = \xi_3 = 1$ empirically.

IV. Experimental Results

A. Datasets

We evaluate the proposed method on i) simulated noisy datasets based on CIFAR-10/100 and ii) two other real-world noisy datasets: **Clothing1M** [32] and **Food101-N** [33]. The properties of Clothing1M and Food101-N are described below.

First, Clothing1M contains images with 14 categories of fashion clothes. The data in Clothing1M are collected from the Internet, and their category labels are generated based on the surrounding texts, which leads to a label noise rate of 38.46%. Clothing1M also provides a small clean set with labels corrected by human annotators. The numbers of images in Clothing1M's clean-label training, validation, and testing sets, and the noisy-label set are about 47.5k, 14.3k, 10.5k, and 1M, respectively. In the four sets, only the 1M noisy-label set contains noisy labels. We use the noisy-label set to train our model to evaluate its performance, and then we conduct the ablation study jointly on the noisy-label set and the human-annotated clean testing set.

Second, Food101-N contains about 310K food images of 101 categories. The images in Food101-N are collected from the Internet with a noisy label rate of 20%. To evaluate the performance of our method, we apply it on Food101-N and derive a label correction result first, and then we train a classifier on the label-corrected dataset and conduct testing on Food-101's testing set [34] that contains 55K images with clean labels.

B. Experiment Settings

We summarize our experiment settings into the following four points. First, after the data splitting phase of each training epoch, we resize training images $x_i^m \in \mathcal{S}_m$ to 256×256, and then cropped them randomly into 224×224 patches. Hence, training samples for each training epoch are different variants of the source data, which improves the diversity of training samples for each epoch. Second, the optimizer is SGD, and the momentum value is 0.9. The initial learning rate is 0.01, and it is reduced by a factor of $\frac{1}{10}$ every 5 epochs. The maximum training epoch is set to be 15, and the batch size is 64. We also adopt ridge (L2) regularization weighted by 5×10^{-3}. Third, the number of branches is $M = 5$, and the number of data packages per class in each branch is $B = 4$. Fourth, the features of all training samples are extracted *initially* by a ResNet-50 [35] pretrained on ImageNet. After the first training epoch, the features used for our data (re-)splitting phase are derived by the latest network model trained in the noisy-label branch.

C. Experiments on Clothing1M

We evaluate our model's performance on Clothing1M in two different scenarios. The first scenario is to train our model and the compared state-of-the-arts on only noisy-label datasets without any additional supervision. In the second scenario, all models are trained on the same hybrid set consisting of i) the noisy-label set and ii) the clean set consisting of 50K human-annotated images, and then the obtained models are further fine-tuned on the 50K-image clean dataset via the cross-entropy loss. The results of the first and second scenarios are shown respectively in Tables II and III. Note that in the second scenario, the labels of the clean dataset are kept unchanged, and their certainty weight are set to be 1. Table II demonstrates that our method achieves the best accuracy of 75.17% compared with the other methods. Meanwhile, in the

TABLE II
CLASSIFICATION ACCURACY OF DIFFERENT METHODS ON THE NOISY-LABEL SET OF CLOTHING1M

Method	Accuracy (%)
CrossEntropy	69.21
MLNT [15]	73.47
T-Revision [8]	74.18
Self-Learning [10]	74.45
MetaCleaner [36]	72.50
DivideMix [16]	74.76
NoiseRank [9]	73.82
ELR [37]	74.81
Ours	**75.17**

second scenario, our method still outperforms all compared state-of-the-arts without any fine-tuning and is comparable to Self-Learning [10] after fine-tuning with the 50K clean set.

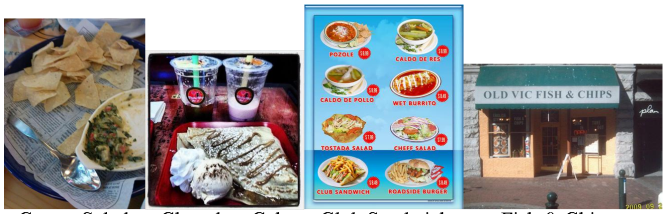
Note that in this experiment, Self-Learning achieve the best result based on the "additional supervision" provided by the 50K clean set. However, our method can achieve the state-of-the-art performance without the need of such additional supervision, making it more practical and generalizable than Self-Learning, as will be discussed in Sec. IV-E.

TABLE III
CLASSIFICATION ACCURACY OF DIFFERENT METHODS ON A HYBRID SET OF THE NOISY SET AND THE CLEAN SET, CONTAINING 50K IMAGES, OF CLOTHING1M

Method	Accuracy (%)	Fine-tuning
CrossEntropy	75.67	x
DataCoef [38]	77.21	x
Ours	**78.01**	x
CrossEntropy	80.32	o
Forward [6]	80.38	o
MetaCleaner [36]	80.78	o
CleanNet [33]	79.90	o
Self-Learning [10]	**81.16**	o
Ours	81.13	o

D. Experiments on Food101-N

Table IV shows that the classifier, trained on the data corrected by our method, achieves the second best accuracy of 85.13% on Food101-N, comparable to the best performance of 85.79% achieved by NoiseRank [9]. Note, typical data cleaning methods like NoiseRank [9] require both a fine-tuning dataset and the information of class prototypes as additional supervision to clean/remove mis-labeled data. The 50K "known clean set" used in this experiment in Table IV exactly matches this presumption of NoiseRank, making it performs slightly better than our method that does not rely on such additional supervision. Nevertheless, data cleaning methods tend to remove hard samples simultaneously, leading to irreversible information loss. Consequently, our method works much better than NoiseRank on Clothing1M, a dataset containing no anomalies.



