Jump-teaching: Combating Sample Selection Bias via Temporal Disagreement

Kangye Ji[1, 2 *], **Fei Cheng**[1 *†], **Zeqing Wang**[1,3], **Qichang Zhang**[4], **Bohu Huang**[1]
[1]School of Computer Science and Technology, Xidian University
[2]Tsinghua Shenzhen International Graduate School, Tsinghua University
[3]College of Design and Engineering, National University of Singapore
[4]Hangzhou Institute of Technology, Xidian University
jky25@mails.tsinghua.edu.cn, chengfei@xidian.edu.cn, zeqing.wang@u.nus.edu,
qichzhang@xidian.edu.cn, hbohu@xidian.edu.cn

Abstract

Sample selection is a straightforward technique to combat noisy labels, aiming to prevent mislabeled samples from degrading the robustness of neural networks. However, existing methods mitigate compounding selection bias either by leveraging dual-network disagreement or additional forward propagations, leading to multiplied training overhead. To address this challenge, we introduce *Jump-teaching*, an efficient sample selection framework for debiased model update and simplified selection criterion. Based on a key observation that a neural network exhibits significant disagreement across different training iterations, Jump-teaching proposes a jump-manner model update strategy to enable self-correction of selection bias by harnessing temporal disagreement, eliminating the need for multi-network or multi-round training. Furthermore, we employ a sample-wise selection criterion building on the intra variance of a decomposed single loss for a fine-grained selection without relying on batch-wise ranking or dataset-wise modeling. Extensive experiments demonstrate that Jump-teaching outperforms state-of-the-art counterparts while achieving a nearly overhead-free selection procedure, which boosts training speed by up to $4.47\times$ and reduces peak memory footprint by 54%.

Code — https://github.com/ky-ji/Jump-teaching
Extended version — https://arxiv.org/abs/2405.17137

1 Introduction

Learning with Noisy Labels (LNL) is a critical challenge for real-world AI applications (Mahajan et al. 2018; Bakhshi and Can 2024). Typically, noisy labels stem from mistaken annotations of the dataset, such as in crowd-sourcing (Welinder et al. 2010) and online query (Blum, Kalai, and Wasserman 2003). As accurate annotations of large datasets are a time-consuming endeavor, noisy labels become inevitable. Deep neural networks can easily overfit to noisy labels, which often leads to poor generalization performance (Zhang et al. 2021; Han et al. 2020).

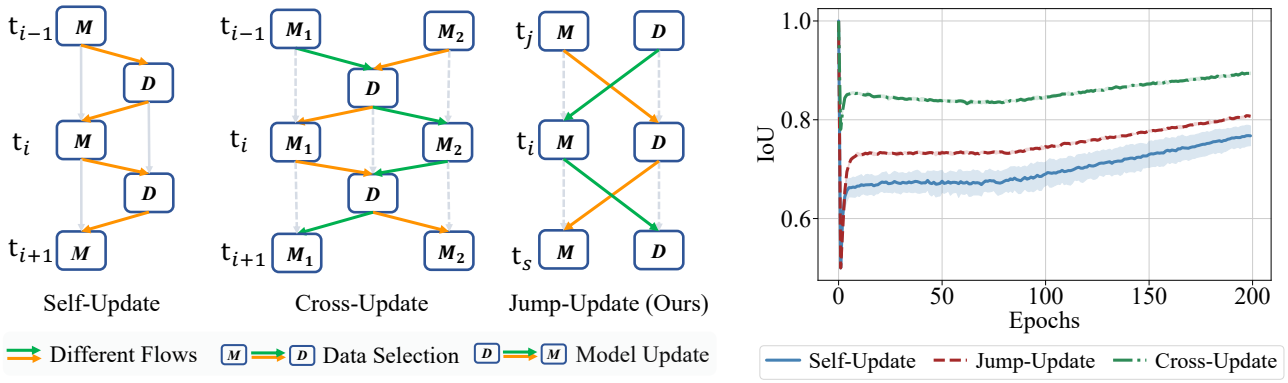
Despite sample selection representing a direct approach to combat label noise by selecting possibly clean samples from the corrupted dataset for training (Song, Kim, and Lee 2019; Kim et al. 2021; Wu et al. 2020; Malach and Shalev-Shwartz 2017; Xia et al. 2023; Xiao et al. 2022), selection bias has always been a fundamental obstacle for sample selection-based LNL methods. Selection bias refers to the tendency to favor noisy samples that have previously been selected and trained on (Han et al. 2018; Yu et al. 2019; Wei et al. 2022). This bias inevitably arises from the exposure of the classifier to noise and causes the inclusion of noisy labels in the training data. When the neural network trains on these data, errors will accumulate and the bias will be compounded.

Mitigating compounding selection bias demands costly training overheads. Explicit works typically utilize a dual-network training paradigm that employs the disagreement between networks in selection to correct the bias (Malach and Shalev-Shwartz 2017; Han et al. 2018; Tanaka et al. 2018; Huang, Zhang, and Shan 2023), resulting in twice the standard training overhead. Implicit works avoid the bias from being frequently amplified by selecting samples from the entire dataset at once (Bai and Liu 2021; Yuan, Feng, and Liu 2023; Yuan et al. 2025), though these methods typically demand several times the computational resources.

In this paper, we introduce *Jump-teaching*, an efficient framework to optimize the workflow of the sample selection-based LNL methods. Our key insight is that *a neural network exhibits significant disagreement across different training iterations*, based on the observation in Figure 1(b) that the intersection of the selected sets between different epochs of the network itself is much smaller than that between the two networks trained simultaneously. Motivated by this insight, we propose a jump-manner strategy termed *Jump-update Strategy* for a debiased model update. As illustrated in Figure 1(a), for each sample, our strategy generates its selection in iteration t_j of the current epoch and updates the model based on the former selection conducted in iteration t_i of the next epoch. In other words, the network in the previous epoch selects samples for the current network to train, which bridges the disagreement from a temporal perspective.

