Robust Self-Training with Closed-loop Label Correction for Learning from Noisy Labels

Zhanhui Lin **Yanlin Liu** **Sanping Zhou**

Abstract

Training deep neural networks on noisy labels is a significant challenge, often leading to degraded performance. While existing methods tackle label noise, they often struggle with high computational costs and inefficient use of noisy data. In this paper, we introduce a novel data purification paradigm, where a lightweight correction model, guided by a minimal set of clean samples, purifies a massive noisy dataset for a large-scale classifier. We realize this through a self-training framework using decoupled bilevel optimization, allowing a classifier and a neural correction function to co-evolve synergistically. Our key innovation lies in achieving both high expressiveness to model complex noise and guaranteed stability, even when clean data is scarce. By simulating noisy posteriors and using intermediate features, our closed-loop system transfers ground-truth knowledge and prevents error amplification, a claim supported by theoretical guarantees. Extensive experiments on CIFAR and Clothing1M confirm state-of-the-art performance with significantly reduced training time, highlighting the framework's practical value for large-scale learning with noisy labels.

1 Introduction

The remarkable success of deep neural networks (DNNs) hinges on large-scale datasets, but obtaining clean data is often prohibitively expensive. Training on noisy labels leads DNNs to memorize errors, degrading performance and generalization [30, 16]. This phenomenon has spurred extensive research into robust training methods that can effectively handle noisy labels, giving rise to the field of learning with noisy labels (LNL).

Although fully clean data sets are scarce, accurately annotating a small subset of data remains feasible, which can significantly enhance model performance. Transition matrix-based methods attempt to estimate the mapping between noisy and clean labels using clean data, but these approaches struggle with instance-dependent noise patterns [27, 36] and suffer from high variance that affects neural network learning. Meta-learning approaches address this challenge by optimizing hyperparameters on small-scale clean data through bilevel optimization. However, their inner-loop virtual updates significantly increase memory usage and computational cost [21, 31], limiting their application to large-scale scenarios. Noise detection techniques use information in the clean data to down-weight or discard samples [14, 29], but these approaches are typically multi-stage, time-consuming, and underutilize noisy samples through weighting or discarding.

These limitations highlight a critical challenge: how to effectively and efficiently use a massive, noisy dataset with guidance from only a handful of clean examples. To address this, we propose a new, scalable purification paradigm through self-training with decoupled bilevel optimization, where the main classifier and a lightweight neural correction function co-evolve synergistically. This creates a closed-loop system that solves the crucial trade-off between expressiveness and stability: our neural corrector is expressive enough to model complex, instance-dependent noise, while our robust convex combination strategy provides a theoretical guarantee that prevents error amplification, ensuring stability even when the clean dataset is extremely limited. Unlike existing methods that rely

Preprint. Under review.

on final output pseudo-labels, we leverage intermediate feature representations that contain richer information about the underlying data. We innovatively apply noisy posterior simulation to bridge the clean and noisy data distributions, enabling the transfer of ground-truth knowledge from the clean dataset to the correction of noisy labels through the correction function. This ensures that correction optimizes in the correct direction while avoiding error amplification. As the classifier trains on higher-quality corrected labels, its feature extractor produces features that increasingly approximate the true semantic distribution of the data, further enhancing the correction function's performance.

Our approach offers several key advantages over existing methods. It explicitly uses clean data as feedback during training, avoiding cumulative error amplification and complex threshold selection issues inherent in traditional label correction methods. Unlike weighting or noise detection approaches, our method corrects rather than down-weights or discards noisy labels, achieving higher utilization of noisy data while reducing variance. Moreover, compared to state-of-the-art meta-learning methods, our approach achieves superior performance with lower computational complexity, better distributed training capability, and robustness in data-scarce scenarios.

Contributions Our main contributions are three-fold. First, we propose a novel and scalable purification framework where a lightweight correction function and a main classifier co-optimize synergistically, using clean labels to guide the self-rectification on incorrect labels and prevent drift from facts (ground truth). Second, we introduce a mechanism that uniquely balances expressiveness with stability, with a theoretical guarantee ensuring that label corrections do not increase risk, allowing an expressive neural network to be used for correction without succumbing to error amplification caused by overfitting. Third, we demonstrate through experiments on CIFAR and Clothing1M that our method achieves state-of-the-art performance while reducing training time, validating its practical value in the large-scale scenario.

2 Related Work

Various Methods for Noisy Label Learning Learning with noisy labels (LNL) is a significant challenge as deep neural networks (DNNs) can memorize erroneous labels, hindering generalization [1, 8]. One major category of LNL methods involves *loss-based strategies*. This includes designing robust loss functions like Generalized Cross-Entropy (GCE) [32] or estimating a noise transition matrix (NTM) to correct the loss [23]. Other approaches directly *refurbish noisy labels*; for example, Robust Logit Adjustment (RLA) dynamically forms corrected labels using model predictions and noisy labels [11]. *Sample selection* techniques aim to identify and utilize presumed clean samples, often exploiting the early-learning phenomenon of DNNs [9]. Co-teaching [3] trains two networks that select small-loss samples for each other, while DivideMix [7] uses Gaussian Mixture Models on per-sample losses to separate clean and noisy data, then applies semi-supervised techniques. *Regularization* is also crucial. Consistency regularization, adapted from fields like partial label learning [22], and advanced data augmentation such as MixUp [7], are widely used. More recently, *generative and diffusion models* have shown promise. Label-Retrieval-Augmented (LRA) Diffusion Models generate pseudo-clean labels through iterative refinement [26], while Transition-aware weighted Denoising Score Matching (TDSM) aims to make conditional diffusion models robust by estimating noise transitions [2].

Noisy Label Learning with a Limited Clean Dataset The availability of a small, trusted clean dataset can substantially bolster LNL strategies. Many methods treat this as a *semi-supervised learning (SSL)* problem, where clean data acts as the labeled set and noisy data as weakly labeled or unlabeled [17]. For instance, ZMT [35] leverages a multi-task architecture where clean data guides the primary SSL task. *Meta-learning* often utilizes the clean dataset as a meta-validation set. Some methods [10, 15, 31] employ sample re-weighting strategies to mitigate the impact of noisy labels on the training loss, and other approaches [34, 33] meta-learn optimal weights for pseudo-labels in a bootstrapping manner. For few-shot learning (FSL) with noisy labels, DCML [20] uses the clean few-shot data to guide curriculum learning from noisy data. The clean subset can also facilitate more *direct noise estimation and correction*. For example, clean data is pivotal for estimating the noise transition matrix (NTM) that models noisy-to-clean label flips [4, 6]; this approach often employs forward loss correction [12, 24] and has been extended to multi-label scenarios [13]. Furthermore, clean data can provide anchors for label refurbishment in methods like RLA [11]. This setting is closely linked to FSL in noisy contexts, as explored by approaches such as DCML [20].

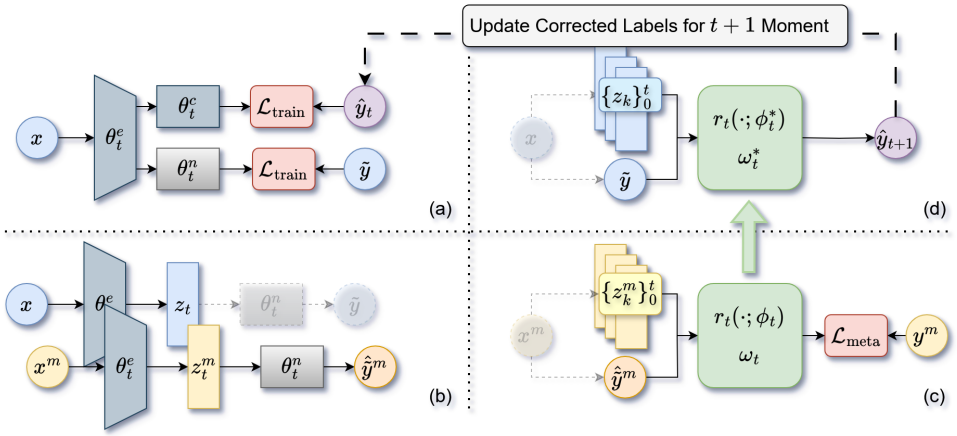

Figure 1: This framework provides a general overview of the approach. (a) Training the classifier on the corrected labels. (b) Feature collection and simulation of the clean-data noisy posterior. (c) Optimization of the correction function. (d) Noisy label correction. The dashed lines and translucent elements only represent potential causal relationships.

