Learning from Noisy Labels with Deep Neural Networks: A Survey

Hwanjun Song, Minseok Kim, Dongmin Park, Jae-Gil Lee

***Abstract*—Deep learning has achieved remarkable success in numerous domains with help from large amounts of big data. However, the quality of data labels is a concern because of the lack of high-quality labels in many real-world scenarios. As noisy labels severely degrade the generalization performance of deep neural networks, learning from noisy labels (robust training) is becoming an important task in modern deep learning applications. In this survey, we first describe the problem of learning with label noise from a supervised learning perspective. Next, we provide a comprehensive review of 47 state-of-the-art robust training methods, all of which are categorized into seven groups according to their methodological difference, followed by a systematic comparison of six properties used to evaluate their superiority. Subsequently, we summarize the typically used evaluation methodology, including public noisy datasets and evaluation metrics. Finally, we present several promising research directions that can serve as a guideline for future studies.**

***Index Terms*—deep learning, noisy label, label noise, robust optimization, robust deep learning, classification, survey**

I. Introduction

With the recent emergence of large-scale datasets, deep neural networks (DNNs) have exhibited impressive performance in numerous machine learning tasks, such as computer vision [1]–[3], information retrieval [4]–[6], and language processing [7]–[9]. Their success is dependent on the availability of massive but carefully labeled data, which are expensive and time-consuming to obtain. Some non-expert sources, such as Amazon's Mechanical Turk and the surrounding tags of collected data, have been widely used to mitigate the high labeling cost; however, the use of these source often results in unreliable labels [10]–[12]. In addition, data labels can be extremely complex even for an inexperienced person [13]; they can also be adversarially manipulated by a label-flipping attack [14]. Such unreliable labels are called *noisy labels* because they may be *corrupted* from ground-truth labels. The ratio of corrupted labels in real-world datasets is reported to range from 8.0% to 38.5% [15]–[18].

In the presence of noisy labels, training DNNs is known to be susceptible to noisy labels because of the significant number of model parameters that render DNNs overfit to even corrupted labels with the capability of learning any complex function [19]. Zhang et al. [20] demonstrated that DNNs can easily fit an entire training dataset with any ratio of corrupted

H. Song, M. Kim, D, Park, and J.-G., Lee are with the Graduate School of Knowledge Service Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea (e-mail: songhwanjun@kaist.ac.kr; minseokkim@kaist.ac.kr; dongminpark@kaist.ac.kr; jaegil@kaist.ac.kr).

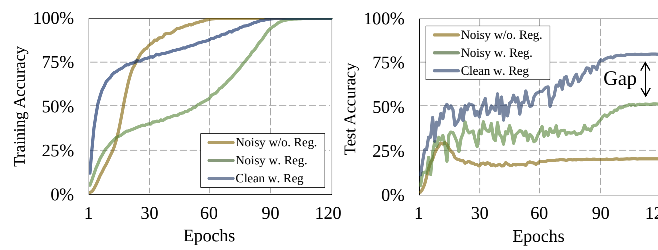


Fig. 1. Convergence curves of training and test accuracy when training WideResNet-16-8 using a standard training method on the CIFAR-100 dataset with the symmetric noise of 40%: "Noisy w/o. Reg." and "Noisy w. Reg." are the models trained on noisy data without and with regularization, respectively, and "Clean w. Reg." is the model trained on clean data with regularization.

labels, which eventually resulted in poor generalizability on a test dataset. Unfortunately, popular regularization techniques, such as data augmentation [21], weight decay [22], dropout [23], and batch normalization [24], do not completely overcome the overfitting issue. As shown in Figure 1, the gap in test accuracy between models trained on clean and noisy data remains significant even though all of the aforementioned regularization techniques are activated. Additionally, the accuracy drop with label noise is considered to be more harmful than with other noises, such as feature noise [25]. Hence, achieving a good generalization capability in the presence of noisy labels is a key challenge.

Several studies have been conducted to investigate supervised learning under noisy labels. Beyond conventional machine learning techniques [13], [26], deep learning techniques have recently gained significant attention in the machine learning community. In this survey, we present the advances in recent deep learning techniques for overcoming noisy labels. We surveyed *113* recent studies by recursively tracking relevant bibliographies in papers published at premier research conferences, such as CVPR, ICCV, NeurIPS, ICML, and ICLR. Although we attempted to comprehensively include all recent studies at the time of submission, some of them may not be included because of the quadratic increase in deep learning papers. The studies included were grouped into *seven* categories, as shown in Figure 2 (see Section III for details).

A. Related Surveys

Frénay and Verleysen [13] discussed the potential negative consequence of learning from noisy labels and provided a comprehensive survey on noise-robust classification methods, focusing on conventional supervised approaches such as naïve Bayes and support vector machines. Furthermore, their survey included the definitions and sources of label noise as well as the taxonomy of label noise. Zhang et al. [26] discussed

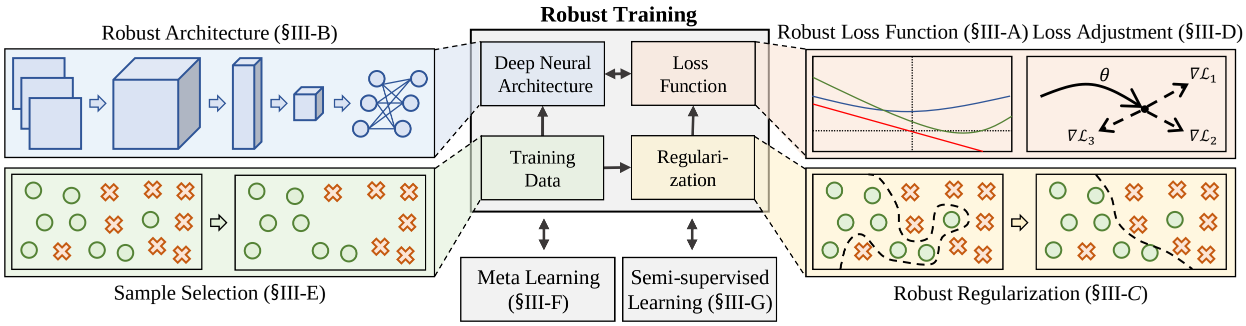


Fig. 2. Categorization of recent deep learning methods for overcomming noisy labels.

TABLE I
SUMMARY OF THE NOTATION.

Notation	Description
$\mathcal{X}$	the data feature space
$\mathcal{Y}$, $\tilde{\mathcal{Y}}$	the true and noisy label space
$\mathcal{D}$, $\tilde{\mathcal{D}}$	the clean and noisy training data
$P_{\mathcal{D}}$, $P_{\tilde{\mathcal{D}}}$	the joint distributions of clean and noisy data
$\mathcal{B}_t$	a set of mini-batch samples at time t
Θ_t	the parameter of a deep neural network at time t
$f(\,\cdot\,;\Theta_t)$	a deep neural network parameterized by Θ_t
$\mathcal{L}$	a specific loss function
$\mathcal{R}$	an empirical risk
$\mathbb{E}_{\mathcal{D}}$	an expectation over $\mathcal{D}$
x, x_i	a data sample of $\mathcal{X}$
y, y_i	a true label of $\mathcal{Y}$
$\tilde{y}$, $\tilde{y}_i$	a noisy label of $\tilde{\mathcal{Y}}$
η	a specific learning rate
τ	a true noise rate
b	the number of mini-batch samples in $\mathcal{B}_t$
c	the number of classes
T, $\hat{\mathrm{T}}$	the true and estimated noise transition matrix

another aspect of label noise in crowdsourced data annotated by non-experts and provided a thorough review of expectation-maximization (EM) algorithms that were proposed to improve the quality of crowdsourced labels. Recently, Nigam et al. [27] provided a brief introduction to deep learning algorithms that were proposed to manage noisy labels; however, the scope of these algorithms was limited to only two categories, i.e., the loss function and sample selection in Figure 2.

II. PRELIMINARIES

In this section, the problem statement for supervised learning with noisy labels is provided along with the taxonomy of label noise. Managing noisy labels is a long-standing issue; therefore, we review the basic conventional approaches as well. Table I summarizes the notation frequently used in this study.

A. Supervised Learning with Noisy Labels

Classification is a representative supervised learning task for learning a function that maps an input feature to a label [28]. In this paper, we consider a c-class classification problem using a DNN with a softmax output layer. Let $\mathcal{X} \subset \mathbb{R}^d$ be the feature space and $\mathcal{Y} = \{0,1\}^c$ be the ground-truth label space in a *one-hot* manner. In a typical classification problem, we are provided with a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ obtained from an unknown joint distribution $P_{\mathcal{D}}$ over $\mathcal{X}\times\mathcal{Y}$, where each (x_i, y_i) is *independent and identically distributed*. The goal of the task is to learn the mapping function $f(\,\cdot\,;\Theta) : \mathcal{X} \rightarrow [0,1]^c$ of the DNN parameterized by Θ such that the parameter Θ minimizes the empirical risk $\mathcal{R}_{\mathcal{L}}(f)$,

$$\mathcal{R}_{\mathcal{L}}(f) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}\big(f(x;\Theta), y\big)] = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} \mathcal{L}\big(f(x;\Theta), y\big), \quad (1)$$

where $\mathcal{L}$ is a certain loss function.