Caesar Salad Chocolate Cake Club Sandwich Fish & Chips

Fig. 4. Images samples of three multi-category images and one anomaly (the rightmost) in Food101-N. The left three images all contains multiple food categories, but only one of the categories is labeled for each image, making their labels ambiguous semantically. The rightmost shows a shopfront photo containing no food but mislabeled as "fish & chips".

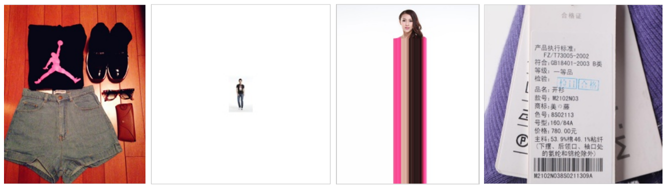


Fig. 5. Image samples with semantic ambiguity in clothing1M. From left to right: i) a multi-category image with one single label, ii) a person too far away and unsure about his clothes, iii) an image containing no clothes but being attributed to a clothes category, and iv) an image of clothing tag and barcode.

E. Comparison of Model Performances on Real-world Data

Clothing1M and Food101-N are very challenging datasets. They at least involve i) confusing labels around decision boundaries rather than typical random noises, and ii) multi-category images with a single label, e.g. photos with multiple food categories but with only one of the categories annotated as illustrated in Fig. 4 (the first three images) and Fig. 5 (the first image), and iii) anomalies, e.g. a shopfront photo labeled as "fish and chips" (the fourth image of Fig. 4), an image containing no clothes but being annotated as a clothes category (the third image of Fig. 5), and an image of clothing tag and barcode (the fourth image of Fig. 5). Because the datasets contain numerous semantically ambiguous labels, the accuracy values of the state-of-the-art methods are distributed in a small range as shown in Tables II, III, and IV. Therefore, a performance improvement of 1–2% accuracy is considered significant for these challenging datasets. Note, there is no effective way so far to tackle the semantic ambiguity caused by the second type (multi-category images with a single label) and third type (anomalies) without data cleaning or additional information. Our method therefore focuses on dealing with confusing label noise around decision boundaries. The results show that our approach indeed achieves notable improvements stably over the previous state-of-the-arts under such a diversity of confusions.

Next, because the testing data of the experiment sets shown in Tables III and IV match the technical presumption of Self-Learning [17] and NoiseRank [9], our method seems to be only comparable with them. Specifically, taking Self-Learning in Tables III and IV for example, it first finds clean cluster centers based on the clean images in the hybrid training set containing a "*known clean subset*" of images, and then takes these clean clusters as prior information for noise label correction. Hence, with a known clean training subset, Self-Learning

TABLE IV
CLASSIFICATION ACCURACY OF DIFFERENT METHODS ON FOOD101-N DATASET WITH ADDITIONAL 50K CLEAN SET.

Method	Accuracy (%)
CrossEntropy	81.97
CleanNet [33]	83.95
MetaCleaner [36]	85.05
Self-Learning [10]	85.11
NoiseRank [9]	**85.79**
Ours	85.13

becomes a semi-supervised method, and its performance is improved accordingly. However, if a training dataset does not provide “known clean labels” (e.g., Clothing1M without clean-label samples), the performance of Self-Learning would be degraded as demonstrated in Table II. Similarly, NoiseRank requires both a fine-tuned dataset and the information of class prototypes to achieve its best performance. The 50K clean set used in the experiments shown in Table IV exactly matches this presumption of NoiseRank, making the performance of NoiseRank (85.79% accuracy) slightly better than ours (85.13% accuracy). However, the performance of NoiseRank is significantly degraded on noisy-label Clothing1M in Table II. By contrast, our method does not rely on such additional supervision while achieving the state-of-the-art performance. Therefore, our method is more practical and generalizable for real-world applications compared with Self-Learning and NoiseRank.

Moreover, since MLNT [15] and DivideMix [16] shown in Table II are two representative ensemble-based methods without any data splitting strategy, the superior performance of our method over them demonstrates that the proposed data splitting strategy is effective in fighting against real-world noisy dataset like Clothing1M.

F. Analyses and Discussions

In this subsection, we discuss the effects of individual hyper-parameters, including i) the number of local manifolds B, ii) the number of ensemble branches M, iii) the settings of $\mathcal{L}_G$, iv) the data re-splitting process, and v) the certainty weight.

• **Number of local manifolds** B

Fig. 6 shows that when $M = 5$, our method achieves the best accuracy 75.17% on the noisy set of **Clothing1M** when $B = 4$. In addition, $B = 1$ and $B = 2,048$ are two extreme conditions,. The former represents that each disjoint subset $\mathcal{S}_m$ contains one large local patch, and therefore it is impossible for each subset to approximate the global data manifold. On the contrary, the latter denotes a strategy similar to random sampling, implying that each subset can coarsely approximate the whole data manifold yet loses the ability to retain the local information. When $B = 2,048$, the proposed ensemble learning strategy has no advantage in coping with noisy labels because data in all disjoint subsets follow the same distribution.