3 Preliminaries

In this study, we address the problem of c-class classification. The input feature space is represented as $x \in \mathbb{R}^d$, and the label space is denoted as $y \in \mathbb{R}^c$, where c signifies the number of classes. We define the noisy training dataset as $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N$, where each $\tilde{y}_i$ is an observed noisy label. We also incorporate a subset of trusted data, referred to as the clean validation dataset $\mathcal{D}^{\text{m}} = \{(x_i^m, y_i^m)\}_{i=1}^M$. Usually, $M \ll N$ holds, as is common in real-world scenarios.

Let $f(x;\theta)$ be a neural network model parameterized by θ for classification. We assume that our classification model comprises a feature extractor $e(\mathbf{x};\theta^e)$ and a prediction head $g(\mathbf{z};\theta^c)$. The extractor $e(\mathbf{x};\theta^e)$ maps an input $\mathbf{x} \in \mathbb{R}^d$ to a feature vector $\mathbf{z} \in \mathbb{R}^{d'}$ (where $d' \ll d$). Subsequently, $g(\mathbf{z};\theta^c)$, parameterized by θ^c, produces class probabilities via a softmax layer, yielding $f(\mathbf{x};\theta^e,\theta^c) = g(e(\mathbf{x};\theta^e);\theta^c)$ to approximate $p(y|\mathbf{x})$. We also augment this model with an auxiliary "noisy" prediction head $g_n(\mathbf{z};\theta^n)$ as detailed in Section 4.1. This noisy head shares the feature extractor $e(\mathbf{x};\theta^e)$ and possesses an identical architecture to $g(\cdot;\theta^c)$, yielding $f_n(\mathbf{x};\theta^e,\theta^n) = g_n(e(\mathbf{x};\theta^e);\theta^n)$ to approximate $p(\tilde{y}|\mathbf{x})$. The complete model parameters are thus $\theta = \{\theta^e,\theta^c,\theta^n\}$. Our goal is to train this augmented model to achieve strong generalization on clean data with $f(\mathbf{x};\theta^e,\theta^c)$, even when primarily learning from noisy labels.

4 Methods

Our method is an iterative self-training framework that jointly optimizes a *primary classifier* and a *label correction mechanism*. This creates a synergistic loop: the correction mechanism, learned on the clean dataset $\mathcal{D}^{\text{m}}$, refines noisy labels $\tilde{y}$ from $\tilde{\mathcal{D}}$ into cleaner estimates $\hat{y}$; the main classifier f then trains on these $\hat{y}$; an improved classifier subsequently helps the correction mechanism produce better $\hat{y}$ for the next iteration.

We summarize our process in Fig. 1. Within each iteration t, key operations include: (a) updating the classifier model parameters $(\theta_t^e,\theta_t^c,\theta_t^n)$ using $\hat{y}_t$ (for f) and $\tilde{y}$ (for f_n); (b) collecting features with $e(\cdot,\theta_t^e)$ on $\tilde{\mathcal{D}}$ and $\mathcal{D}^{\text{m}}$, simulating noisy posteriors via f_n ; (c) optimizing the label correction mechanism using $\mathcal{D}^{\text{m}}$ along with its simulated noisy posteriors and collected features; (d) finally, generating new corrected labels $\hat{y}_{t+1}$ for samples in $\tilde{\mathcal{D}}$ to guide the subsequent classifier training.

4.1 Noisy Posterior Simulation

To learn how to correct noisy labels back to clean ones, we leverage the clean dataset $\mathcal{D}^{\text{m}}$ where ground truth is available. However, $\mathcal{D}^{\text{m}}$ contains only clean labels without corresponding noisy

versions, presenting a fundamental challenge: **how can we train a correction function without noisy-clean label pairs?** To address this limitation, we introduce noisy posterior simulation to generate synthetic noisy labels for the clean dataset. We train a "noisy classifier" on $\tilde{\mathcal{D}}$ to approximate $p(\tilde{y}|x)$ and then apply it to $\mathcal{D}^{\mathrm{m}}$ to estimate $p(\tilde{y}^m|x^m)$. This approach builds a bridge between noisy and clean label distributions, enabling the knowledge from $\tilde{\mathcal{D}}$ to correct labels in $\tilde{\mathcal{D}}$ despite working with entirely different instances.

We implement this simulation using a two-head architecture similar to [6] for efficient training. The noisy classifier shares the feature extractor $e(\cdot;\theta^e)$ with f, but uses a different prediction head which is parameterized with θ^n. This forms a noisy prediction model $f_n(x;\theta) = g_n(e(x;\theta^e);\theta^n)$. The noisy classifier is explicitly trained to predict the observed noisy labels in $\tilde{\mathcal{D}}$. This is accomplished by jointly optimizing θ_t at iteration t using the composite loss function:

$$\theta_t = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Big(\mathcal{L}_{\text{train}}(f(x_i;\theta), \hat{y}_{t,i}) + \lambda \mathcal{L}_{\text{train}}(f_n(x_i;\theta), \tilde{y}_i) \Big). \tag{1}$$

The first term directs the clean classifier f towards the corrected clean labels, while the second term compels the noisy classifier f_n to model the characteristics of the actual noisy labels. The hyperparameter λ balances these objectives, promoting the learning of robust features aligned with clean labels while simultaneously capturing noise patterns via f_n to prevent overfitting to noise.

After updating θ_t, the simulated noisy posterior for each clean sample $x^m \in \mathcal{D}^{\mathrm{m}}$ is generated as $\hat{\tilde{y}}^m = f_n(x^m;\theta_t)$. This simulated posterior $\hat{\tilde{y}}^m$ serves as the indispensable noise-informative input during the process on the clean dataset in the following sections.

4.2 Robust Closed-loop Label Correction

Neural Label Correction Compared to transition matrices, neural networks offer greater expressive capacity for modeling complex, instance-dependent label noise and exhibit lower variance. We propose a neural correction module r, parameterized by ϕ, to map noisy labels to clean label distributions. It leverages two inputs: (i) the noisy label posterior ($\tilde{y}$ for $\tilde{\mathcal{D}}$, or simulated $\hat{\tilde{y}}^m$ for $\mathcal{D}^{\mathrm{m}}$) and (ii) a temporal ensemble of K historical feature representations $\{z_k = e(x;\theta^e_k)\}_{k=0}^{K}$ extracted by the primary classifier throughout its training trajectory.

The use of mid-layer features is inspired by the early learning phenomenon where neural networks initially capture cleaner patterns before memorizing noise, which underlies pseudo-labeling (or bootstrapping) techniques. We extend this intuition and hypothesize that feature embeddings from earlier training epochs may also retain more stable and generalizable semantic information. Importantly, compared to the final-layer logits output (or one-hot prediction), intermediate feature representations contain more information about the original samples, enabling the correction function r to infer potential sources of classification error. Meanwhile, features from multiple timesteps (rather than only the latest features) are used to mitigate fluctuations in neural network training, enhancing prediction stability. While features implicitly convey information about the noisy prediction, the direct noisy posterior—particularly as modeled by the dedicated noisy head $g_n(\cdot;\theta^n)$ offers a more focused signal regarding the characteristics differentiating clean from noisy instances, and is thus fed into the correction network.

The correction network r, parameterized by ϕ, outputs a corrected c-dimensional label distribution $\bar{y} = r(\tilde{y}, \{z_k\}_{k=0}^{K}; \phi)$ via softmax. It has an input dimension of $c + (K+1)\cdot d'$ and an output dimension of c, where d' is the feature dimensionality. To prevent overfitting on the often small $\mathcal{D}^{\mathrm{m}}$, r is designed as a shallow network (e.g., a 2-layer MLP in our experiment).

Training the correction neural network r on the potentially small clean dataset $\mathcal{D}^{\mathrm{m}}$ nearly always presents significant overfitting. To this end, $\mathcal{D}^{\mathrm{m}}$ is partitioned into a training set $\mathcal{D}^{\mathrm{m}}_{\text{train}}$ and a validation set $\mathcal{D}^{\mathrm{m}}_{\text{val}}$. The parameters ϕ are optimized by minimizing a meta-loss (e.g., cross-entropy) on $\mathcal{D}^{\mathrm{m}}_{\text{train}}$, and the final parameters ϕ^* are selected based on performance on the held-out validation set $\mathcal{D}^{\mathrm{m}}_{\text{val}}$, to mitigate overfitting. These parameters underpin the robust label correction process described subsequently.