As data labels are corrupted in various real-world scenarios, we aim to train the DNN from noisy labels. Specifically, we are provided with a noisy training dataset $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N$ obtained from a noisy joint distribution $P_{\tilde{\mathcal{D}}}$ over $\mathcal{X}\times\tilde{\mathcal{Y}}$, where $\tilde{y}$ is a *noisy* label which may not be true. Hence, following the standard training procedure, a mini-batch $\mathcal{B}_t = \{(x_i, \tilde{y}_i)\}_{i=1}^b$ comprising b samples is obtained randomly from the noisy training dataset $\tilde{\mathcal{D}}$ at time t. Subsequently, the DNN parameter Θ_t at time t is updated along the descent direction of the empirical risk on mini-batch $\mathcal{B}_t$,

$$\Theta_{t+1} = \Theta_t - \eta\nabla\Big(\frac{1}{|\mathcal{B}_t|}\sum_{(x,\tilde{y})\in\mathcal{B}_t} \mathcal{L}\big(f(x;\Theta_t), \tilde{y}\big)\Big), \quad (2)$$

where η is a learning rate specified.

Here, the risk minimization process is no longer *noise-tolerant* because of the loss computed by the noisy labels. DNNs can easily memorize corrupted labels and correspondingly degenerate their generalizations on unseen data [13], [26], [27]. Hence, mitigating the adverse effects of noisy labels is essential to enable noise-tolerant training for deep learning.

B. Taxonomy of Label Noise

Even if data labels are corrupted from ground-truth labels without *any* prior assumption, in essence, the corruption probability is affected by the dependency between *data features* or *class labels*. A detailed analysis of the taxonomy of label noise was provided by Frénay and Verleysen [13].

A typical approach for modeling label noise assumes that the corruption process is conditionally *independent* of data features when the true label is given [20], [29]. That is, the true label is corrupted by a *label transition matrix* T, where $\mathrm{T}_{ij} := p(\tilde{y} = j|y = i)$ is the probability of the true label i being flipped into a corrupted label j. In this approach, the noise is called a *symmetric* (or *uniform*) noise with a noise rate $\tau \in [0,1]$ if $\forall_{i=j}\mathrm{T}_{ij} = 1 - \tau \wedge \forall_{i\neq j}\mathrm{T}_{ij} = \frac{\tau}{c-1}$, where a true label is flipped into other labels with equal probability. In contrast to symmetric noise, the noise is called

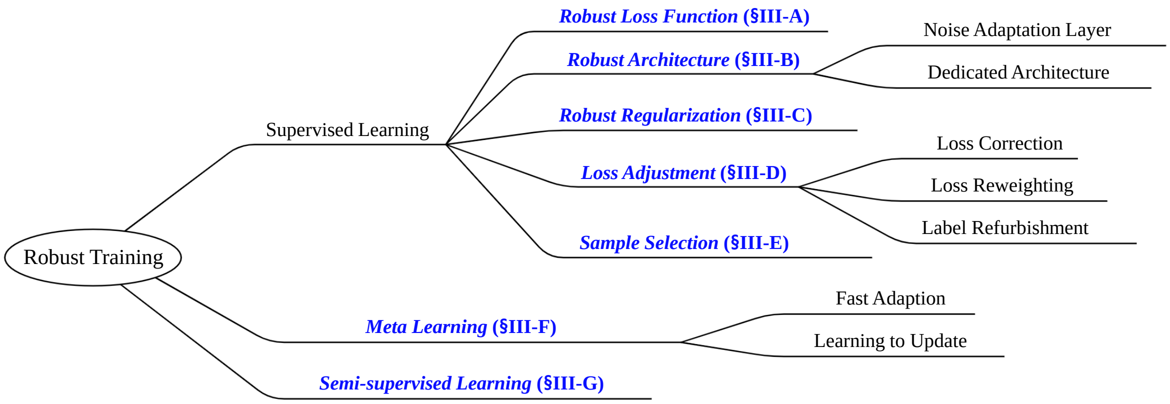


Fig. 3. A high level research overview of robust deep learning for noisy labels. The research directions that are actively contributed by the machine learning community are categorized into seven groups in blue italic.

an *asymmetric* (or *label-dependent*) noise if $\forall_{i=j} \mathrm{T}_{ij} = 1 - \tau \wedge \exists_{i \neq j, i \neq k, j \neq k} \mathrm{T}_{ij} > \mathrm{T}_{ik}$, where a true label is more likely to be mislabeled into a particular label. For example, a "dog" is more likely to be confused with a "cat" than with a "fish." In a stricter case when $\forall_{i=j} \mathrm{T}_{ij} = 1 - \tau \wedge \exists_{i \neq j} \mathrm{T}_{ij} = \tau$, the noise is called a *pair noise*, where a true label is flipped into only a certain label. However, this family of label noises is not realistic because wrong annotations are made regardless of data features.

For more realistic noise modeling, the corruption probability is assumed to be *dependent* on both the data features and class labels [15], [30]. Accordingly, the corruption probability is defined as $\rho_{ij}(x) = p(\tilde{y} = j | y = i, x)$. Unlike the aforementioned noises, because the data feature of a sample x also affects the chance of x being mislabeled, the noise is called an *instance-* and *label-dependent* noise. However, the modeling of this noise has not been investigated extensively yet owing to its complexity.

C. Non-deep Learning Approaches

For decades, numerous methods have been proposed to manage noisy labels using conventional machine learning techniques. These methods can be categorized into *four* groups [13], [27], as follows:

- **Data Cleaning:** Training data are cleaned by excluding samples whose labels are likely to be corrupted. Bagging and boosting are used to filter out false-labeled samples to remove samples with higher weights because false-labeled samples tend to exhibit much higher weights than true-labeled samples [31], [32]. In addition, various methods, such as k-nearest neighbor, outlier detection, and anomaly detection, have been widely exploited to exclude false-labeled samples from noisy training data [33]–[35]. Nevertheless, this family of methods suffers from over-cleaning issue that overly removes even the true-labeled samples.
- **Surrogate Loss:** Motivated by the noise-tolerance of the 0-1 loss function [29], many researchers have attempted to resolve its inherent limitations, such as computational hardness and non-convexity that render gradient methods unusable. Hence, several convex surrogate loss functions, which approximate the 0-1 loss function, have been proposed to train a specified classifier under the binary classification setting [36]–[40]. However, these loss functions cannot support the multi-class classification task.
- **Probabilistic Method:** Under the assumption that the distribution of features is helpful in solving the problem of learning from noisy labels [41], the confidence of each label is estimated by clustering and then used for a weighted training scheme [42]. This confidence is also used to convert hard labels into soft labels to reflect the uncertainty of labels [43]. In addition to these clustering approaches, several Bayesian methods have been proposed for graphical models such that they can benefit from using any type of prior information in the learning process [44]. However, this family of methods may exacerbate the overfitting issue owing to the increased number of model parameters.
- **Model-based Method:** As conventional models, such as the SVM and decision tree, are not robust to noisy labels, significant effort has been expended to improve the robustness of these models. To develop a robust SVM model, misclassified samples during learning are penalized in the objective [45], [46]. In addition, several decision tree models are extended using new split criteria to solve the overfitting issue when the training data are not fully reliable [47], [48]. However, it is infeasible to apply the design principles in these models to deep learning.

III. Deep Learning Approaches

According to our comprehensive survey, the robustness of deep learning can be enhanced in numerous approaches [15], [23], [49]–[53]. Figure 3 shows an overview of recent research directions conducted by the machine learning community. Most of them (i.e., §III-A–§III-E) focused on making a supervised learning process more robust to label noise. *Robust loss function* and *loss adjustment* aim to modify the loss function or its value; *robust architecture* aims to change an architecture to model a noise transition matrix of a noisy dataset; *robust regularization* aims to enforce a DNN to overfit less to false-labeled samples; *sample selection* aims to identify true-labeled samples from noisy training data. Beyond supervised learning, researchers have recently attempted to further improve noise robustness by adopting *meta learning* (§III-F) and *semi-supervised learning* (§III-G). In this paper, we categorize all

TABLE II
SUMMARY OF EXISTING DEEP LEARNING METHODS ACCORDING TO THE SEVEN CATEGORIES IN FIGURE 2.

Category	Deep Learning Method
Robust Loss Function	*Robust MAE* [49], *Generalized Cross Entropy* [54], *Symmetric Cross Entropy* [55], *Curriculum Loss* [56]
Robust Architecture	*Webly Learning* [57], *Noise Model* [58], *Dropout Noise Model* [59], *S–model* [60], *C–model* [60], *NLNN* [61], *Probabilistic Noise Model* [15], *Masking* [62], *Contrastive-Additive Noise Network* [63]
Robust Regularization	*Adversarial Training* [64], *Label Smoothing* [65], *Mixup* [66], *Bilevel Learning* [67], *Annotator Confusion* [68], *Pre-training* [69]
Loss Adjustment	*Backward Correction* [70], *Forward Correction* [70], *Gold Loss Correction* [71], *Importance Reweighting* [72], *Active Bias* [73], *Bootstrapping* [74], *Dynamic Bootstrapping* [75], *D2L* [76], *SELFIE* [18]
Sample Selection	*Decouple* [51], *MentorNet* [77], *Co-teaching* [78], *Co-teaching+* [79], *Iterative Detection* [80], *ITLM* [81], *INCV* [82], *SELFIE* [18], *SELF* [83], *Curriculum Loss* [56]
Meta Learning	*Meta-Regressor* [52], *Knowledge Distillation* [84], *L2LWS* [85], *CWS* [86], *Automatic Reweighting* [87], *MLNT* [88], *Meta-Weight-Net* [89], *Data Coefficients* [101]
Semi-supervised Learning	*Label Aggregation* [53], *Two-Stage Framework* [90], *SELF* [83], *DivideMix* [91]

recent deep learning methods into *seven* groups corresponding to popular research directions, as shown in Figure 3.

Figure 2 illustrates the categorization of robust training methods using these seven groups. Table II summarizes existing deep learning methods according to them. Some methods may belong to more than one categories if they combine multiple approaches.