• **Number of Ensemble Branches** M

The number of ensemble branches, M, is also a key hyper-

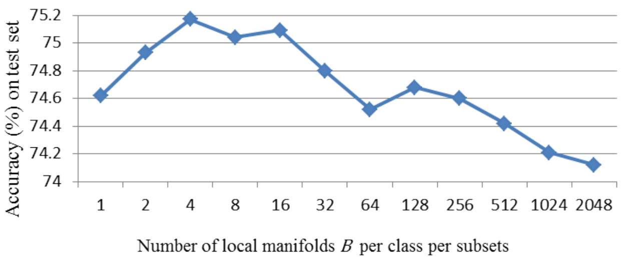


Fig. 6. Test accuracy (%) for different numbers of local manifolds (B) with $M = 5$.

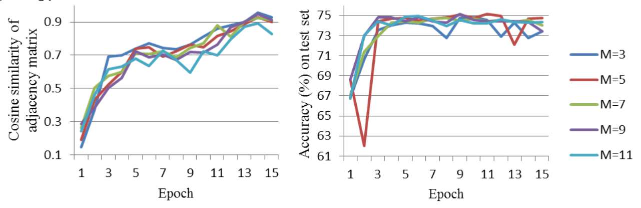


Fig. 7. (a) Average cosine similarity in each adjacency matrix with epoch from 1 to 15 in training process. (b) Test accuracy (%) with epoch from 1 to 15 in training process. The line denote different setting of M and all the model use same $B = 4$.

parameter in our method. To evaluate the impact of M, we conduct experiments by fixing $B = 4$. As shown in Fig. 7(a), the ensemble branches are beneficial for estimating data manifold because the cosine similarities in the adjacency matrices of derived graphs roughly increases with the number of ensemble branches. However, Fig. 7(b) shows that the testing accuracy tends to be saturated when $M \geq 5$. This implies that when $M \geq 5$, some ensemble branches may learn nearly the same coarse approximation of data manifold, and such information might be redundant thus cannot further improve the generalization performance. Therefore, we set $B = 4$ by default.

Moreover, Fig. 7 provides another interpretation. By checking the average cosine similarity values in adjacency matrices derived by different ensemble branches during training, we find that the final ensemble model performs the best when the average cosine similarity is in between 0.7 and 0.8. When the average cosine similarity reaches 0.9, the model becomes overfitted because the testing accuracy of the ensemble model decreases, slightly though.

• **Combinations of** (M, B)

Fixing the number of local patches per class, i.e., $M \cdot B$, we here discuss the influence brought by different combinations of M and B. The experiments demonstrated in Fig. 8 are derived by setting $M \cdot B = 40$ and $M \cdot B = 20$. For $M \times B = 20$, we conduct experiments on $(M, B) = (2, 10)$, $(4, 5)$, $(5, 4)$, and $(10, 2)$. As for $M \times B = 40$, we examine $(M, B) = (2, 20)$, $(4, 10)$, $(5, 4)$, and $(10, 2)$ in turn. Based on these results, we have two observations. First, $(M, B) = (4, 5)$ achieves the best accuracy of 75.15%, whereas $(M, B) = (8, 5)$ reaches about 75.05%. Second, M is far more critical than B, and a suitable M should range from 4 to 10. This is because once a too large M is used, the source noisy data will be divided into too small disjoint subsets, and such small subsets cannot retain the shape of the entire data manifold.

• **Data Re-splitting**

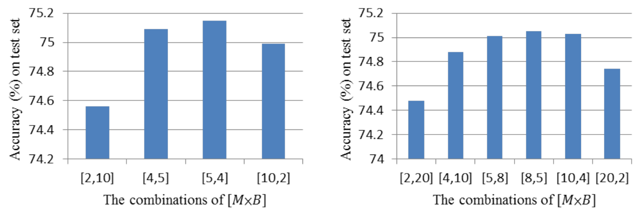


Fig. 8. (a) Test accuracy (%) with different combinations of $[M, B]$ and $MB = 20$. (b) Test accuracy (%) with different combinations of $[M, B]$ and $MB = 40$.

TABLE V
CLASSIFICATION ACCURACY OF DIFFERENT GRAPH SMOOTHNESS LOSS $\mathcal{L}_G$ ON CLOTHING1M.

Method	Accuracy (%)
CrossEntropy	74.08
$\tilde{\mathcal{L}}_G$ + Feature + L_2	74.47
$\tilde{\mathcal{L}}_G$ + Softmax + L_2	75.15
$\mathcal{L}_G$ + Softmax + L_2	**75.17**
$\mathcal{L}_G$ + Softmax + KL	74.92
$\mathcal{L}_G$ + Softmax + JS	74.98

Our experiment shows that the data re-splitting process described in Sec. III-B can on average increase the accuracy by about 0.04%. Specifically, data re-splitting improves the average accuracy from 75.13% to 75.17%. This experiment verifies the idea that re-splitting can assist ensemble branches to learn different combination of local patches and also prevent them from overfitting.