Robust Convex Combination We define $\bar{y}_t = r_t(\tilde{y}, \{z_k\}_{k=0}^t; \phi_t^*)$ as the prediction of the neural correction network r_t at iteration t, and $\bar{y}_t^m = r_t(\hat{\tilde{y}}^m, \{z_k^m\}_{k=0}^t; \phi_t^*)$ as its prediction on the clean validation set. Despite careful training and validation, the neural correction network $r_t(\cdot; \phi_t^*)$ optimized at any given iteration t might still produce suboptimal predictions $\bar{y}_t$ due to factors like limited clean data or inherently challenging noise patterns. Relying solely on the latest prediction $\bar{y}_t$ could be risky: **if $\bar{y}_t$ is flawed, using it directly as the target label $\hat{y}_t$ could amplify errors and destabilize the classifier training.**

To enhance robustness, we combine the noisy label $\tilde{y}$ with historical predictions $\{\bar{y}_k\}_{k=1}^t$ via a convex combination, forming an ensemble that reduces variance and mitigates poor corrections. With a $(t+1)$-dimensional weight vector ω_t , the corrected label after iteration t is defined as:

$$\hat{y}_{t+1} = \omega_{t,0}\tilde{y} + \sum_{k=1}^{t} \omega_{t,k}\bar{y}_k, \quad \text{subject to} \sum_{k=0}^{t} \omega_{t,k} = 1, \omega_{t,k} \geq 0. \tag{2}$$

Proposition 1. *Let $\hat{y}_0^* = \tilde{y}$ for $K = 0$, and $\hat{y}_K^* = \omega_0^*\tilde{y} + \sum_{k=1}^K \omega_k^*\bar{y}_k$ for $K \geq 1$, where the weight vector $\omega^* = (\omega_0^*, \omega_1^*, \ldots, \omega_K^*)$ minimizes the expected risk $\mathcal{R}(\hat{y}_K^*)$, subject to $\sum_{k=0}^K \omega_k^* = 1$ and $\omega_k^* \geq 0$ for all k. Then:*

$$\mathcal{R}(\hat{y}_K^*) \leq \min\left\{\mathcal{R}(\tilde{y}), \min_{1\leq k\leq K} \mathcal{R}(\bar{y}_k)\right\}, \tag{3}$$

$$\mathcal{R}(\hat{y}_K^*) \leq \mathcal{R}(\hat{y}_{K-1}^*) \quad \textit{for all} \quad K \geq 1. \tag{4}$$

That is, the expected risk $\mathcal{R}(\hat{y}_K^*)$ is bounded by the minimum risk of individual components and does not increase as K increases.

This guarantees that the optimal corrected label $\hat{y}^*$ remains robust when arbitrary $\bar{y}_k$ (e.g. suboptimal or even near-uniform predictions) are added, as ω^* mitigates their impact in noisy label scenarios. Since the correction network has already overfitted to $\mathcal{D}_{\text{train}}^{\text{m}}$, we approximate ω_t^* by minimizing the meta-loss on $\mathcal{D}_{\text{val}}^{\text{m}}$:

$$\omega_t^* = \arg\min_{\omega\in\Delta^t} \frac{1}{|\mathcal{D}_{\text{val}}^{\text{m}}|} \sum_{i\in\mathcal{D}_{\text{val}}^{\text{m}}} \mathcal{L}_{\text{meta}}\left(\omega_{t,0}\hat{\tilde{y}}_i^m + \sum_{k=1}^{t} \omega_{t,k}\bar{y}_{k,i}^m, y_i^m\right), \tag{5}$$

where Δ^t is the t-simplex. The low dimensionality of ω mitigates overfitting, producing robust labels $\hat{y}_{t+1}$ for classifier training. This step yields the final high-quality corrected labels $\hat{y}_{t+1}$ used for classifier training in the subsequent iteration.

4.3 Algorithm Details

The overall iterative self-training process is detailed in Algorithm 1. The algorithm begins with initialization and then iteratively updates the classifier parameters, extracts features, simulates noisy posteriors, and optimizes the correction network. In each iteration, it generates new predictions using the correction network, computes optimal combination weights, and updates the corrected labels for the next round. This process continues until a stopping criterion is met, progressively improving both the classifier and the quality of corrected labels. For a comprehensive discussion of our method's details, limitations, and broader impacts, please see Appendix A and Appendix B.

The computational cost associated with optimizing the correction network r and the weights ω remains minimal. This is primarily due to the typically small size of the clean dataset $\mathcal{D}^{\text{m}}$, the shallow architecture of r (e.g., a 2-layer MLP), and the fact that feature embeddings generally have lower dimensionality than the raw input data. In our CIFAR experiments, these optimization steps constituted only about 1% of the total training time, ensuring that our method does not introduce a significant computational overhead.

Iterative Improvement Mechanism The efficacy of our method is significantly driven by its iterative improvement mechanism, which functions based on three interconnected principles:

- Our mechanism establishes a closed-loop feedback system. The clean dataset, $\mathcal{D}^{\text{m}}$, provides essential ground-truth supervision, enabling the transfer of true label information to the noisy training data via the learned correction function and preventing error amplification.

Algorithm 1 Self-Training with Closed-loop Label Correction

Require: Noisy training dataset $\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^{N}$; clean dataset $\mathcal{D}^{\mathrm{m}} = \{(x_i^m, y_i^m)\}_{i=1}^{M}$.

Set iteration counter $t \leftarrow 0$.

Initialize corrected labels $\hat{y}_{0,i} \leftarrow \tilde{y}_i$ for $i = 1, \ldots, N$.

Divide $\mathcal{D}^{\mathrm{m}}$ into $\mathcal{D}^{\mathrm{m}}_{\text{train}}$ and $\mathcal{D}^{\mathrm{m}}_{\text{val}}$.

repeat

Update classifier parameters θ_t with Eq. (1) using $\hat{y}_t$ and $\tilde{y}$ for several epochs.

Extract and store features, simulate the noisy posterior:

$$
\begin{aligned}
z_{t,i} &\leftarrow e(x_i; \theta_t^e), && i = 1, \ldots, N; \\
z_{t,i}^m &\leftarrow e(x_i^m; \theta_t^e), && i = 1, \ldots, M; \\
\hat{\tilde{y}}_i^m &\leftarrow g_n(z_{t,i}^m; \theta_t^n), && i = 1, \ldots, M.
\end{aligned}
$$

Initialize and optimize $r_t(\cdot, \phi_t)$ by training on $\mathcal{D}^{\mathrm{m}}_{\text{train}}$ and validating on $\mathcal{D}^{\mathrm{m}}_{\text{val}}$ to obtain ϕ_t^*.

Generate predictions using the current correction network $r(\cdot; \phi_t^*)$:

$$
\begin{aligned}
\bar{y}_{t,i} &\leftarrow r_t(\tilde{y}_i, \{z_{k,i}\}_{k=0}^{t}; \phi_t^*), && i = 1, \ldots, N; \\
\bar{y}_{t,i}^m &\leftarrow r_t(\hat{\tilde{y}}_i^m, \{z_{k,i}^m\}_{k=0}^{t}; \phi_t^*), && i = 1, \ldots, M.
\end{aligned}
$$

Initialize and optimize ω_t^* on $\mathcal{D}^{\mathrm{m}}_{\text{val}}$.

Update corrected labels $\hat{y}_{t+1}$ for samples in $\tilde{\mathcal{D}}$ using ω_t^*:

$$
\hat{y}_{t+1,i} \leftarrow \omega_{t,0}^* \tilde{y}_i + \sum_{k=1}^{t} \omega_{t,k}^* \bar{y}_{k,i}, \quad i = 1, \ldots, N.
$$

Set $t \leftarrow t + 1$.

until Stopping criterion met

- Our approach leverages the widely recognized early learning phenomenon that neural networks typically learn cleaner, more generalizable patterns from the data before beginning to memorize noisy instances, providing a strong foundation for the label correction module.
- A virtuous cycle emerges through iterative refinement, where increasingly accurate labels enhance the classifier's feature representations, which in turn improve the label correction function. Proposition 1 guarantees that this cycle maintains or improves performance, ensuring the algorithm's stability.

5 Experiments

5.1 Experiments on CIFARs

We first evaluate our method on CIFAR-10 and CIFAR-100 [5], which is widely used for machine learning research. Both datasets contain 50,000 training and 10,000 test images. We randomly select 5,000 images from the training set as the clean dataset.