A. Robust Loss Function

Considering the robustness of risk minimization schemes on the loss function, researchers have attempted to design robust loss functions [49], [54]–[56]. The goal is to provide a loss function that achieves a small risk for unseen clean data even when noisy labels exist in the training data.

Initially, Manwani and Sastry [37] theoretically proved a sufficient condition for the loss function such that risk minimization with that function becomes noise-tolerant for binary classification. Subsequently, the sufficient condition was extended for multi-class classification using deep learning [49]. Specifically, a loss function is defined to be *noise-tolerant* for a c-class classification under *symmetric* noise if the function satisfies the noise rate $\tau < \frac{c-1}{c}$ and

$$\sum_{j=1}^{c} \mathcal{L}\big(f(x;\Theta), y=j\big) = C,\ \forall x \in \mathcal{X},\ \forall f, \tag{3}$$

where C is a constant. This condition guarantees that the classifier trained on noisy data has the same misclassification probability as that trained on noise-free data under the specified assumption. Moreover, if $\mathcal{R}_{\mathcal{L}}(f^*) = 0$, then the function is also noise-tolerant under an *asymmetric* noise, where f^* is a global risk minimizer of $\mathcal{R}_{\mathcal{L}}$.

For the classification task, the categorical cross entropy (CCE) loss is the most widely used loss function owing to its fast convergence and high generalization capability. However, in the presence of noisy labels, the *robust MAE* [49] showed that the mean absolute error (MAE) loss achieves better generalization than the CCE loss because only the MAE loss satisfies the aforementioned condition. A limitation of the MAE loss is that its generalization performance degrades significantly when complicated data are involved. Hence, the *generalized cross entropy* (GCE) [54] was proposed to achieve the advantages of both MAE and CCE losses; the GCE loss is a more general class of noise-robust loss that encompasses both of them. Inspired by the symmetricity of the Kullback-Leibler divergence, the symmetric cross entropy (SCE) [55] was proposed by combining a noise tolerance term, namely reverse cross entropy loss, with the standard CCE loss. Meanwhile, the *curriculum loss* (CL) [56] is a surrogate loss of the 0-1 loss function; it provides a tight upper bound and can easily be extended to multi-class classification.

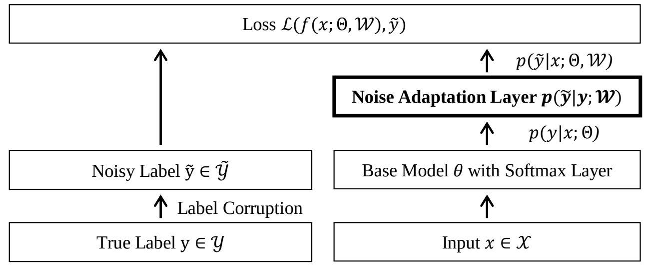


Fig. 4. Noise modeling process using the noise adaptation layer.

Nevertheless, it has been reported that performances with such losses are significantly affected by noisy labels [87]. Such implementations perform well only in simple cases, when learning is easy or the number of classes is small. Moreover, the modification of the loss function increases the training time for convergence [54].

B. Robust Architecture

In numerous studies, architectural changes have been made to model the label transition matrix of a noisy dataset [15], [57]–[63]. These changes include adding a noise adaptation layer at the top of the softmax layer and designing a new dedicated architecture. The resulting architectures yielded improved generalization through the modification of the DNN output based on the estimated label transition probability.

*1) **Noise Adaptation Layer**:* The noise adaptation layer is intended to mimic the noise behavior in learning a DNN. Let $p(y|x;\Theta)$ be the output of the base DNN with a softmax output layer. Subsequently, the probability of a sample x being predicted as its annotated noisy label $\tilde{y}$ is parameterized by

$$\begin{aligned} p(\tilde{y}|x;\Theta,\mathcal{W}) &= \sum_{i=1}^{c} p(\tilde{y}, y\!=\!i|x;\Theta,\mathcal{W}) \\ &= \sum_{i=1}^{c} \underbrace{p(\tilde{y}|y\!=\!i;\mathcal{W})}_{\text{noise adaptation layer}} \underbrace{p(y\!=\!i|x;\Theta)}_{\text{base model}}. \end{aligned} \tag{4}$$

Here, the noisy label $\tilde{y}$ is assumed to be conditionally independent of the input feature x in general. Accordingly, as shown in Figure 4, the noisy adaptation layer is added at the top of the base DNN to model the label transition matrix parameterized by $\mathcal{W}$. This layer should be removed when a test dataset is to be predicted.

Webly learning [57] first trains the base DNN only for easy samples retrieved by search engines; subsequently, the confusion matrix for all training samples is used as the initial weight $\mathcal{W}$ of the noise adaptation layer. It fine-tunes the entire model in an end-to-end manner for hard training samples. Meanwhile, the *noise model* [58] initializes $\mathcal{W}$ to an identity matrix and adds a regularizer to force $\mathcal{W}$ to diffuse during DNN training. The *dropout noise model* [23] applies dropout regularization to the adaptation layer, whose output is normalized by the softmax function to implicitly diffuse $\mathcal{W}$. The *s-model* [60] is similar to the *dropout noise model* but dropout is not applied. The *c-model* [60] is an extension of the s-model that models the instance- and label-dependent noise $p(\tilde{y}|y, x; \mathcal{W})$, which is more realistic than the symmetric and asymmetric noises $p(\tilde{y}|y; \mathcal{W})$. Meanwhile, *NLNN* [61] adopts the EM algorithm to iterate the E-step to estimate the label transition matrix and the M-step to back-propagate the DNN.

The drawback of this family is the strong assumption regarding the noise type, which hinders a model's generalization to complex label noise [15]. Meanwhile, for the EM-based method, becoming stuck in local optima is inevitable, and high computational costs are incurred [60].

*2) **Dedicated Architecture**:* To overcome the aforementioned drawbacks of the noise adaptation layer, several studies have been conducted, where specific architectures have been designed [15], [62], [63]. They typically aimed at increasing the reliability of estimating the label transition probability to handle more complex and realistic label noise. In particular, *probabilistic noise modeling* [15] manages two independent networks, each of which is specialized to predict the noise type and label transition probability. Because an EM-based approach with random initialization is impractical for training the entire network, both networks are trained with massive noisy labeled data after the pre-training step with a small amount of clean data. Meanwhile, *masking* [62] is a human-assisted approach to convey the human cognition of invalid label transitions. Considering that noisy labels are mainly from the interaction between humans and tasks, the invalid transition investigated by humans was leveraged to constrain the noise modeling process. Owing to the difficulty in specifying the explicit constraint, a variant of generative adversarial networks (GANs) [92] was employed in this study. Most recently, the *contrastive-additive noise network* [63] was proposed to adjust incorrectly estimated label transition probabilities by introducing a new concept of quality embedding, which models the trustworthiness of noisy labels.

Compared with the noise adaptation layer, this family of methods significantly improves the robustness to more diverse types of label noise, but it cannot be easily extended to other architectures.

C. Robust Regularization

Regularization methods have been widely studied to improve the generalizability of a learned model in the machine learning community [21]–[24]. By avoiding overfitting in model training, the robustness to label noise improves with widely-used regularization techniques such as *data augmentation* [21], *weight decay* [22], *dropout* [23], and *batch normalization* [24]. Additionally, *adversarial training* [64] enhances the noise tolerance by encouraging the DNN to correctly classify both original inputs and hostilely perturbed ones. *Label smoothing* [65] estimates the marginalized effect of label noise during training, thereby reducing overfitting by preventing the DNN from assigning a full probability to noisy training samples. These methods operate well on moderately noisy data. However, they are generic regularization techniques that are not specialized in handling label noise; hence, poor generalization could be obtained when the noise is heavy [68].

Recently, as learning from noisy labels has become a key challenge, more advanced regularization techniques have been proposed, which further improved robustness to label noise. In particular, *mixup* [66] regularizes the DNN to favor simple linear behaviors in between training samples. First, the mini-batch is constructed using virtual training samples, each of which is formed by the linear interpolation of two noisy training samples $(x_i, \tilde{y}_i)$ and $(x_j, \tilde{y}_j)$ obtained at random from noisy training data $\tilde{\mathcal{D}}$,

$$x_{mix} = \lambda x_i + (1 - \lambda) x_j \ \text{ and } \ y_{mix} = \lambda \tilde{y}_i + (1 - \lambda) \tilde{y}_j, \quad (5)$$

where $\lambda \sim Beta(\alpha, \alpha)$ and $\alpha \in [0, \infty]$. *Mixup* extends the training distribution by updating the DNN for the constructed mini-batch.

Bilevel learning [67] uses a clean validation dataset to regularize the overfitting of a model by introducing a bilevel optimization approach, which differs from the conventional one in that its regularization constraint is also an optimization problem. Overfitting is controlled by adjusting the weights on each mini-batch and selecting their values such that they minimize the error on the validation dataset. Meanwhile, *annotator confusion* [68] assumes the existence of multiple annotators and introduces a regularized EM-based approach to model the label transition probability; its regularizer enables the estimated transition probability to converge to the true confusion matrix of the annotators. In addition, *pre-training* [69] empirically proves that fine-tuning on a pre-trained model provides a significant improvement in robustness compared with models trained from scratch. The universal representations of pre-training prevent the model parameters from being updated in the wrong direction by noisy labels.

The main advantage of this family of methods is its *flexibility* in collaborating with other directions because it only requires simple modifications during training. However, the performance improvement is relatively insignificant, and it tends to yield additional hyperparameters sensitive to both noise and data types.