To achieve a training procedure without additional cost, we confine the selection process to operate solely at the mini-batch level. While the commonly adopted small-loss criterion (Han et al. 2018) meets this constraint, its reliance on noisy rate prior and batch-ranking operations makes it impractical

*These authors contributed equally.
†Corresponding Author.



(a) (b)

Figure 1: (a) The flow path of error under different update strategies. (b) "Disagreement" of different strategies with symmetric noise ratio $\epsilon = 0.5$ on CIFAR-10, quantified by the Intersection over Union (IoU) between two selections. In the cross-update strategy, disagreement arises between different networks, whereas in the other two strategies, it occurs across different iterations.

for standard training. Namely, we need a sample-wise selection criterion to fully unleash the potential of the jump-update strategy. However, one key issue is that a single loss value of a sample can't provide enough information to guide selection. To compensate for this lack of granularity, we decompose the loss for its intrinsic distribution, inspired by representation learning (Yang et al. 2015). With richer information, we further propose the *single-loss criterion*, based on the principle that *a sample with a distorted intra-loss distribution is more likely to have a noisy label*. To operationalize this criterion, we design a lightweight plugin that contains an auxiliary head and a prepared codebook to transform the outputs and labels into semantic embeddings and hash encodings, respectively, within an auxiliary space. This plugin decomposes the single loss value into a vector with each dimension representing the sub-loss corresponding to a specific semantic feature. In this way, a noisy encoding yields larger sub-losses in dimensions affected by noise and smaller sub-losses in unaffected dimensions, exhibiting a distorted distribution in this vector. We then employ variance as a simple yet effective metric to assess this *distorted* characteristic.

Benefiting from the jump-update strategy and the single-loss criterion, Jump-teaching updates the model with minimal bias and selects samples with finer granularity, achieving a new state-of-the-art across symmetric, asymmetric, pair-flip, real-world, and instance-dependent noise benchmarks. Moreover, it achieves this with an almost cost-free procedure, leading to $4.47\times$ increase in training speed and 54% reduction in peak memory usage compared to previous methods. In summary, our main contributions are threefold:

1. Based on the observation that significant disagreement persists across training iterations, we propose the jump-update strategy, which enables a single network to effectively self-correct selection bias.
2. We introduce Jump-teaching, an efficient sample selection framework, which employs the jump-update strategy for debiased model update and the single-loss criterion for sample-wise selection.
3. Jump-teaching outperforms state-of-the-art methods on extensive noisy label benchmarks, particularly under extreme noise conditions.

2 Related Work

Learning with Noisy Labels. Recent LNL methods can be divided into three categories: regularization, label correction, and sample selection. Regularization methods focus on crafting noise-robust loss functions (Ghosh, Kumar, and Sastry 2017; Wang et al. 2019) and regularization techniques (Liu et al. 2020; Zhang et al. 2020; Cao et al. 2021), but they cannot fully avoid fitting to noisy labels during training, resulting in sub-optimal outcomes. Label correction, integrating closely with semi-supervised learning, aims to refine or recreate pseudo-labels (Han, Luo, and Wang 2019; Sohn et al. 2020; Pham et al. 2021). These methods make use of corrected noisy labels but require computational resources for the estimation of noise transition matrix (Goldberger and Ben-Reuven 2022) or the ensemble prediction (Lu and He 2022). The core idea of the sample selection approach is to filter out noisy samples to prevent the network from fitting them. The approach typically operates in an iterative workflow, selecting possibly clean samples through a certain criterion and then updating the model parameters based on those samples. We review key studies regarding selection bias and selection criteria in the next two paragraphs.

Sample-selection Bias. Sample selection inherently involves bias, leading to error accumulation. To mitigate the bias, existing methods either employ multiple networks or multiple rounds, leading to sub-optimal performance. Decoupling (Malach and Shalev-Shwartz 2017) employs a teacher model to select clean samples to guide a student model, Co-teaching (Han et al. 2018) simultaneously trains two networks on data selected by a peer network. Co-teaching+ (Yu et al. 2019) maintains divergence by training only on samples where networks disagree while JoCoR (Wei et al. 2020) employs regularization to remain divergent. Moreover, many methods integrate the co-training framework to achieve advanced performance (Li, Socher, and Hoi 2020; Liu et al. 2020, 2022; Chen et al. 2023). However, these disagreement-based co-

training methods double the computational overhead while relying on a weaker disagreement than the network itself, which motivates us to improve them. A rich body of research implicitly addresses bias by minimizing selection operations, which typically adopt a dataset-wise selection criterion. We will discuss them in the next paragraph.

Sample-selection Criterion. Due to the lack of strong debiasing strategies, a dataset-wise selection criterion is preferred because it can not only improve the selection quality but also help prevent the repetitive amplification of bias. One line of works (Li, Socher, and Hoi 2020; Huang, Zhang, and Shan 2023; Kim et al. 2021) regards the sample selection as a binary classification problem, where the distribution of sample losses is modeled by techniques like Gaussian mixture models. Some other works (Wu et al. 2020; Iscen et al. 2022) conduct nearest neighbor analysis on the entire dataset, regarding the outliers as noisy samples. Several works (Toneva et al. 2018; Wei et al. 2022; Yuan, Feng, and Liu 2023, 2025) leverage sample-wise prediction differences between epochs for selection. They show evident redundancy in the additional forward propagations through the dataset before training and the repeated need to aggregate batches of data for noisy label estimation. Notably, although the observations of prediction differences share some commonality with our work, they have never been considered for debiasing. Despite some methods (Han et al. 2018; Tanaka et al. 2018) relying on the small-loss criterion that ranks samples within each batch by their loss magnitude, prioritizing those with smaller losses, they necessitate prior knowledge about the noise rate, which is impractical in real-world scenarios.

3 Methodology

We introduce Jump-teaching, an efficient sample selection framework. Specifically, it employs the *Jump-update Strategy* to mitigate the selection bias and the *Single-loss Criterion* for a sample-wise selection. We demonstrate them in Section 3.1 and Section 3.2, respectively.