We validate the data re-splitting process in the following two steps. First, we extract the feature vectors of images in the to-be-corrected noisy dataset by using a ResNet-50 pretrained on Clothing1M via cross entropy. Second, with the obtained features, we split the to-be-corrected noisy dataset into M disjoint subsets and then apply our method for label correction without data re-splitting.

• **Weighted Graph Smoothness Loss $\mathcal{L}_G$**

Our $\mathcal{L}_G$ described in (11) is measured based on a weighting factor and the L_2 distance between the probability vectors $\mathbf{p}_s^{\mathcal{I}(s)}$ produced by the softmax layers of individual ensemble branches. This design encourages different sub-networks to produce the same classification results rather than obtaining similar feature vectors. To verify the effectiveness, we run experiments via JS-divergence, KL-divergence, and feature vectors for comparisons, as shown in Table V. First, the experiments show that both $\mathcal{L}_G$ and $\tilde{\mathcal{L}}_G$ (an unweighted version) are more effective than the cross-entropy loss because cross-entropy tends to make all instances in the same class have features of the same kind but $\mathcal{L}_G$ and $\tilde{\mathcal{L}}_G$ do not. Second, by simply using the L_2 distance between probability vectors, i.e., $\tilde{\mathcal{L}}_G$+Softmax+L_2, rather than the L_2 distance between feature vectors, i.e., $\tilde{\mathcal{L}}_G$+Feature+L_2, the accuracy of our model can be boosted from 74.47% to 75.15%. Third, no matter which of L_2 distance, KL-divergence, and JS-divergence is adopted, the softmax output can consistently boost the model accuracy up to 75.17%. This proves the effectiveness of $\mathcal{L}_G$ and the usage of softmax output.

• **Certainty weight**

TABLE VI
NOISE RATE VERSUS WEIGHTING FACTORS FOR ALTERNATIVE LOSS TERMS USED IN OUR ABLATION STUDY.

	CIFAR-10				
Noise Rate	20	50	80	90	asym 40%
$\mathcal{L}_{en\_ce}$	1	1	1	1	1
$\mathcal{L}_{en\_mae}$	0	0	1	1	0
$\mathcal{L}_{cor\_ce}$	1	1	1	1	1
$\mathcal{L}_{cor\_mse}$	0	1	10	10	1

	CIFAR-100				Clothing1M
Noise Rate	20	50	80	90	
$\mathcal{L}_{en\_ce}$	2.5	2.5	2.5	2.5	1
$\mathcal{L}_{en\_mae}$	0	1	1	1	1
$\mathcal{L}_{cor\_ce}$	1	1	1	1	1
$\mathcal{L}_{cor\_mse}$	10	40	80	80	1

Fig. 9 illustrates how certainty weight varies with the progression of training epochs. The certainty weight described in (6) represents the confidence of the correctness of a corrected label. These three figures jointly show that our system design is valid because the number of samples with larger certainty weights increases with the progression of training epochs.

G. Ablation Study

In this section, we examine our system architecture design and examine the effects brought by different combinations of loss terms and system components for all variants of the proposed method. Fig. 10 illustrates the t-SNE maps of three datasets used in this subsection, including Clothing1M, CIFAR-10 with 50% symmetric noise, and CIFAR-10 with 40% symmetric noise. It is obvious that noisy labels of Clothing1M generally locate near class boundaries, and noisy labels of two CIFAR-10 with synthetic label noise almost distributed uniformly in the sample space. Here, we first examine the performance of each variant of our model on Clothing1M and two Person Re-ID datasets, including DukeMTMC [39] and Market-1501 [40], as demonstrated in Table VII.

Since the labels of the two Person-ReID datasets are clean, we conduct our experiments by intentionally injecting 20% confusing noise to the two Person-ReID datasets as follows. i) For each clean ReID dataset, we train a classifier to obtain each image's softmaxed probability vector; ii) we sort the probability vectors of samples by the difference of the top-2 values of their softmaxed probability vector, and pick those samples with the 20% smallest top-2 probability differences, i.e., which are likely to have confusing labels, as candidates; iii) for each candidate, if its ground-truth label is among the two top-2 probability categories, we replace its class with the other of the top-2 categories. Through this way, we can simulate confusing label noise; iv) we apply our label correction method and its variants to correct the noisy labels of the two datasets, and then train a classifier on each corrected training set for performance measurement. Moreover, because the training set and the testing set of each person-ReID dataset contain none-overlapping IDs, we measure the label correction

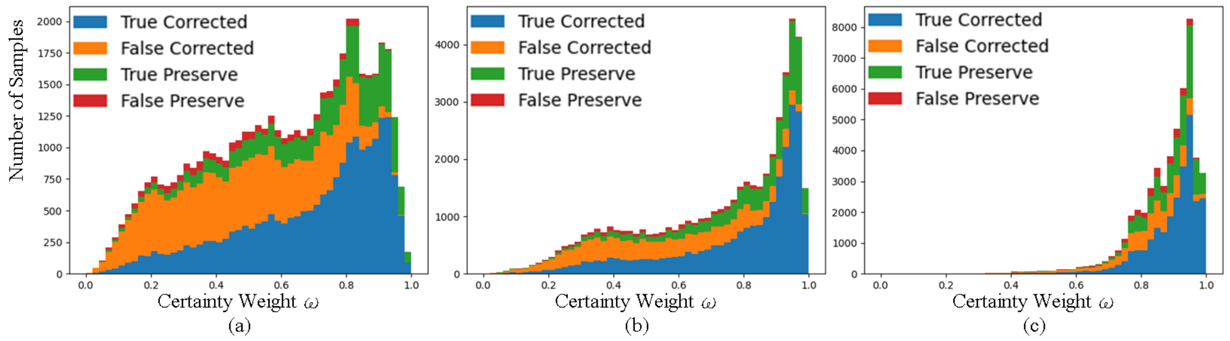


Fig. 9. Distribution of corrected labels and certainty weights on CIFAR-10 with 80% symmetric noise at (a) the 10-th epoch, (b) the 100-th epoch, and (c) the 200-th epoch. Note that the horizontal axis denotes the value of certainty weight, and the vertical axis represents the data amount.