D. Loss Adjustment

Loss adjustment is effective for reducing the negative impact of noisy labels by adjusting the loss of all training samples before updating the DNN [18], [70]–[76], as shown in Figure 5(a). The methods associated with it can be categorized into three groups depending on their adjustment philosophy: *1) loss*

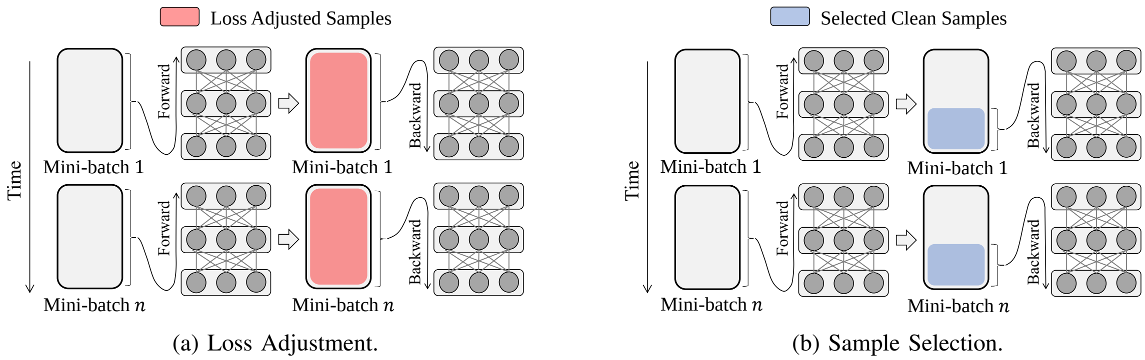


Fig. 5. Comparison of two different training procedures: (a) shows the training procedures of *loss adjustment*; (b) shows the training procedures of *sample selection*. (This figure is adapted from Song et al. [18].)

correction that estimates the label transition matrix to correct the forward or backward loss, *2) loss reweighting* that imposes different importance to each sample for a weighted training scheme, and *3) label refurbishment* that adjusts the loss using the refurbished label obtained from a convex combination of noisy and predicted labels.

Generally, these approaches allow for a *full exploration* of the training data while adjusting the loss of every sample. However, the error incurred by *false* correction is accumulated, especially when the number of classes or the number of mislabeled samples is large [78], [93].

*1) **Loss Correction**:* Similar to the noise adaptation layer presented in Section III-B, this approach modifies the loss of each sample by multiplying the estimated label transition probability by the output of a specified DNN. The main difference is that the learning of the transition probability is decoupled from that of the model. Hence, *backward correction* [70] initially approximates the label transition matrix using the softmax output of the DNN trained without loss correction. Subsequently, it retrains the DNN while correcting the original loss based on the estimated matrix. The corrected loss of a sample $(x, \tilde{y})$ is computed by a linear combination of its loss values for observable labels, whose coefficient is the transition probability from each observable label $j \in \{1, \ldots, c\}$ to its target label $\tilde{y}$. Therefore, the backward correction $\overleftarrow{\mathcal{L}}$ is performed by multiplying the estimated transition probability with its corresponding loss value,

$$\begin{aligned}\overleftarrow{\mathcal{L}}\big(f(x;\Theta), \tilde{y}\big) &= \sum_{j=1}^{c} \hat{p}(\tilde{y}|y=j)\mathcal{L}\big(f(x;\Theta), y=j\big)\\ &= \hat{\mathbf{T}}_{\cdot j}^{-1}\Big(\mathcal{L}\big(f(x;\Theta), y=1\big), \ldots, \mathcal{L}\big(f(x;\Theta), y=c\big)\Big)^{\top},\end{aligned} \quad (6)$$

where $\hat{\mathbf{T}}$ is the estimated label transition matrix.

Conversely, *forward correction* [70] uses a linear combination of a DNN's softmax outputs before applying the loss function. Hence, the forward correction $\vec{\mathcal{L}}$ is performed by multiplying the estimated transition probability with the softmax outputs during the forward propagation step,

$$\begin{aligned}\vec{\mathcal{L}}\big(f(x;\Theta), \tilde{y}\big) &= \mathcal{L}\Big(\big(\hat{p}(\tilde{y}|y=1), \ldots, \hat{p}(\tilde{y}|y=c)\big) f(x;\Theta)^{\top}, \tilde{y}\Big)\\ &= \mathcal{L}\big(\hat{\mathbf{T}}^{-1} f(x;\Theta)^{\top}, \tilde{y}\big).\end{aligned} \quad (7)$$

Furthermore, *gold loss correction* [71] was proposed to leverage available trusted labels for loss correction. To obtain a more accurate transition matrix, the confusion matrix for the trusted labels is utilized as additional information. Owing to the trusted labels, the noise robustness of the forward or backward correction method is further improved.

*2) **Loss Reweighting**:* Inspired by the concept of importance reweighting [94], loss reweighting aims to assign smaller weights to the samples with false labels and greater weights to those with true labels. Accordingly, the reweighted loss on the mini-batch $\mathcal{B}_t$ is used to update the DNN,

$$\Theta_{t+1} = \Theta_t - \eta\nabla\Big(\frac{1}{|\mathcal{B}_t|}\sum_{(x,\tilde{y})\in\mathcal{B}_t} \overbrace{w(x,\tilde{y})\mathcal{L}\big(f(x;\Theta_t), \tilde{y}\big)}^{\text{reweighted loss}}\Big), \quad (8)$$

where $w(x, \tilde{y})$ is the weight of a sample x with its noisy label $\tilde{y}$. Hence, the samples with smaller weights do not significantly affect the DNN learning.

In *importance reweighting* [72], the ratio of two joint data distributions $w(x, y) = P_{\mathcal{D}}(x, \tilde{y})/P_{\tilde{\mathcal{D}}}(x, \tilde{y})$ determines the contribution of the loss of each noisy sample. An approximate solution to estimate the ratio was developed because the two distributions are difficult to determine from noisy data. Meanwhile, *active bias* [73] emphasizes uncertain samples with inconsistent label predictions by assigning their prediction variances as the weights for training.

*3) **Label Refurbishment**:* Refurbishing a noisy label $\tilde{y}$ effectively prevents overfitting to false labels. Let $\hat{y}$ be the current prediction of DNN $f(x; \Theta)$. Therefore, the refurbished label y^{refurb} can be obtained by a convex combination of the noisy label $\tilde{y}$ and the DNN prediction $\hat{y}$,

$$y^{refurb} = \alpha\tilde{y} + (1 - \alpha)\hat{y}, \quad (9)$$

where $\alpha \in [0, 1]$ is the label confidence of $\tilde{y}$. To mitigate the damage of incorrect labeling, this approach backpropagates the loss for the refurbished label instead of the noisy one, thereby yielding substantial robustness to noisy labels.

Bootstrapping [74] is the first method that proposes the concept of label refurbishment to update the target label of training samples. It develops a more coherent network that improves its ability to evaluate the consistency of noisy labels, with the label confidence α obtained via cross-validation. *Dynamic bootstrapping* [75] dynamically adjusts the confidence α of individual training samples. For each training epoch, it estimates the probability of a sample x being true-labeled by fitting a two-component and one-dimensional beta mixture model to the loss distribution of all training samples and then uses it as the confidence α. As true-labeled samples exhibit smaller losses than false-labeled ones, the confidence α of a sample x is obtained through the posterior probability $p(g|\mathcal{L}(f(x;\Theta_t),\tilde{y}))$ of the mixture model, where g is the beta component with a smaller mean.

D2L [76] trains a DNN using a dimensionality-driven learning strategy to avoid overfitting to false labels. A simple measure called *local intrinsic dimensionality* [95] is adopted to evaluate the confidence α in considering that the overfitting is exacerbated by dimensional expansion. Hence, refurbished labels are generated to prevent the dimensionality of the representation subspace from expanding at a later stage of training. Most recently, *SELFIE* [18] introduces a novel concept of *refurbishable samples* that can be corrected with high precision. The key idea is to consider the sample with consistent label predictions as refurbishable because such consistent predictions correspond to its true label with a high probability owing to the learner's perceptual consistency. Accordingly, the labels of only refurbishable samples are corrected to minimize the number of falsely corrected cases.

E. Sample Selection

To avoid any false corrections, many recent studies [18], [51], [56], [77]–[83] have adopted sample selection that involves selecting true-labeled samples from a noisy training dataset, as shown in Figure 5(b). In this case, the update equation in Eq. (2) is modified to render a DNN more robust for noisy labels. Let $\mathcal{C}_t \subseteq \mathcal{B}_t$ be the selected *clean* samples at time t. Therefore, the DNN is updated only for the selected clean samples $\mathcal{C}_t$,

$$\Theta_{t+1} = \Theta_t - \eta\nabla\Big(\frac{1}{|\mathcal{C}_t|}\sum_{(x,\tilde{y})\in\mathcal{C}_t}\mathcal{L}\big(f(x;\Theta_t),\tilde{y}\big)\Big). \tag{10}$$

Here, the remaining mini-batch samples, which are likely to be false-labeled, are excluded from the update to pursue robust learning. The key challenge is to design the *selection criteria* for sample selection.

Initially, *decouple* [51] proposes the decoupling of when to update from how to update. It updates the model for samples selected based on a disagreement between the two classifiers. Hence, two DNNs are maintained simultaneously and updated only using samples with different label predictions from these two DNNs. Subsequently, many researchers have adopted another selection criterion, called a *small-loss* trick, which treats a certain number of small-loss training samples as true-labeled samples. In particular, the small-loss trick successfully separates true-labeled samples from false-labeled samples because many true-labeled samples tend to exhibit smaller losses than false-labeled samples. This phenomenon is well justified by the *memorization effect* [19], i.e., DNNs are prone to learn clean samples first and then gradually learn other noisy samples.