Noise settings Let η denote the corruption ratio. We first simulate two types of instance-independent noise. **Symmetric**: Generated by flipping the labels of a given proportion of training samples to all the other class labels uniformly. Each instance with a true label y is corrupted to all possible classes with probability $\frac{\eta}{c}$. **Asymmetric**: Generated by flipping the labels of a given proportion to one or few particular classes. For CIFAR-10, we disturb the label in its similar class with probability η following [28]. For CIFAR-100, we disturb the label to other classes in its super-class as described in [4]. Noise transition matrices are generated first, and then labels are randomly corrupted based on the matrix for each instance-independent label noise setting. Additionally, we test an **Instance-dependent** noise setting, simulated as described in [36], applied to both datasets. We vary η in the range of $[0, 1]$ to simulate different noise levels for all noise types.

Baselines We compare our methods with a series of advanced methods in noisy label learning that use clean data. The compared methods include: *Transition-Matrix-Based Methods*: **FasTEN.** [6] introduce a two-head architecture to efficiently estimate the label transition matrix every iteration

Table 1: Test accuracy (%) of all competing methods on CIFAR-10 and CIFAR-100 under different noise types with different noise levels. The best results are highlighted in bold.

Dataset	Method		Symmetric			Asymmetric			Instance-dependent		
			η			η			η		
			0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
CIFAR-10	CE	Best	91.13	87.93	82.54	92.87	90.00	71.99	90.75	82.74	68.52
		Last	82.84	71.30	59.93	88.45	79.92	70.19	78.15	54.54	39.72
	FasTEN	Best	91.46	88.42	82.88	92.87	92.68	91.39	91.19	85.44	77.09
		Last	86.19	77.62	70.04	91.63	88.82	85.71	86.82	81.16	61.88
	MW-Net	Best	90.55	88.02	83.45	93.03	91.29	65.42	90.66	82.30	63.83
		Last	84.86	72.58	57.04	90.49	87.21	61.65	79.29	66.94	52.41
	L2B	Best	89.15	86.27	80.38	91.37	90.56	89.66	89.67	85.98	74.23
		Last	88.83	86.27	80.23	91.23	90.51	89.23	89.46	85.79	74.23
	EMLC	Best	90.77	87.84	82.00	91.67	90.92	73.08	83.63	79.35	72.35
		Last	81.60	71.24	58.55	88.51	80.95	68.11	80.32	74.09	61.57
	Ours	Best	**92.12**	**89.40**	**84.71**	**93.54**	**92.80**	**91.71**	**92.48**	**90.37**	**85.26**
		Last	**91.80**	**88.81**	**84.28**	**93.39**	**92.72**	**91.60**	**92.33**	**90.03**	**84.94**
CIFAR-100	CE	Best	67.59	59.99	49.94	69.69	65.33	55.96	62.57	45.90	24.05
		Last	62.93	45.80	26.81	62.80	46.62	30.75	60.01	40.09	20.88
	FasTEN	Best	68.09	62.27	50.42	67.64	53.42	31.50	66.57	59.29	47.39
		Last	59.37	43.65	26.09	61.91	43.27	31.50	59.70	45.79	33.15
	MW-Net	Best	67.94	62.26	**54.24**	70.04	65.26	57.23	62.87	48.31	26.53
		Last	62.38	48.59	29.55	62.87	47.82	31.95	59.25	40.91	22.75
	L2B	Best	64.57	54.54	20.66	64.91	58.42	43.33	66.27	59.55	47.45
		Last	64.28	54.38	20.66	64.66	58.11	43.33	66.05	59.55	47.45
	EMLC	Best	**68.67**	62.49	46.90	70.10	65.06	57.15	63.82	53.99	32.11
		Last	62.80	46.51	31.74	62.94	47.09	32.34	61.94	49.34	28.44
	Ours	Best	68.25	**63.50**	52.00	**71.16**	**67.15**	**64.23**	**69.72**	**62.38**	**53.73**
		Last	**67.34**	**62.04**	**51.38**	**70.21**	**66.21**	**63.75**	**69.67**	**62.38**	**53.73**

within a single back-propagation. *Meta-Learning-Based Methods*: **MW-net.** [15] uses an MLP net and meta-learns the weighting function. **L2B.** [34] uses a bootstrapping method with meta-learned weighting coefficient. **EMLC.** [18] proposes a teacher-and-student architecture and meta-learns the taught soft-label. We exclude label selection methods due to their high clean data requirements and low noise utilization. For fair comparison, we disable EMLC's self-supervised initialization and remove FasTEN's direct training term ($\ell(f(x^m), y^m)$) on clean data (retaining only transition matrix estimation), to focus comparisons on the core noise-handling mechanisms.

Implementations In our experiments on CIFAR-10 and CIFAR-100, we employed ResNet-34 as the backbone architecture. For both datasets, we applied horizontal flip and random crop as data augmentations. The models were trained using stochastic gradient descent (SGD) optimizer with momentum 0.9 and weight decay $5e^{-4}$ for 100 epochs, and the batch size was set to 128. We set the initial learning rate to 0.1 with a step decay scheduler that reduces the learning rate by a factor of 0.1 at epochs 60 and 80. A full description of the experimental setup can be found in Table 5.

For the specific settings of our method, we make the loss balancing factor $\lambda = 0.5$ in Eq. (1). We divide $\mathcal{D}^{\text{m}}$ into training $\mathcal{D}^{\text{m}}_{\text{train}}$ and validation subsets $\mathcal{D}^{\text{m}}_{\text{val}}$ using a ratio $0.8 : 0.2$. The neural network correction model is a 2-layer MLP with 256 hidden units, ReLU activation, and a softmax output layer. We make $\mathcal{L}_{\text{meta}}$ to be Cross-Entropy and use SGD with learning rate 0.001 (reduced to 0.0001 after first validation decline) on $\mathcal{D}^{\text{m}}_{\text{train}}$ to optimize the ω, while we employ sequential least squares programming to optimize ω on $\mathcal{D}^{\text{m}}_{\text{val}}$. We update the feature sets, correction function, and update the corrected labels every 5 epochs, after the initial 40-epoch warm-up, which is intended to allow for an adequately initialized noisy posterior. For a comprehensive analysis of our method's sensitivity to hyper-parameters, we present ablation studies in Appendix C.1, examining both the impact of varying λ values and the effect of different correction function update frequencies.

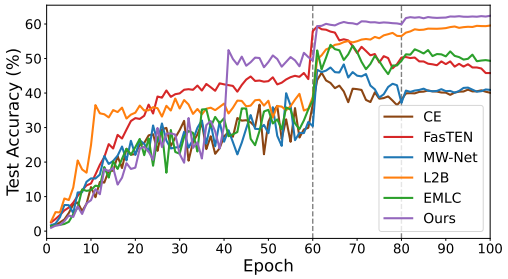

Figure 2: Training dynamics of different methods on CIFAR-100 under instance-dependent noise with a level of 0.4. The vertical dashed lines represent the epochs at which the learning rate decays at 60 and 80 epochs.

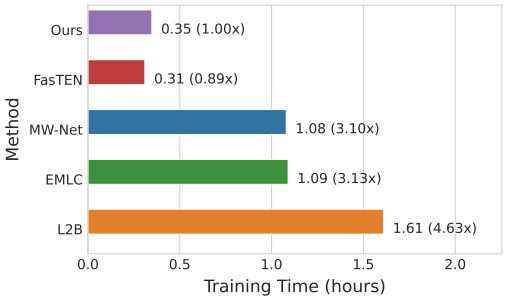

Figure 3: Comparison of average GPU training time on CIFAR-100 (in hours) for various methods using a single RTX 4090. The values in parentheses indicate the relative ratio of training time compared to our method.

Table 2: Test accuracy (%) of our method and its ablations on CIFAR-100 under different noise types with different noise levels. The last recorded performances in the training procedure are reported.

Ablations	Symmetric			Asymmetric			Instance-dependent		
	η			η			η		
	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
Input Final-Layer Logits	67.56	59.33	48.85	68.40	63.08	57.67	67.99	57.41	48.37
Input Pseudo-Soft-Labels	**69.71**	62.08	48.30	69.99	66.31	49.25	**71.18**	**63.44**	51.95
No Feature Ensemble	68.48	61.34	**53.13**	**70.27**	**68.19**	62.94	70.34	62.00	**55.77**
Ours	67.34	**62.04**	51.38	70.21	66.21	**63.75**	69.67	62.38	53.73

Results The comparative performance analysis is presented in Table 1, with demonstrative training dynamics for CIFAR-100 under instance-dependent noise at noise rate 0.4, illustrated in Fig. 2. Generally, our method consistently outperforms other approaches across all noise scenarios, demonstrating strong robustness to label noise. Notably, while other approaches show substantial performance degradation in later training epochs (likely attributable to overfitting on noisy patterns), our approach maintains stable performance trajectories without significant accuracy decline. This stability empirically validates that our closed-loop correction mechanism effectively leverages the small clean dataset to guide the learning process, preventing memorization of noisy patterns.