3.1 Jump-update Strategy

Inspired by the observation in Figure 1(b), we propose the Jump-update Strategy, a jump-manner model update strategy designed to harness disagreement from ancestor iterations in a single network for bias correction. We provide a detailed strategy description, followed by an empirical quantitative analysis to investigate its debiasing mechanism and validate our insights through experimental analysis.

Strategy Description. To simplify our discussion, we dive into the procedure of sample selection and give some definitions. Sample selection is accomplished by the execution of model updates and sample selection with multiple iterations. In other words, model updates and sample selection are performed only once in every iteration. Previous strategies and ours are compared in Figure 1(a).

Suppose t_i denotes the current i-th iteration, **ancestor iteration** refers to the former iteration $t_j, 0 \leq j \leq i-2$, excluding the previous $(i-1)$-th iteration, during the procedure of sample selection. Similarly, **descendant iteration** denotes the future iteration $t_s, i+1 \leq s \leq N_{\text{iterations}}$. $N_{\text{iterations}}$ is the total number of training iterations. The ancestor iteration and descendant iteration will not appear in the same epoch. In our paradigm, the current network in the i-th iteration is trained with clean samples selected only by the network from an ancestor iteration t_j. This behavior of sample selection exhibits a *jump* manner. Concretely, we leverage a binary identifier to represent the outcome of the label judgment after sample selection. Thus, a binary identifier table $\mathcal{I}$ corresponds to the entire dataset. The jump-update strategy is divided into three steps: 1) In the current iteration t_i, the network generates the new binary identifiers by the clean sample selection for the descendant iteration t_s. 2) The network updates the parameters based on the clean data judged only by the old table. Before the update, the network inherits the weights from the previous iteration t_{i-1}. 3) We update the table with identifiers cached in step 1.

The jump-update strategy harnesses the disagreement between different training iterations, enabling more effective bias correction due to a larger disagreement than the cross-update strategy. Suppose S denotes the jump steps, it should be noted that $2 \leq S \leq N_{\text{iterations}}$. The setting of S is illustrated in Section 4.3.

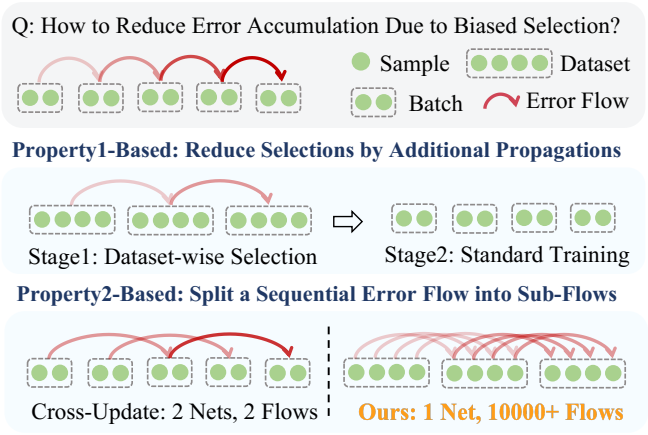


Figure 2: Error flows in Property 1-based and Property 2-based methods. The jump-update strategy reduces error accumulation without additional propagations or networks.

Theoretical Analysis. The bias introduces noise in the training data, leading to the accumulated error in updated parameters. Therefore, a core question in mitigating bias is: *How can we reduce error accumulation caused by biased selection?* To answer this question, we utilize error flow to quantitatively represent the process of error accumulation, inspired by (Han et al. 2018). As shown in Fig. 2, the graph of error flow presents a sequential form by iterations. Existing debiasing works weaken error flow through two properties: 1) reducing iterations or 2) splitting it into multiple sub-flows. Property 1-based methods require additional forward propagations as each iteration involves both selection operation and model update, while standard training procedures necessitate mini-batch updates. This constraint requires dataset-wise selection to minimize iterations. Previous property 2-based methods are built on the cross-update strategy, which split the flow

into two sub-flows, as clearly demonstrated by the different colors in Figure 1(a). Our jump-update strategy, however, limits the error accumulating orthogonally on different samples, extending the number of sub-flows to the dataset size. We formalize the aforementioned properties mathematically below before giving some notations.

Notations. Suppose that N_A is the total number of error accumulations, N_a is the number of error accumulations in an error sub-flow, and N_f is the number of error sub-flows. Besides, the constant e represents the number of training epochs and n represents how many selections are made in one epoch. D_A denotes the overall degree of accumulated error, while d_a^k denotes the degree of accumulated error in the k-th error sub-flow. In the absence of a specific reference, we can also denote the degree of accumulated error in one error sub-flow by d_a.

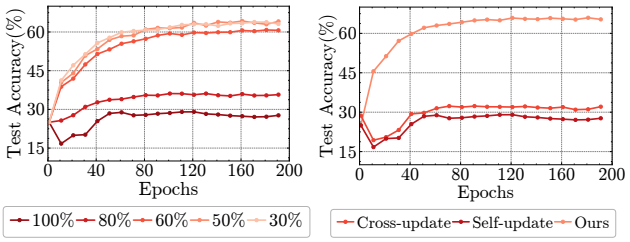
Property 1 *The overall degree of accumulated error D_A is proportional to the total number of error accumulation N_A, $D_A \propto N_A$. Under the hypothesis that the error flow is an uninterrupted model, the number of error accumulation N_A equals the total number of training iterations $N_{iterations}$, while $N_{iterations}$ equals $e \times n$. Therefore, $D_A \propto n$.*

The overall degree of accumulated error D_A depends on n stated in Property 1. When the network selects data from a mini-batch, D_A can be enormous because n is equal to the number of mini-batches, *e.g.,* Co-teaching. If the network selects data from the entire dataset, n can be reduced to 1, thereby significantly reducing D_A, *e.g.,* TopoFilter (Wu et al. 2020) and LateStopping (Yuan, Feng, and Liu 2023). However, Property 1-based methods are inefficient because an additional forward pass over the entire dataset before training is necessitated.