MentorNet [77] introduces a collaborative learning paradigm in which a pre-trained mentor network guides the training of a student network. Based on the small-loss trick, the mentor network provides the student network with samples whose labels are liekly to be correct. *Co-teaching* [78] and *Co-teaching+* [79] also maintain two DNNs, but each DNN selects a certain number of small-loss samples and feeds them to its peer DNN for further training. Compared with *Co-teaching*, *Co-teaching+* further employs the disagreement strategy of *decouple*. *ITLM* [81] iteratively minimizes the trimmed loss by alternating between selecting true-labeled samples at the current moment and retraining the DNN using them. At each training round, only a fraction of small-loss samples obtained in the current round are used to retrain the DNN in the next round. *INCV* [82] randomly divides noisy training data and then employs cross-validation to classify true-labeled samples while removing large-loss samples at each training round. Here, *Co-teaching* is adopted to train the DNN on the identified samples in the final round of training. Other than the small-loss trick, *iterative detection* [80] detects false-labeled samples by employing the local outlier factor algorithm [96]. Furthermore, it uses similar or dissimilar sample pairs to learn deep discriminative features; then, it gradually pulls away false-labeled samples from true-labeled samples in the deep feature space.

Recently, researchers have attempted to combine sample selection with other approaches. *SELFIE* [18] is a hybrid approach of sample selection and loss correction. The loss of refurbishable samples is corrected (i.e., loss correction) and then used together with that of small-loss samples (i.e., sample selection). Consequently, more training samples are considered for updating the DNN. Meanwhile, the *curriculum loss (CL)* [56] is combined with the robust loss function approach and used to extract the true-labeled samples from a noisy dataset based on a manually specified selection threshold. *SELF* [83] is combined with a semi-supervised learning approach to progressively filter out false-labeled samples. It exploits the *mean-teacher* model [97] to obtain a more stable supervisory signal than the noisy model snapshot.

This family of methods effectively avoids the risk of false correction by simply excluding unreliable samples. However, they may eliminate numerous useful samples, and their general philosophy of selecting small-loss samples is applicable to only some limited cases such as symmetric noise [18]. In addition, either the true noise rate or a clean validation dataset must be available to quantify the number of samples that should be selected as true-labeled ones by the model [78], [81].

F. Meta Learning

In recent years, meta learning has become an important topic in the machine learning community [98]–[101]. The key concept is *learning to learn*, which performs learning at a level higher than conventional learning. Generally, meta learning is applied to improve noise robustness based on two approaches: *1) fast adaption* that trains a model applicable to various learning tasks without overfitting to false labels and *2) learning to update* that learns the loss adjustment rule to reduce the negative effects of the noisy labels.

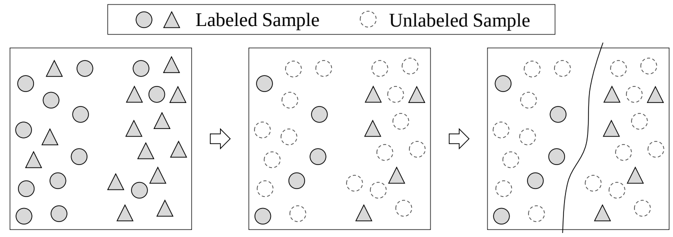
1) ***Fast Adaption:*** The *meta-regressor* [52] proposes to build a regressor that estimates the performances of robust training methods in a new noise learning task. It generates a meta-dataset composed of meta-features and meta-labels, where a meta-feature is the descriptor of a specified dataset, such as the number of classes and the number of features, and a meta-label is the F1-score of a specified robust training method obtained by learning the model for a synthetically corrupted dataset. Using the regressor trained on the meta-dataset, the meta-regressor recommends the most promising training method when applied to a newly given dataset. Meanwhile, *MLNT* [88] aims to obtain model parameters that can be easily fine-tuned or transferred to different label noises. To learn such model parameters, it generates multiple mini-batches with synthetically corrupted labels and then uses them to update the DNN such that the difference between the predictions obtained from the original and corrupted mini-batches is minimized.

These meta learning methods are general and model-agnostic. However, their drawback is the scalability caused by multiple inferences or updates prior to a conventional update for guiding their learning process.

2) ***Learning to Update****:* This approach is similar to loss adjustment in Section III-D, but the adjustment is automated in a meta-learning manner. *Knowledge distillation* [84] adopts the technique of transferring knowledge from one expert model to a target model. For the *label refurbishment* in Eq. (9), the prediction from the expert DNN trained on a small clean validation dataset is used instead of the prediction $\hat{y}$ from the target DNN. In addition, it leverages a knowledge graph, which encodes the structure of the label space, to further elaborate the expert DNN's prediction.

Meanwhile, other methods aim at learning the weight function $w(x, \tilde{y})$ for *loss reweighting* in Eq. (8). Specifically, *L2LWS* [85] and *CWS* [86] propose a unified neural architecture composed of a target DNN and a meta-DNN. The meta-DNN is trained on a small clean validation dataset; it then provides guidance to evaluate the weight score for the target DNN. Here, part of the two DNNs are shared and jointly trained to benefit from each other. *Automatic reweighting* [87] proposes a meta learning algorithm that learns the weights of training samples based on their gradient directions. It includes a small clean validation dataset into the training dataset and reweights the backward loss of the mini-batch samples such that the updated gradient minimizes the loss of this validation dataset. *Meta-Weight-Net* [89] parameterizes the weighting function as a multi-layer perceptron network with only one hidden layer. A meta-objective is defined to update its parameters such that they minimize the empirical risk of a small clean dataset. At each iteration, the parameter of the target network is guided by the weight function updated via the meta-objective. Likewise, *Data Coefficient* [101] proposes a meta re-weighting framework that optimizes the weights of samples by leveraging a small clean set, which is only 0.2% of the total training set, while refurbishing the label of samples probably mislabeled.

By learning the update rule via meta-learning, the trained network easily adapts to various types of label noise. Nevertheless, an unbiased clean validation dataset is essential to minimize the auxiliary objective for meta-learning, although it may not be available in real-world scenarios.



(a) Noisy Data. (b) Transformed Data. (c) SSL.

Fig. 6. Procedures for semi-supervised learning under label noise.

G. Semi-supervised Learning

Semi-supervised learning is another method that reduces the annotation cost in the presence of both labeled and unlabeled data [102], [103]. To overcome noisy labels, several recent studies transformed the problem of learning from noisy labels into a semi-supervised learning task [53], [83], [90], [91]. In general, as shown in Figure 6, possibly false-labeled samples in noisy data are treated as *unlabeled*, whereas the remaining samples are treated as *labeled*. Subsequently, semi-supervised learning is performed using the transformed data.

Label aggregation [53] adopts a simple label aggregation strategy used in semi-supervised learning. Several low-cost weak networks are trained on a clean validation dataset as multiple annotators. Subsequently, the true label of a noisy sample is obtained through a weighted label aggregation (i.e., ensemble) of the annotators' predictions. This method skips the process of extracting clean-labeled samples in Figure 6(b) by assuming the existence of the clean validation dataset. By contrast, the *two-stage framework* [90] pre-trains a DNN on noisy training data and then manages the labeled set by only maintaining samples for which the DNN's prediction probability to the annotated label is higher than a certain threshold, while the rest samples are included in the unlabeled set. Next, the transformed data are used to train another network using a semi-supervised method called *temporal ensembling* [104]. Most recently, *SELF* [83] adopted the concept of self-ensemble in semi-supervised learning to produce a more supervisory signal to filter out false-labeled samples during training. By maintaining the running average model as the backbone, it obtains the self-ensemble predictions of all training samples and then progressively removes samples whose ensemble predictions do not agree with their annotated labels. This method further leverages unsupervised loss from the samples not included in the selected set. Meanwhile, *DivideMix* [91] is an extension of the semi-supervised data augmentation technique called *MixMatch* [105]. A two-component and one-dimensional Gaussian mixture model is fitted to the training loss to obtain the confidence of an annotated label. By setting a confidence threshold, the training data is categorized into a labeled set and an unlabeled set. Subsequently, *MixMatch* is employed to train a DNN for the transformed data.

Noise robustness is significantly improved using several semi-supervised techniques. However, the hyperparameters introduced by these techniques render a DNN more susceptible to changes in data and noise types, and an increase in computational cost is inevitable.

TABLE III
COMPARISON OF ALL PROPOSED DEEP LEARNING METHODS FOR OVERCOMING NOISY LABELS.