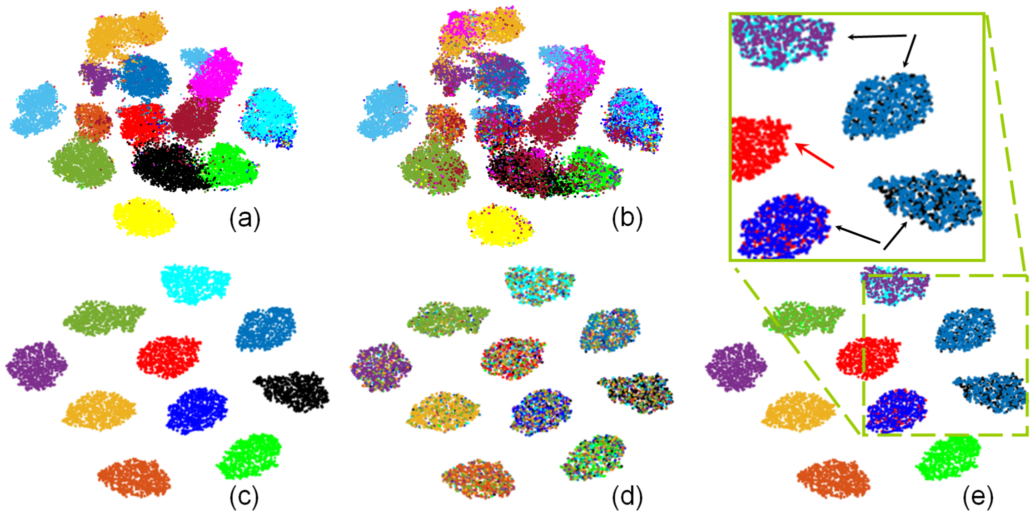


Fig. 10. t-SNE maps of clean datasets and to-be-corrected noisy datasets. (a) clean Clothing1M. (b) Clothing1M with 38.36% noise rate. (c) clean CIFAR-10. (d) CIFAR-10 with 50% symmetric noise. (e) CIFAR-10 with 40% asymmetric noise. Note that, in (e), only five of the ten classes are affected by the asymmetric noise, as indicated by the black arrows, and the remaining classes are noise free, as pointed by the red arrow.

performance using the Top-1 score, rather than the accuracy used for the Clothing1M dataset.

Table VII shows the ablation study of our method compared with its two variants, each lacking either of the two key components (noisy branches and data splitting). It lists the numbers of total injected wrong labels at class boundaries, i.e., 3305 for DukeMTMC and 2588 for Market-1501, and the numbers of samples corrected by our method and its two variants. Our method successfully corrects 2769 out of 3305 wrong labels for DukeMTMC and 1811 out of 2588 wrong labels for Market-1501, clearly validating its efficacy in label denoising. Moreover, our method also significantly outperforms its two variants, demonstrating the necessity of our two key components, noisy branch and data splitting stage, in fighting against real-world confusing label noises in Clothing1M and the simulated confusing label noises in the two person-ReID datasets: DukeMTMC and Market-1501.[1]

Next, we examine the performance of each of our model variants on CIFAR-10 contaminated by synthetic symmetric noises.

[1]The MATLAB code for synthesizing confusing labels is available for download in [41].

This experiment set also verifies i) the necessity of noisy label branch in our method, ii) the impact of Mixup [42] used for data augmentation, and iii) the effects brought by different loss terms used in the ensemble branches. Mixup is a procedure developed for data augmentation for semi-supervised learning, that enables the obtained model to produce linear prediction results for sample pairs. Hence, it can yield more smooth and robust pseudo labels and also avoid the overfitting problems. We summarize the comparison among different loss term designs as follows:

• Ensemble branch

Although the graph smoothness loss described in (11) is beneficial for deriving a better adjacency matrix, this loss may overfit wrongly-labelled samples easily. Hence, in our default setting, we measure this loss only on samples, whose labels remain unchanged after our correction procedure. However, such setting may reduce the effectiveness of this loss. It is because that the amount of uncorrected samples involved in the computation of the graph smoothness loss may become too small when the loss rate is high. As a result, we measure the weighted sum of i) the **MAE** (mean absolute error) $\mathcal{L}_{en\_mae}$ between 1 and the classification probability

TABLE VII
ABLATION STUDY: TEST ACCURACY (%) WITH OUR METHOD AND IT TWO VARIANTS ON (A) CLOTHING1M, (B) DUKEMTMC [39], AND (C) MARKET-1501 [40], WHERE EACH ROW CORRESPONDS TO A DIFFERENT SYSTEM ARCHITECTURE.