Property 2 *The accumulated error could be reduced by splitting the error flow into multiple error sub-flows. The degree of accumulated error in the k-th error sub-flow d_a^k is proportional to the number of error accumulations in each error sub-flow N_a, $d_a^k \propto N_a$. The number of error accumulations in each error flow N_a equals $\frac{N_A}{N_f}$.*

As stated in Property 1, D_A can be reduced by minimizing n, while D_A can also be reduced by splitting into error sub-flows as illustrated in Property 2. The first property relates to the sample selection mechanism, and the second property is associated with the different strategies of model updates: 1) The self-update strategy follows a single error flow, resulting in $d_a = D_A$, which leads to rapid error accumulation. 2) The cross-update strategy has two error sub-flows and $d_a = 0.5D_A$, which mitigates the error accumulation to some extent. 3) The jump-update strategy reduces d_a to $0.5e$, which is a significantly minimal value.

Experimental Analysis. We verify Property 1 and Property 2 with toy examples, respectively. We employ the small-loss sample selection method from Co-teaching to establish the baselines for self-update and cross-update. Our experiments choose *CIFAR-10* (Krizhevsky, Hinton et al. 2009) dataset with the symmetric noise ratio $\epsilon = 80\%$. We utilize ResNet-18 (He et al. 2016) as the backbone and warm up it for one epoch before formal training.



(a) D_A with different n (b) D_A with different strategies

Figure 3: Test accuracies on *CIFAR-10* with Sym. $\epsilon = 0.8$.

To verify Property 1, we observe D_A by testing the accuracy of the network with different values of n.

To control n, We set r as the proportion of sample selection that takes effects, D_A equals $r \times n$ in this way. We set r to 30%, 50%, 60%, 80%, and 100%. As shown in Figure 3(a), rapidly accumulated errors lead to extreme deterioration of the model such as when $r = 100\%$ and $r = 80\%$, while slower accumulated ones achieve better performance, with a moderate $r = 50\%$ yielding the best results. This is consistent with Property 1. Our experiments not only verify Property 1, but also reveal that simply setting r offers an effective and cost-free alternative to mitigate the bias.

To verify Property 2, we observe D_A by testing the accuracy of the neural network with three strategies: self-update, cross-update, and jump-update. As shown in Figure 3(b), cross-update slightly outperforms self-update, while jump-update is significantly more effective than both. Thus, splitting flows is an effective way to reduce error accumulation.

Remark. The jump-update strategy is more effective and efficient than previous works. Compared to methods that implicitly leverage Property 1, it can reduce D_a to a small number *i.e.,* $0.5e$ at a constant cost. Compared to the cross-update strategy that leverages Property 2, it can not only reduce the degree of accumulated error significantly more but also halve the training cost.

3.2 Single-loss Criterion

Due to the redundancy in the current sample selection criteria, we propose the single-loss criterion for sample-wise selection. Our design tackles two pivotal questions:

Q1. How to overcome the information insufficiency caused by relying solely on a single loss value?

Q2. How to design a more effective selection criterion with the information enriched by addressing Q1?

To address the first question, we propose analyzing the distribution within the loss across different semantic dimensions, drawing inspiration from prior work in representation learning (Yang et al. 2015; Liu et al. 2016; Yuan et al. 2020). In implementation, we design a lightweight plugin to decompose the single loss for a detailed distribution of sub-losses in different semantics. The plugin is structurally composed of a pre-prepared non-orthogonal codebook and an auxiliary head at the last layer of the network. It spans an auxiliary space where a loss can be decomposed semantically. Specifically, the codebook transforms the label into hash encodings while the head maps outputs into feature embedding. We will first

detail the codebook and auxiliary head respectively, and then introduce the decomposed loss function by employing both.

Codebook. Inspired by Yang et al. (2015), we utilize the favorable properties of Hadamard matrices to construct mappings for category encoding. A K-bit Hadamard matrix can generate $2K$ codewords, each K bits long, with a minimum Hamming distance of $\frac{K}{2}$. For K-bit hash encodings, we construct a $K \times K$ Hadamard matrix. From this, we select c row vectors as category encodings, each with a Hamming distance of $\frac{K}{2}$. Noisy labels $\tilde{y}$ are mapped into hash encodings $\tilde{y}'$ through this codebook. Given a classification task with C classes, the mapping is formalized as:

$$H : \tilde{y}_i \in \{0, \cdots, C-1\} \rightarrow \tilde{y}'_i \in \{-1, 1\}^K.$$

Auxiliary Head. The auxiliary detection head is an additional three-layer MLP with a Tanh activation function. It shares the same feature extractor with the original classification head and maps the outputs $\boldsymbol{x}$ of neural networks to K-bit feature embeddings $\boldsymbol{z}$, as represented by the function:

$$f : x_i \in \mathbb{R}^n \rightarrow z_i \in \mathbb{R}^K.$$

For an ideally clean sample, the distance $\mathbf{d}(\mathbf{z}_i^{(t)}, \mathbf{y}_i)$ between the output $\mathbf{z}_t$ of an ideally clean sample indexed i at time t and its label $\mathbf{y}$ can be expressed as:

$$\lim_{t \to \infty} \mathbf{d}(\mathbf{z}_i^{(t)}, \mathbf{y}_i) = \mathbf{0}, \tag{1}$$

where the notation $\mathbf{0}$ denotes a vector of zeros, indicating that the distances across different dimensions uniformly converge towards zero as t approaches infinity.

We employ binary cross-entropy (Ruby and Yendapalli 2020) to define the distance $\mathbf{d}$ between feature embeddings and hash encodings. It indicates how well the network captures the semantics of each dimension, formulated as:

$$\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i) = -\left[\tilde{\mathbf{y}}'_i \odot \log(\mathbf{z}_i) + (1 - \tilde{\mathbf{y}}'_i) \odot \log(1 - \mathbf{z}_i)\right], \tag{2}$$

where the distance vector $\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i)$ obtained through decomposition directly describes the distribution of sample loss in semantic space, which addresses the first question.