Category		Method	P1	P2	P3	P4	P5	P6	Implementation
Robust Loss Function (§III-A)		*Robust MAE* [49]	○	○	○	○	×	×	N/A
		Generalized Cross Entropy [54]	○	○	○	○	×	×	Unofficial (PyTorch)[1]
		Symmetric Cross Entropy [55]	○	○	○	○	×	×	Official (Keras)[2]
		Curriculum Learning [56]	○	○	○	×	○	△	N/A
Robust Architecture (§III-B)	Noisy Adaptation Layer	*Webly Learning* [57]	△	×	○	○	×	×	Official (Caffe)[3]
		Noise Model [58]	△	○	○	○	×	×	Unofficial (Keras)[4]
		Dropout Noise Model [59]	△	○	○	○	×	×	Official (MATLAB)[5]
		S-model [60]	△	○	○	○	×	×	Official (Keras)[6]
		C-model [60]	△	○	○	○	×	○	Official (Keras)[6]
		NLNN [61]	△	○	○	○	×	×	Unofficial (Chainer)[7]
	Dedicated Architecture	*Probablistic Noise Model* [15]	×	×	○	×	△	○	Official (Caffe)[8]
		Masking [62]	×	○	○	×	△	○	Official (TensorFlow)[9]
		Contrastive-Additive Noise Network [63]	×	○	○	○	△	○	N/A
Robust Regularization (§III-C)		*Adversarial Training* [64]	○	○	○	○	△	△	Unofficial (PyTorch)[10]
		Label Smoothing [65]	○	○	○	○	△	△	Unofficial (PyTorch)[11]
		Mixup [66]	○	○	○	○	△	△	Official (PyTorch)[12]
		Bilevel Learning [67]	○	○	○	×	△	△	Official (TensorFlow)[13]
		Annotator Confusion [68]	○	×	○	○	△	△	Official (TensorFlow)[14]
		Pre-training [69]	○	×	○	○	△	△	Official (PyTorch)[15]
Loss Adjustment (§III-D)	Loss Correction	*Backward Correction* [70]	○	○	○	×	×	×	Official (Keras)[16]
		Forward Correction [70]	○	○	○	×	×	×	Official (Keras)[16]
		Gold Loss Correction [71]	○	×	○	×	×	×	Official (PyTorch)[17]
	Loss Reweigting	*Importance Reweighting* [72]	○	○	○	○	×	△	Unofficial (PyTorch)[18]
		Active Bias [73]	○	○	○	○	×	△	Unofficial (TensorFlow)[19]
	Label Refurbishment	*Bootstrapping* [74]	○	○	○	×	×	△	Unofficial (Keras)[20]
		Dynamic Bootstrapping [75]	○	○	○	○	×	△	Official (PyTorch)[21]
		D2L [76]	○	○	○	○	×	△	Official (Keras)[22]
		SELFIE [18]	○	○	○	×	○	△	Official (TensorFlow)[23]
Sample Selection (§III-E)		*Decouple* [51]	○	○	×	○	×	△	Official (TensorFlow)[24]
		MentorNet [77]	×	×	×	×	○	△	Official (TensorFlow)[25]
		Co-teaching [78]	○	○	×	×	○	△	Official (PyTorch)[26]
		Co-teaching+ [79]	○	○	×	×	○	△	Official (PyTorch)[27]
		Iterative Detection [80]	○	○	×	○	○	△	Official (Keras)[28]
		ITLM [81]	○	○	×	×	○	△	Official (GluonCV)[29]
		INCV [82]	○	○	×	○	○	△	Official (Keras)[30]
Meta Learning (§III-F)	Fast Adaption	*Meta-Regressor* [52]	○	○	○	×	○	○	Official (R)[31]
		MLNT [88]	○	○	○	○	×	○	Official (PyTorch)[32]
	Learning to Update	*Knowledge Distillation* [84]	○	×	○	×	△	○	N/A
		L2LWS [85]	×	○	○	×	△	○	Unofficial (TensorFlow)[33]
		CWS [86]	×	○	○	×	△	○	N/A
		Automatic Reweighting [87]	○	○	○	×	△	○	Official (TensorFlow)[34]
		Meta-Weight-Net [89]	△	○	○	×	△	○	Official (PyTorch)[35]
		Data Coefficients [101]	○	○	○	×	○	○	Official (TensorFlow)[36]
Semi-supervised Learning (§III-E)		*Label Aggregation* [53]	○	×	○	×	×	△	N/A
		Two-Stage Framework [90]	○	×	○	○	○	△	N/A
		SELF [83]	○	○	○	○	○	△	N/A
		DivideMix [91]	○	○	○	○	○	△	Official (PyTorch)[37]

[1] https://github.com/AlanChou/Truncated-Loss
[2] https://github.com/YisenWang/symmetric_cross_entropy_for_noisy_labels
[3] https://github.com/endernewton/webly-supervised
[4] https://github.com/delchiaro/training-cnn-noisy-labels-keras
[5] https://github.com/ijindal/Noisy_Dropout_regularization
[6] https://github.com/udibr/noisy_labels
[7] https://github.com/Ryo-Ito/Noisy-Labels-Neural-Network
[8] https://github.com/Cysu/noisy_label
[9] https://github.com/bhanML/Masking
[10] https://https://github.com/sarathknv/adversarial-examples-pytorch
[11] https://github.com/CoinCheung/pytorch-loss
[12] https://github.com/facebookresearch/mixup-cifar10
[13] https://github.com/sjenni/DeepBilevel
[14] https://rt416.github.io/pdf/trace_codes.pdf
[15] github.com/hendrycks/pre-training
[16] https://github.com/giorgiop/loss-correction
[17] https://github.com/mmazeika/glc
[18] https://github.com/xiaoboxia/Classification-with-noisy-labels
[19] https://github.com/songhwanjun/ActiveBias
[20] https://github.com/dr-darryl-wright/Noisy-Labels-with-Bootstrapping
[21] https://github.com/PaulAlbert31/LabelNoiseCorrection
[22] https://github.com/xingjunm/dimensionality-driven-learning
[23] https://github.com/kaist-dmlab/SELFIE
[24] https://github.com/emalach/UpdateByDisagreement
[25] https://github.com/google/mentornet
[26] https://github.com/bhanML/Co-teaching
[27] https://github.com/bhanML/coteaching_plus
[28] https://github.com/YisenWang/Iterative_learning
[29] https://github.com/yanyao-shen/ITLM-simplecode
[30] https://github.com/chenpf1025/noisy_label_understanding_utilizing

TABLE IV
COMPARISON OF ROBUST DEEP LEARNING CATEGORIES FOR OVERCOMING NOISY LABELS.

Category		**P1** Flexibility	**P2** No Pre-train	**P3** Full Exploration	**P4** No Supervision	**P5** Heavy Noise	**P6** Complex Noise
Robust Loss Function (§III-A)		○	○	○	○	×	×
Robust Architecture (§III-B)	Noise Adaptation Layer	△	○	○	○	×	×
	Dedicated Architecture	×	○	○	×	△	○
Robust Regularization (§III-C)		○	○	○	○	△	△
Loss Adjustment (§III-D)	Loss Correction	○	○	○	×	×	×
	Loss Reweighting	○	○	○	○	×	△
	Label Refurbishment	○	○	○	△	×	△
Sample Selection (§III-E)		○	○	×	×	○	△
Meta Learning (§III-F)	Fast Adaption	○	○	○	△	△	○
	Learning to Update	○	○	○	×	△	○
Semi-supervised Learning (§III-G)		○	△	○	○	○	△

IV. METHODOLOGICAL COMPARISON

In this section, we compare the 47 deep learning methods for overcoming noisy labels introduced in Section III with respect to the following *six* properties. When selecting the properties, we refer to the properties that are typically used to compare the performance of robust deep learning methods [18], [78]. To the best of our knowledge, this survey is the first to provide a systematic comparison of robust training methods. This comprehensive comparison will provide useful insights for future studies.

- **(P1) Flexibility:** With the rapid evolution of deep learning research, a number of new network architectures are constantly emerging and becoming available. Hence, the ability to support any type of architecture is important. "Flexibility" ensures that the proposed method can quickly adapt to the state-of-the-art architecture.
- **(P2) No Pre-traing:** A typical approach to improve noise robustness is to use a pre-trained network; however, this incurs additional computational cost to the learning process. "No Pre-training" ensures that the proposed method can be trained from scratch without any pre-training.
- **(P3) Full Exploration:** Excluding unreliable samples from the update is an effective method for robust deep learning; however, it eliminates hard but useful training samples as well. "Full Exploration" ensures that the proposed methods can use *all* training samples without severe overfitting to false-labeled samples by adjusting their training losses.
- **(P4) No Supervision:** Learning with supervision, such as a clean validation set or a known noise rate, is often impractical because they are difficult to obtain. Hence, such supervision had better be avoided to increase practicality in real-world scenarios. "No Supervision" ensures that the proposed methods can be trained without any supervision.
- **(P5) Heavy Noise:** In real-world noisy data, the noise rate can vary from light to heavy. Hence, learning methods should achieve consistent noise robustness with respect to the noise rate. "Heavy Noise" ensures that the proposed methods can combat even the heavy noise.
- **(P6) Complex Noise:** The type of label noise significantly affects the performance of a learning method. To manage real-world noisy data, diverse types of label noise should be considered when designing a robust training method. "Complex Noise" ensures that the proposed method can combat even the complex label noise.

Table III shows a comparison of all robust deep learning methods, which are grouped according to the most appropriate category. In the first row, the aforementioned six properties are labeled as P1–P6, and the availability of open source implementation is added in the last column. For each property, we assign "○" if it is completely supported, "×" if it is not supported, and "△" if it is supported but not completely. More specifically, "△" is assigned to P1 if the method can be flexible but requires additional effort, to P5 if the method can combat only moderate label noise, and to P6 if the method does not make a strict assumption about the noise type but without explicitly modeling complex noise. The remaining properties (i.e., P2, P3, and P4) are only assigned "○" or "×". Regarding the implementation, we assign "N/A" if a publicly available source code is not available.

No existing method supports all the properties. Each method achieves noise robustness by supporting a different combination of the properties. The supported properties are similar among the methods of the same (sub-)category because those methods share the same methodological philosophy; however, they differ significantly depending on the (sub-)category. Therefore, we investigate the properties generally supported in each (sub-)category and summarize them in Table IV. Here, the property of a (sub-)category is marked as the majority of the belonging methods. If no clear trend is observed among those methods, then the property is marked "△".

V. EXPERIMENTAL DESIGN

This section describes the typically used experimental design for comparing robust training methods in the presence of label noise. We introduce publicly available image datasets and then describe widely-used evaluation metrics.

[31] https://github.com/lpfgarcia/m2n
[32] https://github.com/LiJunnan1992/MLNT
[33] https://github.com/krayush07/learn-by-weak-supervision
[34] https://github.com/uber-research/learning-to-reweight-examples
[35] https://github.com/xjtushujun/meta-weight-net
[36] https://github.com/google-research/google-research/tree/master/ieg
[37] https://github.com/LiJunnan1992/DivideMix

TABLE V
SUMMARY OF PUBLICLY AVAILABLE DATASETS USED FOR STUDYING LABEL NOISE.