	Dataset				
	Clothing1M	DukeMTMC		Market-1501	
	Accuracy (%)	Top-1 score (%)	number of wrong labels	Top-1 score (%)	number of wrong labels
CrossEntropy	69.21	47.54	3305 (20%)	63.42	2588 (20%)
Ours (full-set)	**75.17**	**62.65**	**536 (3.24%)** (2,769 corrected)	**69.18**	**777 (6.00%)** (1,811 corrected)
Ours (w/o noisy branch)	73.91	57.13	1,270 (7.69%) (2,035 corrected)	67.57	1,025 (7.92%) (1,563 corrected)
Ours (w/o data splitting)	73.07	58.30	1,476 (8.93%) (1,829 corrected)	66.24	1,758 (13.59%) (830 corrected)

of corrected samples, and ii) the **cross entropy** $\mathcal{L}_{en\_ce}$ of uncorrected samples as an alternative of the loss term for ensemble branches. We then compare this alternative design with our default setting $\mathcal{L}_G$ described in (11).

• Corrected label branch
Similarly, we adopt the sum of i) **MAE** $\mathcal{L}_{cor\_mae}$ and ii) **cross entropy** $\mathcal{L}_{cor\_ce}$ as an alternative loss design for the corrected label branch. We use the soft-label, which is estimated as a convex combination, weighted by the certainty weight $\bar{\omega}$ depicted in (6), of the prediction result derived by the corrected label branch and that derived the ensemble branches, to compute both $\mathcal{L}_{cor\_mae}$ and the $\mathcal{L}_{cor\_ce}$. A larger $\bar{\omega}$ will produce a soft-label closer to the label suggested by the corrected label branch. In addition, the weighting factors for loss terms used in our ablation study are listed in Table VI.

Tables VIII and IX show that by removing the noisy label branch, changing the loss functions, or using soft-labels, our method can better correcting CIFAR-10/CIFAR-100 datasets with synthetic random label noises. However, such modifications degrade our system performance for real-world applications, e.g. Clothing1M, whose data are contaminated largely by confusing label noises, as shown in Fig. VII. Hence, we finally choose to adopt our default architecture and loss settings because this combination is most beneficial for real-world applications.

We would like to emphasize that it is difficult to combat against different types of label noise using a single approach. Therefore, it is worthy to combine our method into some toolbox with other designs for noisy label correction. As already shown in Tables VIII and IX, our method can deal with synthetic random symmetric/asymmetric noises as well as real-world noisy labels, when collaborating with other strategies such as Mixup and Soft-label.

• Data Splitting Stage
Here we verify the efficacy of the data splitting stage via Tables VII, VIII, and IX. First, these three tables all show that the removal of data splitting stage degrades our model's performance on Clothing1M, CIFAR-10 with random noise, and CIFAR-100 with random noise, evidencing the effectiveness of the data splitting stage. Second, the data splitting stage

TABLE VIII
TEST ACCURACY (%) OF EXPERIMENTS ON CIFAR-10 WITH SYMMETRIC AND ASYMMETRIC NOISES. FOR SYMMETRIC NOISES, THE NOTE RATE WAS SET TO BE 20%, 50%, 80%, AND 90% IN TURN, WHEREAS THE NOISE RATE OF THE EXPERIMENT ON ASYMMETRIC NOISE IS FIXED TO BE 40%. NOTE THAT VALUES IN **LAST** ROWS ARE THE MEAN TEST ACCURACY VALUES DERIVED BY MODELS OF LAST 10 TRAINING EPOCHS.

Noise Level (%)		20	50	80	90	asym
Cross Entropy	Best	86.8	79.4	62.9	42.7	85.0
	Last	82.7	57.9	26.1	16.8	72.3
Mixup [42]	Best	95.6	87.1	71.6	52.2	–
	Last	92.3	77.6	46.7	43.9	–
P-correction [43]	Best	92.4	89.1	77.5	58.9	88.5
	Last	92.0	88.7	76.5	58.2	88.3
Meta-Learning [15]	Best	92.9	89.3	77.4	58.7	89.2
	Last	92.0	88.8	76.1	58.3	88.6
M-correction [21]	Best	94.0	92.0	86.8	69.1	87.4
	Last	93.8	91.9	86.8	68.7	86.3
DivideMix [16]	Best	96.1	94.6	93.2	76.0	93.4
	Last	95.7	94.4	92.9	75.4	92.2
Ours (default)	Best	90.1	81.8	64.4	46.2	88.6
	Last	88.8	81.6	56.1	39.8	88.4
Ours (w/o noisy branch)	Best	93.3	91.1	77.1	52.2	91.2
	Last	93.1	90.8	76.5	45.6	90.9
Ours (w/o data splitting)	Best	88.1	79.1	53.3	37.4	87.4
	Last	83.7	76.6	47.3	33.2	84.9
Ours (w/o data splitting) + Mixup + Soft Label	Best	95.6	93.9	91.0	70.6	87.9
	Last	95.5	93.8	90.8	70.3	87.7
Ours (w/o noisy branch) + Mixup	Best	93.9	91.8	81.3	61.6	92.1
	Last	93.4	91.2	80.4	59.8	91.8
Ours (w/o noisy branch) + Mixup + Soft Label	Best	94.2	93.9	92.5	76.2	92.3
	Last	94.0	93.7	92.3	73.6	92.1

can stabilize our model performance since there is an obvious performance gap between the *best* value and the *last* value (the mean accuracy with a model learned in the last 10 epochs) in each experiment where data splitting is removed, as shown in Tables VIII and IX. Consequently, the data splitting stage is indispensable in our design, especially in real-world noisy label correction applications.