Training Time Consumption Comparison We have quantified the training time for our method on a single RTX 4090 GPU. A detailed breakdown of the costs with respect to each step of our algorithm is available in Appendix C.1. We compared the complete training time with the baseline methods. These results are presented in Fig. 3. Our approach secures a leading position in terms of training efficiency, necessitating less time compared to existing meta-learning-based methods.

Ablations on the Correction Network Input In this experiment, we test the influence of different input inside the correction network. We input the logits and pseudo-soft-labels (which is softmax after logits) to test whether features extracted from earlier layers could contribute more meaningfully to label correction. In these situations, the input dimension of r_t becomes $(1+t)\cdot c$ at each iteration t. We also test whether using all previous representations enhances performance compared to using only the latest one, even though the most recent representation theoretically captures the best features because the classifier is trained on the non-increasing risk labels. As a result, the input dimension of r_t becomes $c + d'$ where d' is the dimension of the features.

The ablation results are presented in Table 2, which reveal several key insights into input selection for the correction network. First, when considering direct classifier outputs, pseudo-soft-labels generally prove more effective than raw logits as input for r_t; moreover, pseudo-labels emerge as a strong input candidate under various conditions. However, as noise intensity increases, such as at the 0.6 noise ratio across all tested noise types, our primary approach utilizing intermediate feature embeddings demonstrates superior robustness and accuracy compared to configurations based on either logits or pseudo-labels. Furthermore, the experiments indicate that employing only the most recent feature

Table 3: Comparison of test accuracy (%) on Clothing1M with advanced methods utilizing the clean data (or its subset). The symbol ✦denotes the use of meta-knowledge, ✓ denotes that the method is directly trained on clean instances with full forward and back propagation on the classifier parameters, and ♦ denotes methods that are trained with additional self-supervised learning procedures.

Method	CE	FasTEN	MW-Net	EMLC♦	DMLP♦			Ours		
Accuracy	69.21	77.83	73.72	79.35	75.50	77.30	77.31	78.40	79.61	**80.33**
(Extra data)	(✗)	(✓)	(✦)	(✦)	(10%, ✦)	(50%, ✦)	(✦)	(10%, ✦)	(50%, ✦)	(✦)

representation, as opposed to the full historical ensemble, also consistently achieves commendable results, highlighting its efficacy as a streamlined alternative.

These observations underscore that the optimal choice of input for the correction network can be context-dependent. In practical applications, selection should be guided by factors such as prior knowledge of the task, the anticipated corruption level, and the scale of the available clean dataset. Such considerations regarding the design and input modalities for the correction network are further discussed in Appendix A.1.

5.2 Experiments on Clothing1M

Clothing1M [25] is a real-world large-scale dataset where image instances come from online shopping websites with noisy labels derived from user tags. For these experiments, we used ResNet-50 pre-trained on ImageNet as our backbone, following previous works. The model was trained on a single RTX 4090 GPU for 15 epochs using SGD optimizer with momentum 0.9, weight decay $5e^{-4}$, and batch size 256. We set the initial learning rate to 0.01 with step decay at epochs 5 and 10. The training pipeline included more data augmentations: resized crop, horizontal flip, random color jitter, and random grayscale conversion following previous works. All label correction parameters remain consistent with previous experiments, maintaining the loss balancing factor $\lambda = 0.5$. We adjusted the correction function update frequency to every 3 epochs due to the dataset's scale. In addition to the baseline methods, we also consider another leading approach DMLP [19] for comparison. To investigate the impact of clean data quantity, we conduct experiments with two other different clean sample set sizes (10% and 50%), revealing insights about our method's robustness to varying levels of supervision quality.

Results The comparison results are presented in Table 3. Our method is categorized as ✦in the table, because it *leverages the clean data as meta-knowledge for noisy label correction*, differentiating this from *direct training on such clean instances*. Our approach achieves the highest accuracy of 80.33%, outperforming other leading methods such as EMLC (79.35%) and DMLP (77.31%), even when these competing methods incorporate additional self-supervised learning techniques. Furthermore, when utilizing only 10% of the clean data, our approach still surpasses DMLP, demonstrating that our method is capable of effectively utilizing limited clean data, achieving competitive performance even with smaller subsets. We also compare the computational efficiency of our method against L2B and EMLC on Clothing1M in Appendix C.2, demonstrating our approach's advantages in terms of both training time and memory consumption. These findings indicate that our method offers a more reliable strategy for learning with noisy labels in real-world, large-scale scenarios.

6 Conclusion

In this paper, we proposed a novel self-training label correction framework to address learning with noisy labels. Our approach employs decoupled optimization, enabling a classifier and a label correction function to co-evolve, guided by a small clean dataset to prevent error accumulation and enhance correction quality. We introduce noisy posterior simulation to bridge distributions between clean data and noisy labels, and leverage intermediate features for finer-grained label correction. Furthermore, a robust convex combination strategy, theoretically guaranteed for stability, is used for label updates. Empirical experiments on benchmarks like CIFAR and Clothing1M demonstrate that our method significantly outperforms current advanced techniques while also achieving higher training efficiency. This work offers an effective and robust solution for tackling large-scale noisy data with limited clean supervision in practical settings.

References

[1] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 70 of *PMLR*, pages 233–242, 06–11 Aug 2017.

[2] Z. Fu, T. Liu, N. Li, C. Zhang, Z. Yang, and Y. Liu. Transition-aware weighted denoising score matching for training conditional diffusion models with noisy labels. *ICLR 2024 submission*, 2024.

[3] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. W. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[4] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. *Advances in neural information processing systems*, 31, 2018.

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

[6] S. M. Kye, K. Choi, J. Yi, and B. Chang. Learning with noisy labels by efficient transition matrix estimation to combat label miscorrection. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV*, pages 717–738. Springer, 2022.

[7] J. Li, R. Socher, and S. C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[8] M. Li and C. Zhu. Noisy label processing for classification: A survey. *arXiv preprint arXiv:2404.04159*, 2024.

[9] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Advances in Neural Information Processing Systems*, volume 33, pages 16324–16335, 2020.

[10] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[11] Z. Lyu, Y. Zhang, Y. Li, Y. Li, G. Niu, Q. Wu, and M. Sugiyama. Learning from long-tailed noisy data with robust logit adjustment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(15): 16856–16864, 2024.

[12] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[13] C. O. Pene, A. Ghiassi, T. Younesian, R. Birke, and L. Y. Chen. Multi-label gold asymmetric loss correction with single-label regulators. *arXiv preprint arXiv:2108.02032*, 2021.

[14] N. Shafir, G. Hacohen, and D. Weinshall. Active learning with a noisy annotator. *arXiv preprint arXiv:2504.04506*, 2025.

[15] 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. *Advances in neural information processing systems*, 32, 2019.

[16] R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[17] H. Song, M. Kim, D.-G. Park, Y. Shin, and J.-G. Lee. SELF: A self-paced learning framework for learning with noisy labels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *PMLR*, pages 5930–5939, 09–15 Jun 2019.

[18] M. K. Taraday and C. Baskin. Enhanced meta label correction for coping with label corruption. *arXiv preprint arXiv:2305.12961*, 2023.

[19] Y. Tu, B. Zhang, Y. Li, L. Liu, J. Li, Y. Wang, C. Wang, and C. R. Zhao. Learning from noisy labels with decoupled meta label purifier. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19934–19943, 2023.

[20] H. Wang, R. Qian, Y. Liu, Z. Chen, Y. Dou, S. Y. Philip, and Q. Li. Dual-level curriculum meta-learning for complicated noisy few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17210–17218, 2024.

[21] Z. Wang, G. Hu, and Q. Hu. Training noise-robust deep neural networks via meta-learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4524–4533, 2020.

[22] D.-D. Wu, D.-B. Wang, and M.-L. Zhang. Revisiting consistency regularization for deep partial label learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *PMLR*, pages 23980–23997, 17–23 Jul 2022.

[23] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[24] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama. Are anchor points really indispensable in label-noise learning? *Advances in Neural Information Processing Systems*, 32, 2019.

[25] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015.

[26] E. Yang, P. Chen, Z. Liu, T. Liu, A. Weller, and W. You. Label-retrieval-augmented diffusion models for learning from noisy labels. 36, 2023.

[27] S. Yang, E. Yang, B. Han, Y. Liu, M. Xu, G. Niu, and T. Liu. Estimating instance-dependent label-noise transition matrix using dnns. 2021.