	Dataset	**# Training**	**# Validation**	**# Testing**	**# Classes**	**Noise Rate (%)**
Clean Data	MNIST [106][38]	60K	N/A	10K	10	≈ 0.0
	Fashion-MNIST [107][39]	60K	N/A	10K	10	≈ 0.0
	CIFAR-10 [108][40]	50K	N/A	10K	10	≈ 0.0
	CIFAR-100 [108][40]	50K	N/A	10K	100	≈ 0.0
	SVHN [109][41]	73K	N/A	26K	10	≈ 0.0
	Tiny-ImageNet [110][43]	100K	10K	10K	200	≈ 0.0
	ImageNet [1][42]	1.3M	50K	50K	1000	≈ 0.0
Real-world Noisy Data	ANIMAL-10N [18][44]	50K	N/A	5K	10	≈ 8.0
	Food-101N [17][45]	310K	5K	25K	101	≈ 18.4
	Clothing1M [15][46]	1M	14K	10K	14	≈ 38.5
	WebVision [16][47]	2.4M	50K	50K	1000	≈ 20.0

A. *Publicly Available Datasets*

To validate the robustness of the proposed algorithms, an image classification task was widely conducted on numerous image benchmark datasets. Table V summarizes popularly-used public benchmark datasets, which are classified into two categories: *1)* a "clean dataset" that consists of mostly true-labeled samples annotated by human experts and *2)* a "real-world noisy dataset" that comprises real-world noisy samples with varying numbers of false labels.

1) Clean Datasets: According to the literature [18], [80], [91], *seven* clean datasets are widely used: MNIST[38], classification of handwritten digits [106]; Fashion-MNIST[39], classification of various clothing [107]; CIFAR-10[40] and CIFAR-100[40], classification of a subset of 80 million categorical images [108]; SVHN[41], classification of house numbers in Google Street view images [109]; ImageNet[42] and Tiny-ImageNet[43], image database organized according to the WordNet hierarchy and its small subset [1], [110]. Because the labels in these datasets are almost all true-labeled, their labels in the training data should be artificially corrupted for the evaluation of synthetic noises, namely *symmetric* noise and *asymmetric* noise.

2) Real-world Noisy Datasets: Unlike the clean datasets, real-world noisy datasets inherently contain many mislabeled samples annotated by non-experts. According to the literature [15]–[18], *four* real-world noisy datasets are widely used: ANIMAL-10N[44], real-world noisy data of human-labeled online images for 10 confusing animals [18]; Food-101N[45], real-world noisy data of crawled food images annotated by their search keywords in the Food-101 taxonomy [17], [111]; Clothing1M[46], real-world noisy data of large-scale crawled clothing images from several online shopping websites [15]; WebVision[47], real-world noisy data of large-scale web images crawled from Flickr and Google Images search [16]. To support sophisticated evaluation, most real-world noisy datasets contain their own clean validation set and provide the estimated noise rate of their training set.

B. *Evaluation Metrics*

A typical metric to assess the robustness of a particular method is the prediction accuracy for unbiased and clean samples that are not used in training. The prediction accuracy degrades significantly if the DNN overfits to false-labeled samples [20]. Hence, *test accuracy* has generally been adopted for evaluation [13]. For a test set $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{T}|}$, let $\hat{y}_i$ be the predicted label of the i-th sample in $\mathcal{T}$. Subsequently, the test accuracy is formalized by

$$\text{Test Accuracy} = \frac{|\{(x_i, y_i) \in \mathcal{T} : \hat{y}_i = y_i\}|}{|\mathcal{T}|}. \tag{11}$$

If the test data are not available, *validation accuracy* can be used by replacing $\mathcal{T}$ in Eq. (11) with validation data $\mathcal{V} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{V}|}$ as an alternative,

$$\text{Validation Accuracy} = \frac{|\{(x_i, y_i) \in \mathcal{V} : \hat{y}_i = y_i\}|}{|\mathcal{V}|}. \tag{12}$$

Furthermore, if the specified method belongs to the "sample selection" category, *label precision* and *label recall* [78], [82] can be used as the metrics,

$$\begin{aligned} \text{Label Precision} &= \frac{|\{(x_i, \tilde{y}_i) \in \mathcal{S}_t : \tilde{y}_i = y_i\}|}{|\mathcal{S}_t|}, \\ \text{Label Recall} &= \frac{|\{(x_i, \tilde{y}_i) \in \mathcal{S}_t : \tilde{y}_i = y_i\}|}{|\{(x_i, \tilde{y}_i) \in \mathcal{B}_t : \tilde{y}_i = y_i\}|}, \end{aligned} \tag{13}$$

where $\mathcal{S}_t$ is the set of selected clean samples from a mini-batch $\mathcal{B}_t$. These two metrics are indicators of performance for the samples selected from the mini-batch as true-labeled ones [78].

Meanwhile, if the specified method belongs to the "label refurbishment" category, *correction error* [18] can be used as an indicator of how many samples are incorrectly refurbished,

$$\text{Correction Error} = \frac{|\{x_i \in \mathcal{R} : \text{argmax}(y_i^{refurb}) \neq y_i\}|}{|\mathcal{R}|}, \tag{14}$$

[38] http://yann.lecun.com/exdb/mnist
[39] https://github.com/zalandoresearch/fashion-mnist
[40] https://www.cs.toronto.edu/~kriz/cifar.html
[41] http://ufldl.stanford.edu/housenumbers
[42] http://www.image-net.org
[43] https://www.kaggle.com/c/tiny-imagenet
[44] https://dm.kaist.ac.kr/datasets/animal-10n
[45] https://kuanghuei.github.io/Food-101N
[46] https://www.floydhub.com/lukasmyth/datasets/clothing1m
[47] https://data.vision.ee.ethz.ch/cvl/webvision/download.html

where $\mathcal{R}$ is the set of samples whose labels are refurbished by Eq. (9) and y_i^{refurb} is the refurbished label of the i-th samples in $\mathcal{R}$.

VI. Future Research Directions

This section presents a few challenging but interesting future research directions:

- **Undistinguishable Sample:** Most studies have exploited the small-loss trick to select clean samples from noisy training data. However, difficult-to-learn samples with true labels are not distinguishable from samples with false labels because they exhibit large losses as well. Hence, they are easily misclassified as false-labeled samples and then excluded from the update, although they are informative for learning. This over-cleaning issue becomes more challenging, especially when many ambiguous classes exist in the training data. Distinguishing them from noisy data significantly affects robust deep learning.
- **Complex Label Noise:** Complex label noise, such as instance- and label-dependent label noise, has not been studied extensively. Most studies have mainly focused on two artificial label noises, i.e., symmetric and asymmetric, though these two noises are not very realistic in the real world. Hence, more diverse types of complex label noise should be studied to improve the practicality or generalizability of the algorithm.
- **Multi-label Data:** It is typically assumed that each data sample has only one true label. However, the data sample can be associated with a set of multiple true labels in modern applications, such as semantic scene classification [112] and music categorization [113]. In this setup, the annotator may omit some labels for a sample, thereby resulting in more diverse types of label noise. Therefore, managing label noise in multi-label data is a challenging future direction.
- **Learning Efficiency:** For robust deep learning, most studies have neglected the efficiency of the algorithm because their main goal is to improve the robustness to label noise. For example, maintaining multiple networks or training the DNN multiple rounds is frequently used, though it significantly degrades the learning efficiency of the algorithm. Owing to the rapid increase in the amount of available data, learning efficiency has become critical. Therefore, enhancing the efficiency of the method will significantly increase its usability in the big data era.

VII. Conclusion

DNNs easily overfit to false labels owing to their high capacity in totally memorizing all noisy training samples. This overfitting issue still remains even with various conventional regularization techniques, such as dropout and batch normalization, thereby significantly decreasing their generalization performance. Even worse, in real-world applications, the difficulty in labeling renders the overfitting issue more severe. Therefore, learning from noisy labels has recently become one of the most active research topics in the machine learning community.

In this survey, we presented a comprehensive understanding of modern deep learning methods to address the negative consequences of learning from noisy labels. All the methods were grouped into seven categories according to their underlying strategies and described in chronological order along with their methodological weaknesses. Furthermore, a systematic comparison was conducted using six popular properties used for evaluation in the recent literature. According to the comparison results, there is no ideal method that supports all the required properties; the supported properties varied depending on the category to which each method belonged. Several experimental guidelines were also discussed, including publicly available datasets and evaluation metrics. Finally, we provided insights and directions for future research in this domain.

Acknowledgements

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00862, DB4DL: High-Usability and Performance In-Memory Distributed DBMS for Deep Learning).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012, pp. 1097–1105.
[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016, pp. 779–788.
[3] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. CVPR*, 2017, pp. 3156–3164.
[4] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proc. ECIR*, 2016, pp. 45–57.
[5] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng, "Deeprank: A new deep architecture for relevance ranking in information retrieval," in *Proc. CIKM*, 2017, pp. 257–266.
[6] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara *et al.*, "Neural information retrieval: At the end of the early years," *Information Retrieval Journal*, vol. 21, no. 2-3, pp. 111–182, 2018.
[7] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. ACL*, 2018, pp. 328–339.
[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. ACL*, 2019, pp. 4171–4186.
[9] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proc. ACL*, 2015, pp. 959–962.
[10] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on amazon mechanical turk," *Judgment and Decision Making*, vol. 5, no. 5, pp. 411–419, 2010.
[11] V. Cothey, "Web-crawling reliability," *Journal of the American Society for Information Science and Technology*, vol. 55, no. 14, pp. 1228–1238, 2004.
[12] W. Mason and S. Suri, "Conducting behavioral research on amazon's mechanical turk," *Behavior Research Methods*, vol. 44, no. 1, pp. 1–23, 2012.
[13] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Transaction on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2013.
[14] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines." in *Proc. ECAI*, 2012, pp. 870–875.
[15] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proc. CVPR*, 2015, pp. 2691–2699.
[16] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.