V. CONCLUSION

We proposed an ensemble learning method for noisy label correction by using local patches of feature manifold. Based on the facts that most real-world noisy labels come from confusing label noises, we have proposed a nearest-neighbor-based data splitting scheme to tackle noisy labels locally-

TABLE IX
TEST ACCURACY (%) OF EXPERIMENTS ON CIFAR-100 WITH SYMMETRIC NOISE. NOTE THAT 20%, 50%, 80%, AND 90% DENOTE NOISE RATES; AND, VALUES IN **LAST** ROWS ARE THE MEAN TEST ACCURACY VALUES DERIVED BY MODELS OF LAST 10 TRAINING EPOCHS.

Noise Level (%)		20	50	80	90
Cross Entropy	Best	62.0	46.7	19.9	10.1
	Last	61.8	37.3	8.8	3.5
Mixup [42]	Best	67.8	57.3	30.8	14.6
	Last	66.0	46.6	17.6	8.1
P-correction [43]	Best	69.4	57.5	31.1	15.3
	Last	68.1	56.4	20.7	8.8
Meta-Learning [15]	Best	68.5	59.2	42.4	19.5
	Last	67.7	58.0	40.1	14.3
M-correction [21]	Best	73.9	66.1	48.2	24.3
	Last	73.4	65.4	47.6	20.5
DivideMix [16]	Best	77.3	74.6	60.2	31.5
	Last	76.9	74.2	59.6	31.0
Ours (default)	Best	65.6	50.3	23.1	10.4
	Last	65.0	49.8	21.6	8.2
Ours (w/o noisy branch)	Best	74.1	65.9	40.8	15.2
	Last	73.8	65.5	40.3	13.2
Ours (w/o data splitting)	Best	58.7	48.1	19.5	9.5
	Last	56.8	46.0	17.3	8.5
Ours (w/o data splitting) + Mixup + Soft Label	Best	75.0	71.9	56.3	20.4
	Last	74.8	71.6	55.4	20.2
Ours (w/o noisy branch) + Mixup	Best	76.1	70.2	45.3	21.8
	Last	75.9	69.8	44.6	19.9
Ours (w/o noisy branch) + Mixup + Soft Label	Best	76.6	73.9	57.2	28.6
	Last	76.4	73.1	56.9	28.4

concentrated near decision boundaries by capturing the local structures of data manifolds. Hence, each sub-model can learn a coarse representation of the feature manifold, and only a few sub-models will be affected by locally-dense noisy labels. We have also proposed a multi-graph label propagation algorithm to derive reasonable label correction suggestions, and a confidence-guided label correction mechanism to determine proper label from the correction suggestions via majority decision. Our contribution is not only to design a label correction scheme based on data splitting and ensemble learning, but also to deal with confusing label noise that has not been addressed in previous literature. Extensive experiments and in-depth analyses on real-world datasets evidently demonstrate the superiority of the proposed method over existing noisy label correction strategies.

REFERENCES

[1] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. European Conf. Comput. Vis.*, 2014, pp. 740–755.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*. Master Thesis, Dept. Comput. Sci., Univ. Toronto, 2009.

[6] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1944–1952.

[7] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10 456–10 465.

[8] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama, "Are anchor points really indispensable in label-noise learning?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6838–6849.

[9] K. Sharma, P. Donmez, E. Luo, Y. Liu, and I. Z. Yalniz, "Noiserank: Unsupervised label noise reduction with dependence models," *arXiv preprint arXiv:2003.06729*, 2020.

[10] J. Han, P. Luo, and X. Wang, "Deep self-learning from noisy labels," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5138–5147.

[11] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5552–5560.

[12] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8778–8788.

[13] H. Wei, L. Feng, X. Chen, and B. An, "Combating noisy labels by agreement: A joint training method with co-regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13 726–13 735.

[14] F. Chen, R. Ji, J. Su, D. Cao, and Y. Gao, "Predicting microblog sentiments via weakly supervised multimodal deep learning," *IEEE Trans. Multimedia*, vol. 20, pp. 997–1007, 2017.

[15] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to learn from noisy labeled data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5051–5059.

[16] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," *arXiv preprint arXiv:2002.07394*, 2020.

[17] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox, "Self: Learning to filter noisy labels with self-ensembling," *arXiv preprint arXiv:1910.01842*, 2019.

[18] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[19] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1195–1204.

[20] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2568–2580, 2017.

[21] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 312–321.

[22] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," *arXiv preprint arXiv:1712.09482*, 2017.

[23] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 322–330.

[24] Y. Yan, Z. Xu, I. W. Tsang, G. Long, and Y. Yang, "Robust semi-supervised learning through label aggregation," in *Proc. AAAI Conf. Artificial Intell.*, 2016.

[25] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 321–328.

[26] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5070–5079.