Moreover, to supervise the detection head, we apply the arithmetic mean on the distance vector $\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i)$. Therefore, the loss function can be articulated as:

$$\mathcal{L}_i^{BCE} = -\frac{1}{K} \sum_{j=1}^{K} \left[\tilde{y}'_{ij} \log(z_{ij}) + (1 - \tilde{y}'_{ij}) \log(1 - z_{ij})\right]. \tag{3}$$

Selection Criterion. To address the second question, we consider a sample with a distorted intra-loss distribution to have a noisy label. Our principle comes from memorization effects that neural networks prioritize learning simpler patterns from data (Zhang et al. 2021). The small-loss criterion leverages this effect: clean labels are learned first by the network, hence exhibiting smaller losses compared to noisy ones. However, the relative magnitude of losses is determined by comparison with other samples *e.g.,* rank sample losses with Top-k algorithm (Han et al. 2018; Jiang et al. 2018) or modeling loss distribution (Li, Socher, and Hoi 2020; Permuter, Francos, and Jermyn 2006). The single-loss criterion avoids such costly overheads by leveraging memorization effects in intra-loss distribution where the clean dimensions will be learned first and incur smaller sub-losses, while the noisy dimensions result in larger sub-losses. Therefore, a sample with a distorted intra-loss distribution is more likely to have noisy labels. To identify whether a label contains noise, we use variance to characterize the distortion:

Datasets	CIFAR-10		CIFAR-100	
Criterion	Small-loss	Single-loss	Small-loss	Single-loss
w/o Jump-update	59.94	72.70	35.25	48.64
w/ Jump-update	69.26	84.93	36.15	51.11

Table 1: Ablation study on different criteria with symmetric noise $\epsilon = 0.5$. We utilize ResNet18 as our backbone.

$$\begin{aligned} \mathrm{Var}\big(\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i)\big) = \frac{1}{K}\Big(\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i) - \mathcal{L}_i^{BCE}\mathbf{1}\Big)^T \\ \times \Big(\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i) - \mathcal{L}_i^{BCE}\mathbf{1}\Big). \end{aligned} \tag{4}$$

We set a unified threshold to distinguish clean samples from noisy samples. Since this threshold is expected to approach zero infinitely, we set it to 0.001. The identifier $\mathcal{I}_{\text{detection}}$ is updated by the detection head as follows:

$$\mathcal{I}_{\text{detection}} = \begin{cases} \text{True} & \text{if } \mathrm{Var}(\mathbf{d}(\mathbf{z}_i, \tilde{\mathbf{y}}'_i)) \leq \tau, \\ \text{False} & \text{otherwise.} \end{cases} \tag{5}$$

3.3 Training Pipeline

In this section, we illustrate how the proposed plugin can assist in selection. During training, the detection head is trained by Eq. 3 while the classification head is trained by the cross-entropy loss. During inference, only the classification head takes effect. However, the two heads converge at markedly different rates during training, causing the network to prematurely overfit noisy labels. Specifically, the cross-entropy loss used for the classification head is easier to optimize and converges much faster than the binary cross-entropy loss of the detection head. Consequently, errors accumulate before the detection head is sufficiently trained to perform effective selection.

To balance the convergence rate of the two heads, we employ temperature scaling (Guo et al. 2017) to calibrate the label probabilities $\boldsymbol{p}$ of the classification head. Thus, the soften softmax function is modified as follows:

$$\sigma_{\text{softmax}}(p_{ij}) = \frac{\exp(p_{ij}/T)}{\sum_{j=1} \exp(p_{ij}/T)}, \tag{6}$$

where T is a temperature scaling factor, controlling the convergence rates of the head.

To make use of the classifier head, we follow the widely accepted principle that, assuming the model is well-trained, predictions of clean samples should align with true labels (Xiao et al. 2023). Based on this principle, we apply a straightforward criterion, which further recovers the discarded clean labels. The clean table of the classifier head is evaluated as:

$$\mathcal{I}_{\text{classifier}} = (\hat{y}_i == \tilde{y}_i), \tag{7}$$

Dataset	CIFAR-10				CIFAR-100			
Noise Type	Sym.			Asym.	Sym.			Asym.
Noise ratio	0.2	0.5	0.8	0.4	0.2	0.5	0.8	0.4
Standard	84.6±0.1	62.4±0.3	27.3±0.3	75.9±0.4	56.1±0.1	33.6±0.2	8.2±0.1	40.1±0.2
Decoupling('17)	86.4±0.1	72.9±0.2	48.4±0.6	83.3±0.2	53.3±0.1	28.0±0.1	7.9±0.1	39.9±0.4
Co-teaching('18)	89.9±0.6	67.3±4.2	28.1±2.0	79.2±0.5	61.8±0.4	34.7±0.5	7.5±0.5	40.0±1.2
Co-teaching+('19)	88.1±0.0	61.8±0.2	22.3±0.6	58.2±0.2	54.5±0.1	27.6±0.1	8.4±0.1	19.9±0.3
PENCIL('19)	88.2±0.6	73.4±1.5	36.0±0.7	77.3±3.4	57.4±1.0	11.4±3.2	5.4±1.2	45.7±0.9
Topofilter('20)	89.5±0.1	84.6±0.2	45.9±2.6	89.9±0.1	63.9±0.8	51.9±1.7	16.9±0.3	66.6±0.7
FINE('21)	90.2±0.1	85.8±0.8	70.8±1.8	87.8±0.1	70.1±0.3	57.9±1.2	22.2±0.7	53.5±0.8
SPRL('23)	91.7±0.2	88.4±0.6	63.9±1.4	89.8±0.5	69.5±0.8	57.2±1.6	23.8±2.2	58.7±1.3
LateStopping('23)	91.1±0.1	66.0±0.7	32.3±1.9	82.7±0.3	69.4±0.2	45.4±0.3	12.8±0.2	55.4±0.3
RML('24)	92.2±0.1	88.3±0.3	35.3±1.7	79.8±4.0	67.2±0.2	62.0±0.5	15.7±0.8	64.5±0.6
Jump-teaching	**94.8±0.1**	**92.2±0.1**	**84.1±1.1**	**90.7±0.3**	**72.7±0.5**	**67.1±0.2**	**40.0±1.1**	**68.4±0.7**