[28] J. Yao, H. Wu, Y. Zhang, I. W. Tsang, and J. Sun. Safeguarded dynamic label regression for noisy supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9103–9110, 2019.

[29] C. Yu, X. Ma, and W. Liu. Delving into noisy label detection with clean data. In *International Conference on Machine Learning*, pages 40290–40305. PMLR, 2023.

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

[31] W. Zhang, Y. Wang, and Y. Qiao. Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2019.

[32] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[33] Z. Zhang, H. Zhang, S. O. Arik, H. Lee, and T. Pfister. Distilling effective supervision from severe label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9294–9303, 2020.

[34] Y. Zhou, X. Li, F. Liu, Q. Wei, X. Chen, L. Yu, C. Xie, M. P. Lungren, and L. Xing. L2b: Learning to bootstrap robust models for combating label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23523–23533, 2024.

[35] W. Zhu, Y. Chen, X. Wu, X. Wang, and D. N. Metaxas. Zmt: Zero-shot multi-task learning for semi-supervised learning with noisy pseudo-labels. *arXiv preprint arXiv:2502.12584 (Under review for ICML 2025)*, 2025.

[36] Z. Zhu, T. Liu, and Y. Liu. A second-order approach to learning with instance-dependent label noise. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10113–10123, 2021.

A Further Details on Our Method

A.1 Considerations for the Correction Neural Network

Achieving robust generalization with the neural correction network r, especially when the guiding clean dataset $\mathcal{D}^{\text{m}}$ is of limited size, necessitates careful design of both its input modalities and structural complexity. The characteristics of $\mathcal{D}^{\text{m}}$, particularly its size, critically dictate these choices to ensure generalization. When $\mathcal{D}^{\text{m}}$ is small, the primary concern is mitigating overfitting. As noted in the main text, r is thus designed as a shallow network (e.g., a 2-layer MLP). Crucially, beyond architectural simplicity, the input space to r must also be constrained. For instance, instead of an ensemble of K historical feature representations $\{z_k\}_{k=0}^{K}$, one might use only the most recent feature embedding $z_t = e(x; \theta_t^e)$, significantly reducing input dimensionality (from $c + (K+1) \cdot d'$ to $c + d'$) and the number of parameters in the correction neural network. Alternative or supplementary inputs include:

- **Final layer logits:** Raw logits from the primary classifier $f(x; \theta)$ or the noisy classifier $f_n(x; \theta)$ offer a rich, pre-softmax representation.
- **Pseudo-labels:** Labels from earlier training stages or auxiliary models can serve as useful signals, which have been widely used in LNL context.
- **Other Prior Knowledge:** Domain-specific information or encoded prior knowledge relevant to the noise or true label characteristics.

Conversely, if $\mathcal{D}^{\text{m}}$ is relatively large, a more expressive correction network r with a richer set of inputs, including the full historical feature ensemble, can be employed, as there is more data to learn complex relationships. A practical alternative is to train multiple correction networks with various input configurations and select the one yielding the highest performance on $\mathcal{D}^{\text{m}}_{\text{val}}$ as the final predictor. To further bolster generalization, especially with a smaller $\mathcal{D}^{\text{m}}$, the validation of r on a held-out set $\mathcal{D}_{\text{valm}}$ remains essential for selecting a model ϕ^* that performs robustly on unseen clean data.

The choice of input for the correction network is a data-dependent trade-off. While using only the latest features is effective and computationally efficient for moderately-sized datasets like CIFAR where mitigating overfitting on a small clean set is paramount, a historical ensemble can provide enhanced stability on more complex, large-scale datasets. This is because the ensemble averages out training fluctuations, providing a richer and more stable semantic signal. To validate this, we conducted an ablation on the **Clothing1M** dataset, comparing the performance of our full method (using a historical feature ensemble) against a variant using only the most recent feature representation.

Table 4: Ablation on correction network input for Clothing1M.

Method on Clothing1M	**Test Accuracy (%)**
Using Latest Feature Only	80.13
Ours (with historical features)	**80.33**

The results in Table 4 show that the historical feature ensemble provides a tangible performance benefit in this large-scale, real-world noise setting, validating its role in enhancing stability.

A.2 Elaboration on Robust Convex Combination

The main text introduces a robust convex combination strategy to generate corrected labels $\hat{y}_{t+1}$. This section reiterates the proposition regarding its risk-non-increasing nature and further discusses its implications.

Proposition 1. *Let $\hat{y}_0^* = \tilde{y}$ for $K = 0$, and $\hat{y}_K^* = \omega_0^* \tilde{y} + \sum_{k=1}^{K} \omega_k^* \bar{y}_k$ for $K \geq 1$, where the weight vector $\omega^* = (\omega_0^*, \omega_1^*, \ldots, \omega_K^*)$ minimizes the expected risk $\mathcal{R}(\hat{y}_K^*)$, subject to $\sum_{k=0}^{K} \omega_k^* = 1$ and $\omega_k^* \geq 0$ for all k. Then:*

$$\mathcal{R}(\hat{y}_K^*) \leq \min\left\{\mathcal{R}(\tilde{y}), \min_{1 \leq k \leq K} \mathcal{R}(\bar{y}_k)\right\}, \tag{6}$$

$$\mathcal{R}(\hat{y}_K^*) \leq \mathcal{R}(\hat{y}_{K-1}^*) \quad \textit{for all} \quad K \geq 1. \tag{7}$$

That is, the expected risk $\mathcal{R}(\hat{y}_K^*)$ is bounded by the minimum risk of individual components and does not increase as K increases.

Proof. We prove both inequalities by examining special cases of the weight optimization problem.

For the first inequality, consider the specific weight vectors that place all weight on a single component: $\omega^{(0)} = (1, 0, \ldots, 0)$ and $\omega^{(j)} = (0, \ldots, 1, \ldots, 0)$ with 1 at position j for each $j \in \{1, \ldots, K\}$. These vectors yield $\hat{y}_K^{(0)} = \tilde{y}$ and $\hat{y}_K^{(j)} = \bar{y}_j$, respectively. Since ω^* is optimal:

$$\mathcal{R}(\hat{y}_K^*) \leq \min\left\{\mathcal{R}(\tilde{y}), \min_{1 \leq k \leq K} \mathcal{R}(\bar{y}_k)\right\}. \tag{8}$$

For the second inequality, observe that any feasible solution for the $(K-1)$-component problem can be extended to a feasible solution for the K-component problem by appending a zero weight for the new component. Specifically, if ω_{K-1}^* is optimal for $K-1$, then $\omega^{(K-1)} = (\omega_{K-1,0}^*, \ldots, \omega_{K-1,K-1}^*, 0)$ is feasible for K and yields $\hat{y}_K^{(K-1)} = \hat{y}_{K-1}^*$. By optimality of ω_K^*:

$$\mathcal{R}(\hat{y}_K^*) \leq \mathcal{R}(\hat{y}_K^{(K-1)}) = \mathcal{R}(\hat{y}_{K-1}^*). \tag{9}$$

Both results hold for any risk function $\mathcal{R}$, as they rely only on the structure of the feasible region and the principle of optimality. □

Proposition 1 provides crucial theoretical support for the robustness of the proposed label correction mechanism. Its implications are multifaceted:

1. **Bounded Risk:** The first inequality, $\mathcal{R}(\hat{y}_K^*) \leq \min\left\{\mathcal{R}(\tilde{y}), \min_{1 \leq k \leq K} \mathcal{R}(\bar{y}_k)\right\}$, guarantees that the expected risk of the optimally combined corrected label $\hat{y}_K^*$ is no worse than using the original noisy label $\tilde{y}$ directly, nor is it worse than relying on any single prediction $\bar{y}_k$ from the correction network r_k (if one could identify the best $\bar{y}_k$ in advance).
2. **Monotonically Non-Increasing Risk:** The second inequality, $\mathcal{R}(\hat{y}_K^*) \leq \mathcal{R}(\hat{y}_{K-1}^*)$ for $K \geq 1$, demonstrates that as more historical predictions $\bar{y}_k$ are incorporated into the ensemble, the expected risk of the resulting optimally weighted label $\hat{y}_K^*$ does not increase. This is a key factor for the stability of the iterative self-training process.