[17] K.-H. Lee, X. He, L. Zhang, and L. Yang, "CleanNet: Transfer learning for scalable image classifier training with label noise," in *Proc. CVPR*, 2018, pp. 5447–5456.
[18] H. Song, M. Kim, and J.-G. Lee, "SELFIE: Refurbishing unclean samples for robust deep learning," in *Proc. ICML*, 2019, pp. 5907–5915.
[19] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *Proc. ICML*, 2017, pp. 233–242.
[20] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. ICLR*, 2017.
[21] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.
[22] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc, NeurIPS*, 1992, pp. 950–957.
[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.
[25] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial Intelligence Review*, vol. 22, no. 3, pp. 177–210, 2004.
[26] J. Zhang, X. Wu, and V. S. Sheng, "Learning from crowdsourced labeled data: A survey," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 543–576, 2016.
[27] N. Nigam, T. Dutta, and H. P. Gupta, "Impact of noisy labels in learning techniques: A survey," in *Proc. ICDIS*, 2020, pp. 403–411.
[28] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
[29] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Proc. NeurIPS*, 2013, pp. 1196–1204.
[30] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proc. ICLR*, 2017.
[31] V. Wheway, "Using boosting to detect noisy data," in *Proc. PRICAI*, 2000, pp. 123–130.
[32] B. Sluban, D. Gamberger, and N. Lavrač, "Ensemble-based noise detection: Noise ranking and visual performance evaluation," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 265–303, 2014.
[33] S. J. Delany, N. Segata, and B. Mac Namee, "Profiling instances in noise reduction," *Knowledge-Based Systems*, vol. 31, pp. 28–40, 2012.
[34] D. Gamberger, N. Lavrac, and S. Dzeroski, "Noise detection and elimination in data preprocessing: Experiments in medical domains," *Applied Artificial Intelligence*, vol. 14, no. 2, pp. 205–223, 2000.
[35] J. Thongkam, G. Xu, Y. Zhang, and F. Huang, "Support vector machine for outlier detection in breast cancer survivability prediction," in *Proc. APWeb*, 2008, pp. 99–109.
[36] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proc. ICML*, 2012, pp. 567–574.
[37] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.
[38] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.
[39] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *Proc. NeurIPS*, 2015, pp. 10–18.
[40] G. Patrini, F. Nielsen, R. Nock, and M. Carioni, "Loss factorization, weakly supervised learning and label noise robustness," in *Proc. ICML*, 2016, pp. 708–717.
[41] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.
[42] U. Rebbapragada and C. E. Brodley, "Class noise mitigation through instance weighting," in *Proc. ECML*, 2007, pp. 708–715.
[43] T. Liu, K. Wang, B. Chang, and Z. Sui, "A soft-label method for noise-tolerant distantly supervised relation extraction," in *Proc. EMNLP*, 2017, pp. 1790–1795.
[44] F. O. Kaster, B. H. Menze, M.-A. Weber, and F. A. Hamprecht, "Comparative validation of graphical models for learning tumor segmentations from noisy manual annotations," in *Proc. MICCAI*, 2010, pp. 74–85.
[45] A. Ganapathiraju and J. Picone, "Support vector machines for automatic data cleanup," in *Proc. ICSLP*, 2000.
[46] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proc. ACML*, 2011, pp. 97–112.
[47] C. J. Mantas and J. Abellán, "Credal-C4. 5: Decision tree based on imprecise probabilities to classify noisy data," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4625–4637, 2014.
[48] A. Ghosh, N. Manwani, and P. Sastry, "On the robustness of decision tree learning under label noise," in *Proc. PAKDD*, 2017, pp. 685–697.
[49] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proc. AAAI*, 2017.
[50] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. ICLR*, 2015.
[51] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," in *Proc. NeurIPS*, 2017, pp. 960–970.
[52] L. P. Garcia, A. C. de Carvalho, and A. C. Lorena, "Noise detection in the meta-learning level," *Neurocomputing*, vol. 176, pp. 14–25, 2016.
[53] Y. Yan, Z. Xu, I. W. Tsang, G. Long, and Y. Yang, "Robust semi-supervised learning through label aggregation," in *Proc. AAAI*, 2016.
[54] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. NeurIPS*, 2018, pp. 8778–8788.
[55] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. ICCV*, 2019, pp. 322–330.
[56] Y. Lyu and I. W. Tsang, "Curriculum loss: Robust learning and generalization against label corruption," in *Proc. ICLR*, 2020.
[57] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *Proc. ICCV*, 2015, pp. 1431–1439.
[58] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *Proc. ICLRW*, 2015.
[59] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization," in *Proc. ICDM*, 2016, pp. 967–972.
[60] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proc. ICLR*, 2017.
[61] A. J. Bekker and J. Goldberger, "Training deep neural-networks based on unreliable labels," in *Proc. ICASSP*, 2016, pp. 2682–2686.
[62] B. Han, J. Yao, G. Niu, M. Zhou, I. Tsang, Y. Zhang, and M. Sugiyama, "Masking: A new perspective of noisy supervision," in *Proc. NeurIPS*, 2018, pp. 5836–5846.
[63] J. Yao, J. Wang, I. W. Tsang, Y. Zhang, J. Sun, C. Zhang, and R. Zhang, "Deep learning from noisy image labels with quality embedding," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1909–1922, 2018.
[64] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, 2014.
[65] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," in *Proc. ICLRW*, 2017.
[66] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.
[67] S. Jenni and P. Favaro, "Deep bilevel learning," in *Proc. ECCV*, 2018, pp. 618–633.
[68] R. Tanno, A. Saeedi, S. Sankaranarayanan, D. C. Alexander, and N. Silberman, "Learning from noisy labels by regularized estimation of annotator confusion," in *Proc. CVPR*, 2019, pp. 11 244–11 253.
[69] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in *Proc. ICML*, 2019.
[70] 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. CVPR*, 2017, pp. 1944–1952.
[71] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *Proc. NeurIPS*, 2018, pp. 10 456–10 465.
[72] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2568–2580, 2017.
[73] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active Bias: Training more accurate neural networks by emphasizing high variance samples," in *Proc. NeurIPS*, 2017, pp. 1002–1012.
[74] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. ICLR*, 2015.
[75] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *Proc. ICML*, 2019.

[76] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, "Dimensionality-driven learning with noisy labels," in *Proc. ICML*, 2018.

[77] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proc. ICML*, 2018.

[78] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. NeurIPS*, 2018, pp. 8527–8537.

[79] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption?" in *Proc. ICML*, 2019.

[80] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," in *Proc. CVPR*, 2018, pp. 8688–8696.

[81] Y. Shen and S. Sanghavi, "Learning with bad training data via iterative trimmed loss minimization," in *Proc. ICML*, 2019.

[82] P. Chen, B. Liao, G. Chen, and S. Zhang, "Understanding and utilizing deep neural networks trained with noisy labels," in *Proc. ICML*, 2019.

[83] 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," in *Proc. ICLR*, 2020.

[84] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, "Learning from noisy labels with distillation," in *Proc. ICCV*, 2017, pp. 1910–1918.

[85] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps, "Learning to learn from weak supervision by full supervision," in *Proc. NeurIPSW*, 2017.

[86] ——, "Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision," *arXiv preprint arXiv:1711.00313*, 2017.

[87] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. ICML*, 2018.

[88] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to learn from noisy labeled data," in *Proc. CVPR*, 2019, pp. 5051–5059.

[89] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-Weight-Net: Learning an explicit mapping for sample weighting," in *Proc. NeurIPS*, 2019, pp. 1917–1928.

[90] Y. Ding, L. Wang, D. Fan, and B. Gong, "A semi-supervised two-stage approach to learning from noisy labels," in *Proc. WACV*, 2018, pp. 1215–1224.

[91] J. Li, R. Socher, and S. C. Hoi, "DivideMix: Learning with noisy labels as semi-supervised learning," in *Proc. ICLR*, 2020.

[92] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NeurIPS*, 2014, pp. 2672–2680.

[93] H. Song, M. Kim, D. Park, and J.-G. Lee, "Prestopping: How does early stopping help generalization against label noise?" *arXiv preprint arXiv:1911.08059*, 2019.

[94] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[95] M. E. Houle, "Local intrinsic dimensionality I: An extreme-value-theoretic foundation for similarity applications," in *Proc. SISAP*, 2017, pp. 64–79.

[96] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

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

[98] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

[99] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. ICML*, 2016, pp. 1842–1850.

[100] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. ICML*, 2017, pp. 1126–1135.

[101] Z. Zhang, H. Zhang, S. O. Arik, H. Lee, and T. Pfister, "Distilling effective supervision from severe label noise," in *CVPR*, 2020, pp. 9294–9303.

[102] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[103] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: A brief survey," *A Review of Machine Learning Techniques for Processing Multimedia Content*, vol. 1, pp. 9–16, 2004.

[104] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *Proc. ICLR*, 2017.

[105] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "MixMatch: A holistic approach to semi-supervised learning," in *Proc. NeurIPS*, 2019, pp. 5050–5060.

[106] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, p. 34, 1998.

[107] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[108] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 and CIFAR-100 datasets," 2014, https://www.cs.toronto.edu/~kriz/cifar.html.

[109] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NeurIPSW*, 2011.

[110] A. Karpathy *et al.*, "Cs231n convolutional neural networks for visual recognition," *Neural Networks*, vol. 1, p. 1, 2016.

[111] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *Proc. ECCV*, 2014, pp. 446–461.

[112] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[113] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.