Table 2: Average test accuracy(%) on *CIFAR-10* and *CIFAR-100* with symmetric and asymmetric noise. The mean and standard deviation over three trials are presented. All methods employ PreActResNet-18 and train 200 epochs. The best results are highlighted in bold.

where $\hat{y}_i = \arg\max_j p_i^j$ is the prediction label and p_i^j represents the probability of the j-th class for the i-th sample. $\tilde{y}_i$ denotes the label of the i-th sample. Finally, we can update the table by combining Eq. 5 and Eq. 7:

$$\mathcal{I}' = \mathcal{I}_{\text{detection}} \vee \mathcal{I}_{\text{classifier}}. \tag{8}$$

To validate the single-loss criterion, we compare it with the cross-update criterion under various settings. As shown in Table 1, the single-loss criterion achieves over a 10% improvement compared to the cross-update strategy, highlighting its effectiveness.

4 Experiments

In this section, we first describe the experimental settings in Section 4.1. Then, we demonstrate the effectiveness of Jump-teaching, compared with the state-of-the-art in Section 4.2. As the jump-update strategy is the key component of our method, we thoroughly examine it in Section 4.3.

4.1 Experimental Setup

Noisy Benchmark Datasets. We verify the experiments on three benchmark datasets, including *CIFAR-10*, *CIFAR-100* and *Clothing1M* (Xiao et al. 2015). These datasets are popular for evaluating noisy labels. Following the setup on (Li, Socher, and Hoi 2020; Liu et al. 2020), we simulate two types of label noise: symmetric noise, where a certain proportion of labels are uniformly flipped across all classes, and asymmetric noise, where labels are flipped to specific classes, *e.g.,* $bird \rightarrow airplane$, $cat \leftrightarrow dog$. We assume ϵ denotes the noise ratio.

Baselines. To be more convincing, we compare the competitive methods of LNL. These methods are as follows: Standard, which is simply the standard deep network trained on noisy datasets, Decoupling (Malach and Shalev-Shwartz 2017), Co-teaching (Han et al. 2018), Co-teaching+ (Yu et al. 2019), PENCIL (Yi and Wu 2019), TopoFilter (Wu et al. 2020), ELR (Liu et al. 2020), FINE (Kim et al. 2021), SPRL (Shi et al. 2023), LateStopping (Yuan, Feng, and Liu 2023), RML (Li et al. 2024), APL (Ma et al. 2020), CDR (Xia et al. 2021), MentorNet (Jiang et al. 2018), SIGUA (Han et al. 2020), JoCoR (Wei et al. 2020) and CoDis (Xia et al. 2023).

Implementation Details. All experiments operate on a server equipped with an NVIDIA A800 GPU and PyTorch platform. In the following experiments, Jump-teaching *almost* employs the same configuration. It trains the network for 200 epochs by SGD with a momentum of 0.9, a weight decay of $1e-3$, and a batch size of 128. The initial learning rate is set to 0.2, and a cosine annealing scheduler finally decreases the rate to $5e-4$. The warm-up strategy is utilized by Jump-teaching, and the warm-up period is 30 epochs. Exceptionally, we set the weight decay as $5e-4$ to facilitate learning on fewer available samples when the noise ratio ϵ equals 50% and 80% in *CIFAR-100*, respectively. The single network of Jump-teaching employs three types of backbone networks. The baseline methods fully follow the experimental setup in the literature (Han et al. 2018; Li, Socher, and Hoi 2020).

4.2 Comparison with the State-of-the-Arts

Synthetic Noisy Benchmark. To verify the effectiveness of Jump-teaching, we compare our proposed method with a range of representative sample selection approaches. As shown in Table 2, Jump-teaching demonstrates superior performance with all noise settings, confirmed as statistically significant ($p < 0.05$) by a Wilcoxon signed-rank test. Its superiority is particularly evident under extreme noise, where it achieves accuracy improvements of 13.3% and 16.2% on two datasets with a symmetric noise ratio of $\epsilon = 0.8$.

Real-World Noisy Benchmark. We operate real-world experiments including the *Clothing1M*, *Food-101*, *WebVision*, and *ImageNet ILSVRC12* datasets. Following the setup in Xia et al. (2023), we employ ResNet-50 pre-trained on *ImageNet*

Dataset	Metric	APL	CDR	MentorNet	SIGUA	Decoupling	Co-teaching	Co-teaching+	JoCoR	CoDis	Jump-teaching
Clothing1M	Top-1	54.46	66.59	67.25	65.37	67.65	67.94	63.83	69.06	71.60	**71.93**
Food-101	Top-1	82.17	86.36	81.25	79.68	83.73	78.88	76.89	84.04	86.13	**86.67**
WebVision	Top-1	62.30	62.84	63.00	57.38	62.54	63.58	61.18	63.33	63.80	**75.60**
	Top-5	84.02	84.11	81.40	78.92	84.74	85.20	83.30	85.06	85.54	**90.41**
ILSVRC12	Top-1	61.27	61.85	57.66	52.88	57.26	61.22	58.74	58.76	62.29	**71.66**
	Top-5	84.82	85.80	80.01	74.67	80.50	84.78	82.72	82.85	85.39	**90.15**

Table 3: Test accuracy (%) on real-world noisy benchmarks.