These properties are vital for preventing error amplification, especially when individual corrected predictions $\bar{y}_k$ might be suboptimal. The optimization of weights ω^* on $\mathcal{D}_{\text{valm}}$ (as per Eq. (5)) allows the framework to adaptively assign importance to each component, effectively down-weighting less reliable or even detrimental predictions. It is important to recognize that the strength of this framework lies in its ability to handle diverse qualities of $\bar{y}_k$ components. For instance, even if a particular $\bar{y}_k$ represents a very uncertain or "average" prediction (e.g., a uniform distribution across classes), the optimization of ω^* will determine its appropriate contribution. If incorporating such a component, despite its inherent uncertainty, aids in regularizing or smoothing the final $\hat{y}_K^*$ relative to other potentially overconfident or erroneous labels (like $\tilde{y}$ or other $\bar{y}_j$), it can be assigned a beneficial weight ω_k^*. This means the convex combination, guided by the risk minimization objective of Proposition 1, inherently performs a type of label smoothing. This reduces the variance of the effective training targets by producing a more stable and less noisy signal, thereby contributing to the overall robustness and performance of the learning process. The resulting $\hat{y}_K^*$ thus not only benefits from a bounded and non-increasing risk profile but also from these implicit regularization effects.

A.3 Computational Considerations for Correction Mechanism Optimization

A pertinent consideration is the computational cost associated with optimizing the parameters of the label correction mechanism (i.e., ϕ for the neural corrector r and ω for the convex combination, as detailed in Section 4.2 and Section 4.2 of the main paper). Although this introduces an inner optimization loop within the broader self-training iterations, the incurred time overhead is minimal. This efficiency is largely attributable to two key aspects: first, the clean meta-dataset $\mathcal{D}^{\mathrm{m}}$, on which these parameters are optimized, is typically substantially smaller than the primary noisy training dataset $\tilde{\mathcal{D}}$. Second, the correction network r often operates on feature embeddings z

(with dimensionality d'), which are of significantly lower dimensionality than the raw input data x (with dimensionality d). Consequently, the optimization steps for r and ω are computationally inexpensive. As empirically validated in our experiments, our overall approach demonstrates favorable computational efficiency, particularly when contrasted with some meta-learning strategies that may involve more complex and resource-intensive computations, such as those requiring second-order derivatives or extensive unrolled optimization paths.

A.4 Summary of Methodological Advantages

Our approach, when compared to existing Learning with Noisy Labels (LNL) methods, particularly those leveraging a small clean dataset, offers several distinct advantages:

Computational Efficiency The decoupled optimization strategy employed in our framework circumvents the need for computationally expensive operations, such as the high-order derivative calculations often required by meta-gradient techniques. This design, complemented by feature reuse from the primary classifier and potentially less frequent updates to the correction mechanism, substantially minimizes computational overhead. Consequently, our method exhibits enhanced scalability and is well-suited for deployment in distributed training environments.

Flexibility and Adaptability The neural correction network, r, is inherently capable of modeling complex, instance-dependent noise distributions, thereby surpassing the limitations of methods reliant on simpler, often global, noise assumptions (e.g., noise transition matrices). This architectural flexibility allows our framework to be potentially adapted to more diverse problem settings, including continuous or multi-label classification tasks where transition matrices are typically less suitable. Furthermore, the training of the correction network r is designed to be lightweight, facilitating its use as a plug-and-play module and readily allowing the incorporation of new domain knowledge for enhanced label refinement.

Robustness The synergistic interplay between the learned neural corrector r (designed to offer refined, potentially lower-variance label predictions) and the generalized robust convex combination strategy (as detailed in Section 4.2 of the main paper) fosters stable and reliable learning dynamics. This robustness is not merely heuristic; it is underpinned by theoretical risk guarantees (Proposition 1) and is continually fortified by the iterative improvement mechanism that is anchored to the ground-truth information within the clean dataset.

B Limitations and Broader Impact

Limitations Our framework, while effective, has certain limitations. Firstly, the default formulation where the set of historical features $\{z_{k,i}\}_{k=0}^{t}$ for the correction network r_t and the set of historical corrected predictions $\{\bar{y}_{k,i}\}_{k=1}^{t}$ for the robust convex combination grow with iteration t could lead to increasing computational and memory demands per iteration over very long training runs. However, adopting a fixed-size window of the most recent historical data, a strategy whose potential was highlighted in our feature input ablations (Section 5.1), would mitigate this. Such an approach would make the architecture of r_t consistent across iterations, thus enabling warm-start to facilitate more efficient updates. Secondly, if f_n poorly approximates the true noise distribution $p(\tilde{y}|x)$, the simulated noisy posteriors $\hat{\tilde{y}}_i^m$ become inaccurate representations of the noise characteristics relevant to $\tilde{\mathcal{D}}$. Consequently, this mismatch between the simulated noise context (used for learning ω_t^*) and the actual noise context (where ω_t^* is applied) can lead to suboptimal corrected labels $\hat{y}_{t+1}$, thereby limiting overall classifier performance. Other limitations include the dependency on a small, genuinely clean dataset $\mathcal{D}^{\mathrm{m}}$ and the presence of several hyperparameters that may require tuning for new datasets or noise characteristics.

Broader Impact Our method offers several significant broader impacts. By effectively relabeling training samples, our method directly reduces label noise and variance within the training set, leading to more robust model training and improved generalization. This inherent label correction capability suggests that our framework could be compatible with, and potentially serve as an enhancement for, other LNL strategies, for example, by providing cleaner labels for their subsequent processing stages. The closed-loop, iterative nature of our framework, where the classifier and correction function

Table 5: Experimental details for our method across different datasets.

Dataset	CIFAR-10	CIFAR-100	Clothing1M
Architecture	ResNet-34	ResNet-34	ResNet-50
Pretraining	None	None	ImageNet
Training Epochs	100	100	15
Training Augmentations	Horizontal Flip Random Crop	Horizontal Flip Random Crop	Resized Crop Horizontal Flip Random ColorJitter Random Grayscale
Optimizer	SGD	SGD	SGD
Batch Size	128	128	256
Momentum	0.9	0.9	0.9
Weight decay	$5e^{-4}$	$5e^{-4}$	$5e^{-4}$
Initial LR	0.1	0.1	0.01
Scheduler	LR-step	LR-step	LR-step
Scheduler Milestone	$60, 80$	$60, 80$	$5, 10$
Scheduler Step-Decay	0.1	0.1	0.1
Label Correction Frequency	5	5	3
Clean Dataset Split Ratio	0.8 : 0.2		
Correction Network Type	2-layer MLP		
Correction Network Hidden Units	256		
Correction Network Activation	ReLU		
Correction Network Optimizer	SGD		
Correction Network LR	$1e^{-3} \rightarrow 1e^{-4}$		

Table 6: Time consumption breakdown (in hours) of our method's different phases on CIFAR datasets using a single RTX 4090.

Phase	Classifier Training	Feature Extraction & Simulation	Train Neural Label Correction	Train Convex Combination	Noisy Label Update	Total
Total Time	0.327	0.014	0.004	0.001	0.002	0.348
(Execution Times)	(100×)	(12×)	(12×)	(12×)	(12×)	

co-evolve under the continuous guidance of clean data (via $\mathcal{D}^{\mathrm{m}}$), inherently limits the accumulation of errors and reduces the risk of model collapse, a common challenge in many self-correction or pseudo-labeling schemes.

Beyond standard LNL scenarios, our approach may offer promise for certain transfer learning settings. For instance, it could potentially establish a more robust mapping between a source and a target domain by correcting noisy or mismatched labels in the source domain, guided by feedback from a small, clean dataset representative of the target domain. This would effectively refine the source data, akin to adaptively improving sample quality for more effective knowledge transfer.

C Further Details on the Experiments

We will publicly release our code and implementation details upon acceptance of this paper. The repository will include all experiments conducted on CIFAR and Clothing1M datasets, enabling full reproducibility of our results. In the following sections, we provide additional experimental details, comprehensive analyses, and extended results that complement the main paper.

We detail the specific implementation aspects and hyper-parameters used in our experiments across various datasets. Table 5 outlines the key training parameters and settings employed in our experiments.

Table 7: Test accuracy (%) of our method using different values of loss balancing factor λ on CIFAR-100 under different noise types with different noise levels. The last recorded performances in the training procedure are reported.

λ value	Symmetric			Asymmetric			Instance-dependent		
	η			η			η		
	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
0.2	**69.11**	**62.77**	50.06	**70.32**	**67.15**	62.63	68.95	61.53	53.38
0.5	67.34	62.04	**51.38**	70.21	66.21	**63.75**	**69.67**	**62.38**	**53.73**
1.0	67.03	61.07	49.30	70.20	66.49	62.90	68.20	61.54	53.56

Table 8: Test accuracy (%) of our method using different label correction frequencies on CIFAR-100 under different noise types with different noise levels. The frequency value indicates how often (in epochs) the feature sets and correction function are updated.