[27] A. Erdem and M. Pelillo, "Graph transduction as a noncooperative game," *Neural Comput.*, vol. 24, no. 3, pp. 700–723, 2012.

[28] M. Douze, A. Szlam, B. Hariharan, and H. Jégou, "Low-shot learning with large-scale diffusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3349–3358.

[29] M. Bontonou, C. Lassance, G. B. Hacene, V. Gripon, J. Tang, and A. Ortega, "Introducing graph smoothness loss for training deep learning architectures," *arXiv preprint arXiv:1905.00301*, 2019.

[30] R. Wang, X.-J. Wu, and J. Kittler, "Graph embedding multi-kernel metric learning for image set classification with grassmannian manifold-valued features," *IEEE Trans. Multimedia*, vol. 23, pp. 228–242, 2020.

[31] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.

[32] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2691–2699.

[33] K.-H. Lee, X. He, L. Zhang, and L. Yang, “Cleannet: Transfer learning for scalable image classifier training with label noise,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5447–5456.

[34] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101–mining discriminative components with random forests,” in *Proc. European Conf. Comput. Vis.* Springer, 2014, pp. 446–461.

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[36] W. Zhang, Y. Wang, and Y. Qiao, “Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7373–7382.

[37] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, “Early-learning regularization prevents memorization of noisy labels,” *arXiv preprint arXiv:2007.00151*, 2020.

[38] Z. Zhang, H. Zhang, S. O. Arik, H. Lee, and T. Pfister, “Distilling effective supervision from severe label noise,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9294–9303.

[39] Z. Zheng, L. Zheng, and Y. Yang, “Unlabeled samples generated by gan improve the person re-identification baseline in vitro,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2017, pp. 3754–3762.

[40] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2015, pp. 1116–1124.

[41] H.-C. Shao, “Confusing label-noise generator,” https://www.mathworks.com/matlabcentral/fileexchange/99714-confusing-label-noise-generator.

[42] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” *arXiv preprint arXiv:1710.09412*, 2017.

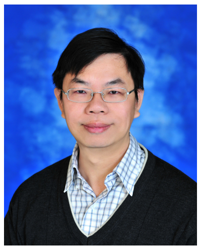
[43] K. Yi and J. Wu, “Probabilistic end-to-end noise correction for learning with noisy labels,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7017–7025.

Hao-Chiang Shao (Member, IEEE) received his Ph.D. degree in electrical engineering from National Tsing Hua University, Taiwan, in 2012. He has been an Assistant Professor with the Dept. Statistics and Information Science, Fu Jen Catholic University, Taiwan, since 2018. During 2012 to 2017, he was a postdoctoral researcher with the Institute of Information Science, Academia Sinica, involved in a series of *Drosophila* brain research projects; in 2017–2018, he was an R&D engineer with the Computational Intelligence Technology Center, Industrial Technology Research Institute, Taiwan, taking charges of DNN-based automated optical inspection (AOI) projects. His research interests include 2D+Z image atlasing, 3D mesh processing, big industrial image data analysis, and machine learning.

Hsin-Chieh Wang received his B.S. degree, in Mechanical and Electromechanical Engineering, National Sun Yat-sen University in 2018 and his M.S. degree, in Electrical Engineering, from National Tsing Hua University in 2020. He has been working for Realtek Semiconductor Corp. as a video codec algorithm engineer since 2021. His research interests lie in computer vision, machine learning, and visual analytics.

Weng-Tai Su (Student Member, IEEE) received the B.S. degree in electrical engineering from the National Yunlin University of Science and Technology, Yunlin, Taiwan, in 2012, Taiwan, the M.S. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2014. He is currently pursuing his Ph.D. degree at the Department of Electrical Engineering of NTHU.

His research interests mainly lie in machine learning, image and video processing, and computer vision.



Chia-Wen Lin (Fellow, IEEE) received the Ph.D. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2000. Dr. Lin is currently Professor with the Department of Electrical Engineering and the Institute of Communications Engineering, NTHU, and R&D Director of the Electronic and Optoelectronic System Research Laboratories, Industrial Technology Research Institute. He is also Deputy Director of NTHU AI Research Center. He was with the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan, during 2000–2007. Prior to joining academia, he worked for the Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan, during 1992–2000. His research interests include image and video processing, computer vision, and video networking.

Dr. Lin served as Distinguished Lecturer of IEEE Circuits and Systems Society from 2018 to 2019, a Steering Committee member of IEEE TRANSACTIONS ON MULTIMEDIA from 2014 to 2015, and the Chair of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and Systems Society from 2013 to 2015. His articles received the Best Paper Award of IEEE VCIP 2015, Top 10% Paper Awards of IEEE MMSP 2013, and the Young Investigator Award of VCIP 2005. He received Outstanding Electrical Professor Award presented by Chinese Institute of Electrical Engineering in 2019, and Young Investigator Award presented by Ministry of Science and Technology, Taiwan, in 2006. He is currently the Chair of the Steering Committee of IEEE ICME. He has been serving as the President of the Chinese Image Processing and Pattern Recognition Association, Taiwan, since 2019. He has served as a Technical Program Co-Chair for IEEE ICME 2010, and a General Co-Chair for IEEE VCIP 2018, and a Technical Program Co-Chair for IEEE ICIP 2019. He has served as an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON MULTIMEDIA, and IEEE MULTIMEDIA.