Dataset	CIFAR-10					CIFAR-100			
Noise type	Sym.				Asym.	Sym.			
Methods/Noise ratio	0.2	0.5	0.8	0.9	0.4	0.2	0.5	0.8	0.9
DivideMix	95.7	94.4	92.9	75.4	**92.1**	76.9	**74.2**	59.6	31.0
J-DivideMix	**96.0**	**94.6**	**93.2**	**77.7**	91.9	**77.3**	**74.2**	**59.7**	**32.5**
DivideMix with θ_1 test	95.0	93.7	92.4	74.2	91.4	74.8	72.1	57.6	29.2
J-DivideMix with θ_1 test	**95.8**	**94.8**	**92.7**	**79.4**	**92.2**	**77.3**	**73.2**	**57.8**	**30.8**
DivideMix w/o co-training	**94.8**	93.3	92.2	73.2	90.6	74.1	71.7	**56.3**	27.2
J-DivideMix w/o co-training	94.7	**94.0**	**92.3**	**79.3**	**91.3**	**74.2**	**71.7**	55.3	**27.6**

Table 4: Comparison of test accuracies(%) for DivideMix and J-DivideMix on *CIFAR-10* and *CIFAR-100* with different noise. The mean value of the last ten epochs are reported.

Method	Single Net	Throughput ↑	Peak Mem ↓
Standard	✓	4.16	0.68
Decoupling	✗	1.72	1.50
Co-teaching	✗	1.81	1.53
Co-teaching+	✗	2.63	1.53
PENCIL	✓	3.13	0.73
Topofilter	✓	1.61	1.40
FINE	✓	2.17	0.94
SPRL	✓	2.94	0.77
RML	✓	0.91	0.81
Jump-teaching	✓	**4.07**	**0.70**

Table 5: Comparison of training throughput (in kfps) and peak memory consumption (in GB) across methods.

(Deng et al. 2009) dataset for *Clothing1M* and *Food-101*, and InceptionV2 for *Web-Vision*, and *ILSVRC12*. As shown in Table 3, the robustness of Jump-teaching for real-world noisy labels is favorable when compared to other methods.

Efficiency Analysis. We compare the efficiency of Jump-teaching with the above representative methods. The efficiency of these methods is evaluated by throughput and peak memory usage. Throughput, expressed in *thousands of frames per second* (kfps), measures the training speed. It is calculated as the total number of training samples divided by the average time per epoch. Peak memory usage refers to the maximum amount of memory consumed during training. We evaluate these methods with symmetric noise ratio $\epsilon = 0.5$ in Table 5, while the accuracy of these methods is illustrated in Table 2. As shown in Table 2 and Table 5, our method achieves almost up to $4.47\times$ training speedup, 54% peak memory footprint reduction, and superior robustness overall.

4.3 Experiments on Jump-update Strategy

The jump-update strategy is not only suitable for Jump-teaching but also easily integrated into other methods in LNL. DivideMix not only uses two networks to exchange error flow but also employs them to create pseudo-labels collaboratively. Consequently, we establish three baselines following Li, Socher, and Hoi (2020): DivideMix, DivideMix with θ_1 test, and DivideMix w/o co-training. DivideMix with θ_1 test utilizes a single network for sample selection, while DivideMix w/o co-training retains a single sample and employs self-update for updating. In each case, we replace the original update strategy with the jump-update strategy. As Table 4 shows, the jump-update strategy significantly improves the accuracy of all baselines, particularly under extreme noise ratios, indicating its effectiveness in combating selection bias.

5 Conclusion

In this paper, we propose Jump-teaching, an efficient sample selection framework to combat label noise. Our work explores the sample-selection bias mechanism and effectively mitigates the bias in a single network by leveraging temporal disagreement, especially under extreme noise. Moreover, with the proposed single-loss criterion, Jump-teaching achieves sample-wise sample selection, fully unleashing the power of the jump-update strategy. Extensive experimental validations confirm that Jump-teaching achieves SOTA performance in both robustness and efficiency.

Acknowledgments

We would like to thank Suqin Yuan for the insightful discussions and constructive suggestions. The work was supported by the Fundamental Research Funds for the Central Universities (No. XJSJ25005), Ministry of Education Top-notch Student Training Program in Basic Disciplines 2.0 Research Topics (No. 20252012), the Natural Science Basis Research Plan in Shaanxi Province of China (No. 2025JC-JCQN-089) and the Outstanding Youth Science Foundation of Shaanxi Province under Grant 2025JC-JCQN-083. We also acknowledge the support from the National Experimental Teaching Demonstration Center for Computer Network and Information Security affiliated with Xidian University.

References

Bai, Y.; and Liu, T. 2021. Me-momentum: Extracting hard confident examples from noisily labeled data. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9312–9321.

Bakhshi, S.; and Can, F. 2024. Balancing efficiency vs. effectiveness and providing missing label robustness in multi-label stream classification. *Knowledge-Based Systems*, 111489.

Blum, A.; Kalai, A.; and Wasserman, H. 2003. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4): 506–519.

Cao, K.; Chen, Y.; Lu, J.; Arechiga, N.; Gaidon, A.; and Ma, T. 2021. Heteroskedastic and imbalanced deep learning with adaptive regularization. In *International Conference on Learning Representations*.

Chen, M.; Cheng, H.; Du, Y.; Xu, M.; Jiang, W.; and Wang, C. 2023. Two wrongs don't make a right: Combating confirmation bias in learning with label noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 14765–14773.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Ghosh, A.; Kumar, H.; and Sastry, P. S. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Goldberger, J.; and Ben-Reuven, E. 2022. Training deep neural-networks using a noise adaptation layer. In *International conference on learning representations*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *International conference on machine learning*, 1321–1330. PMLR.

Han, B.; Niu, G.; Yu, X.; Yao, Q.; Xu, M.; Tsang, I.; and Sugiyama, M. 2020. Sigua: Forgetting may make learning with noisy labels more robust. In *International Conference on Machine Learning*, 4006–4016. PMLR.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31.

Han, J.; Luo, P.; and Wang, X. 2019. Deep Self-Learning From Noisy Labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, Z.; Zhang, J.; and Shan, H. 2023. Twin contrastive learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11661–11670.

Iscen, A.; Valmadre, J.; Arnab, A.; and Schmid, C. 2022. Learning With Neighbor Consistency for Noisy Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4672–4681.

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, 2304–2313. PMLR.

Kim, T.; Ko, J.; Choi, J.; Yun, S.-Y.; et al. 2021. Fine samples for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34: 24137–24149.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, F.; Li, K.; Tian, J.; and Zhou, J. 2024. Regroup Median Loss for Combating Label Noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 13474–13482.

Li, J.; Socher, R.; and Hoi, S. C. 2020. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. In *International Conference on Learning Representations*.

Liu, H.; Wang, R.; Shan, S.; and Chen, X. 2016. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2064–2072.

Liu, S.; Niles-Weed, J.; Razavian, N.; and Fernandez-Granda, C. 2020. Early-learning regularization prevents memorization of noisy labels. *Advances in neural information processing systems*, 33: 20331–20342.

Liu, S.; Zhu, Z.; Qu, Q.; and You, C. 2022. Robust training under label noise by over-parameterization. In *International Conference on Machine Learning*, 14153–14172. PMLR.

Lu, Y.; and He, W. 2022. SELC: self-ensemble label correction improves learning with noisy labels. *arXiv preprint arXiv:2205.01156*.

Ma, X.; Huang, H.; Wang, Y.; Romano, S.; Erfani, S.; and Bailey, J. 2020. Normalized Loss Functions for Deep Learning with Noisy Labels. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 6543–6553. PMLR.

Mahajan, D.; Girshick, R.; Ramanathan, V.; He, K.; Paluri, M.; Li, Y.; Bharambe, A.; and Van Der Maaten, L. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, 181–196.

Malach, E.; and Shalev-Shwartz, S. 2017. Decoupling" when to update" from" how to update". *Advances in neural information processing systems*, 30.

Permuter, H.; Francos, J.; and Jermyn, I. 2006. A study of Gaussian mixture models of color and texture features for image classification and segmentation. *Pattern recognition*, 39(4): 695–706.

Pham, H.; Dai, Z.; Xie, Q.; and Le, Q. V. 2021. Meta pseudo labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11557–11568.

Ruby, U.; and Yendapalli, V. 2020. Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10).

Shi, X.; Guo, Z.; Li, K.; Liang, Y.; and Zhu, X. 2023. Self-paced resistance learning against overfitting on noisy labels. *Pattern Recognition*, 134: 109080.

Sohn, K.; Berthelot, D.; Carlini, N.; Zhang, Z.; Zhang, H.; Raffel, C. A.; Cubuk, E. D.; Kurakin, A.; and Li, C.-L. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33: 596–608.

Song, H.; Kim, M.; and Lee, J.-G. 2019. Selfie: Refurbishing unclean samples for robust deep learning. In *International conference on machine learning*, 5907–5915. PMLR.

Tanaka, D.; Ikami, D.; Yamasaki, T.; and Aizawa, K. 2018. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5552–5560.

Toneva, M.; Sordoni, A.; Combes, R. T. d.; Trischler, A.; Bengio, Y.; and Gordon, G. J. 2018. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*.

Wang, Y.; Ma, X.; Chen, Z.; Luo, Y.; Yi, J.; and Bailey, J. 2019. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF international conference on computer vision*, 322–330.

Wei, H.; Feng, L.; Chen, X.; and An, B. 2020. Combating Noisy Labels by Agreement: A Joint Training Method with Co-Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wei, Q.; Sun, H.; Lu, X.; and Yin, Y. 2022. Self-filtering: A noise-aware sample selection for label noise with confidence penalization. In *European Conference on Computer Vision*, 516–532. Springer.

Welinder, P.; Branson, S.; Perona, P.; and Belongie, S. 2010. The multidimensional wisdom of crowds. *Advances in neural information processing systems*, 23.

Wu, P.; Zheng, S.; Goswami, M.; Metaxas, D.; and Chen, C. 2020. A topological filter for learning with label noise. *Advances in neural information processing systems*, 33: 21382–21393.

Xia, X.; Han, B.; Zhan, Y.; Yu, J.; Gong, M.; Gong, C.; and Liu, T. 2023. Combating noisy labels with sample selection by mining high-discrepancy examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1833–1843.

Xia, X.; Liu, T.; Han, B.; Gong, C.; Wang, N.; Ge, Z.; and Chang, Y. 2021. Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*.

Xiao, R.; Dong, Y.; Wang, H.; Feng, L.; Wu, R.; Chen, G.; and Zhao, J. 2022. Promix: Combating label noise via maximizing clean sample utility. *arXiv preprint arXiv:2207.10276*.

Xiao, R.; Dong, Y.; Wang, H.; Feng, L.; Wu, R.; Chen, G.; and Zhao, J. 2023. ProMix: Combating Label Noise via Maximizing Clean Sample Utility. In Elkind, E., ed., *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 4442–4450. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Xiao, T.; Xia, T.; Yang, Y.; Huang, C.; and Wang, X. 2015. Learning From Massive Noisy Labeled Data for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, S.; Luo, P.; Loy, C. C.; Shum, K. W.; and Tang, X. 2015. Deep representation learning with target coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Yi, K.; and Wu, J. 2019. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7017–7025.

Yu, X.; Han, B.; Yao, J.; Niu, G.; Tsang, I.; and Sugiyama, M. 2019. How does disagreement help generalization against label corruption? In *International conference on machine learning*, 7164–7173. PMLR.

Yuan, L.; Wang, T.; Zhang, X.; Tay, F. E.; Jie, Z.; Liu, W.; and Feng, J. 2020. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3083–3092.

Yuan, S.; Feng, L.; Han, B.; and Liu, T. 2025. Enhancing Sample Selection by Cutting Mislabeled Easy Examples. *arXiv preprint arXiv:2502.08227*.

Yuan, S.; Feng, L.; and Liu, T. 2023. Late stopping: Avoiding confidently learning from mislabeled examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16079–16088.

Yuan, S.; Feng, L.; and Liu, T. 2025. Early stopping against label noise without validation data. *arXiv preprint arXiv:2502.07551*.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115.

Zhang, Z.; Zhang, H.; Arik, S. O.; Lee, H.; and Pfister, T. 2020. Distilling Effective Supervision From Severe Label Noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.