Label Correction Frequency	Symmetric			Asymmetric			Instance-dependent		
	η			η			η		
	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
5	67.34	**62.04**	51.38	70.21	66.21	**63.75**	69.67	**62.38**	53.73
8	**68.50**	60.41	**52.53**	**70.35**	**67.63**	62.19	**69.76**	61.04	**54.16**
10	67.93	60.04	51.65	69.84	67.40	62.06	69.47	60.97	53.30

C.1 Experiments on CIFARs

Our approach consists of five main phases: (1) **Classifier Training**, where we update the model parameters using corrected labels; (2) **Feature Extraction & Noisy Posterior Simulation**, which involves extracting feature representations from both datasets and simultaneously generating the simulated noisy posteriors on the clean dataset; (3) **Neural Correction Optimization**, which includes training and validating the correction network; (4) **Convex Combination**, where we compute optimal weights for combining noisy labels and correction network predictions; and (5) **Label Update**, where we apply the computed weights to generate the refined corrected labels for the subsequent training iteration.

In Section 5.1, we update the feature sets and correction function every 5 epochs, starting after the initial 40 epochs. As shown in Table 6, the computational overhead associated with our correction mechanism (phases 2-5) is not significant. This confirms that our method achieves significant performance improvements without introducing substantial computational burden. While the neural correction optimization remains computationally light, the classifier training phase dominates the overall time consumption, which is consistent with standard deep learning training procedures.

Ablations on the Loss Balancing Factor In addition, we also conduct experiments to examine how the loss balancing factor λ affects performance. Selecting the appropriate λ in Eq. (1) is a balancing act: it should enable the classifier to estimate the noisy posterior accurately, while preventing the feature extractor from overfitting to noisy patterns. We vary λ across different values in $\{0.2, 0.5, 1.0\}$, with the results summarized in Table 7. The table shows that the impact of different λ values is relatively modest, with performance variations typically within 1-2 percentage points across all noise types and levels.

Ablations on the Label Correction Frequency We investigate the impact of label correction frequency on model performance. The correction frequency determines how often we update the feature sets and correction function during training. As described in Section 5.1, our default setting updates every 5 epochs after the initial 40 epochs on CIFAR datasets. We vary this frequency across different values in $\{5, 8, 10\}$ to examine its influence on performance. The results are summarized in Table 8. The experimental results demonstrate that the performance differences are relatively modest, with variations typically within 1-2 percentage points. This suggests that our method maintains robust performance across a range of correction frequencies, though more frequent updates tend to provide

Table 9: Test accuracy (%) on CIFAR-100 with only 1000 clean samples. The results also demonstrate the stabilizing effect of the convex combination in this low-data regime.

Method	Symmetric			Asymmetric			Instance-dependent		
	η			η			η		
	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
w/o convex	59.75	54.01	42.26	61.35	56.74	53.72	60.31	52.81	43.98
Ours (w/ convex)	**66.17**	**58.34**	**46.77**	**67.97**	**61.23**	**54.14**	**67.01**	**57.26**	**45.97**

Table 10: Test accuracy (%) on CIFAR-100 with 5000 clean samples. The convex combination continues to act as an effective regularizer, leading to stable performance gains.

Method	Symmetric			Asymmetric			Instance-dependent		
	η			η			η		
	0.2	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
w/o convex	66.93	60.68	**52.03**	69.24	**67.33**	63.19	69.34	60.40	53.10
Ours (w/ convex)	**67.34**	**62.04**	51.38	**70.21**	66.21	**63.75**	**69.67**	**62.38**	**53.73**

incremental benefits, particularly in high-noise scenarios where label quality is more critical to model convergence.

Sensitivity to Clean Data Quantity A crucial aspect of our framework is its ability to perform robustly even with very limited clean supervision. To test this, we conducted an extensive ablation on CIFAR-100 under two conditions: a standard setting with 5,000 clean samples (and 45,000 noisy samples) and a data-scarce setting with the clean set reduced by 80% to only 1,000 samples (and 49,000 noisy samples). The results are detailed in Table 9.

The experiments show that our framework exhibits graceful degradation, not catastrophic failure. Even when the clean set is drastically reduced, our full method maintains strong, competitive performance. This affirms its practical applicability in real-world scenarios where clean data is expensive and scarce. The performance drop is modest, indicating that the framework can effectively distill knowledge from even a tiny set of trusted labels to guide the correction process across a much larger noisy dataset.

The Role of the Convex Combination Our theoretical analysis posits that the robust convex combination is key to preventing overfitting that will cause error amplification. The following ablation experiments provide strong empirical validation for this claim. We compare our full framework ("Ours w/ convex") against a setting where the convex combination is removed, and labels are corrected using only the raw output of the neural correction network ("w/o convex").

As shown in Table 9 and Table 10, the convex combination is a critical stabilizer, especially in the low-data regime. With only 1,000 clean samples, its removal causes a severe performance drop . This validates our claim that it provides an essential safety net, preventing the model from being misled by a correction network prone to overfitting on the small clean set. When more clean data is available (Table 10), it continues to act as an effective regularizer, smoothing the learning process and consistently delivering performance gains.

This confirms that the robust convex combination is a highly adaptive core component of our framework, creating value by ensuring stability across different scales of clean data supervision.

Complete Results Presentation We present experimental results for various datasets under different noise conditions on CIFAR-10 and CIFAR-100. Fig. 6 and Fig. 7 show the test accuracy trends across training epochs. The dashed vertical lines at epochs 60 and 80 indicate learning rate drop points. The figures demonstrate the effectiveness of our method compared to existing approaches across different datasets and noise settings. As shown in the plots, our method consistently achieves higher test accuracy, especially in high noise ratio scenarios (60%), where the performance gap is more significant. Our method exhibits more stable learning behavior and better convergence properties.

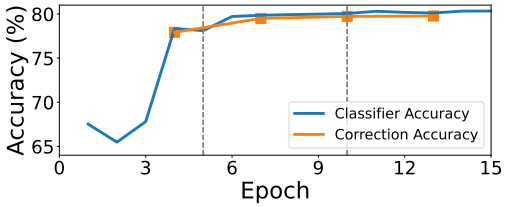

Figure 4: Classification accuracy on test data and clean meta data during training on Clothing1M. The vertical dashed lines represent the epochs at which the learning rate decays at 5 and 10 epoch.

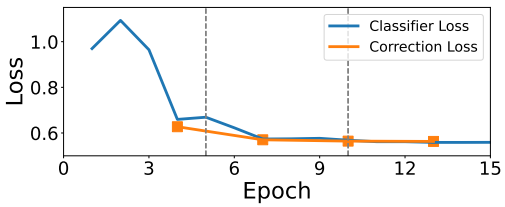

Figure 5: Classification error on test data and clean data during training on Clothing1M. The vertical dashed lines represent the epochs at which the learning rate decays at 5 and 10 epoch.

Table 11: Comparison of average memory usage and training time per epoch for our method, L2B, and EMLC on Clothing1M under the specified training configurations. We calculate our training time as the average (avg) by dividing the total training time by the complete 15 epochs of our experiments.

Method	GPU Memory Cost (MiB)	Training Time per Epoch (hours)
L2B	24120	3.47
EMLC	22190 / 21813	3.00
Ours	**16915** (no additional cost)	**0.56** (avg)

C.2 Experiments on Clothing1M

Training Dynamics of Our Method on Clothing1M Fig. 4 and Fig. 5 illustrate the performance of our method during the training on Clothing1M. In these figures, we measure the performance on the test data and the clean data in $\mathcal{D}^{\mathrm{m}}_{\mathrm{val}}$ as the correction neural network is not significantly overfitted on this validation subset. We observe steady decreases in the classification error in the test dataset and $\mathcal{D}^{\mathrm{m}}_{\mathrm{val}}$, demonstrating the robustness of our method.

Training Efficiency Comparison In order to validate our potential in practical usage, we test the training efficiency for the advanced methods, mainly the meta-learning-based methods, on Clothing1M. We set up comparative analysis of our method, L2B and EMLC regarding their time cost and performance. We unify the classifier backbone as ResNet-50. Our method and L2B utilize a single RTX 4090 GPU with 128 mini-batches, whereas EMLC employs a 2-GPU setup, with each GPU processing 64 mini-batches. This configuration ensures a fair basis for comparison.

The training resource comparison is presented in Table 11. Notably, our framework demonstrated superior efficiency. Additionally, our method used less memory, with EMLC consuming more than twice the memory of ours. These results highlight the advantages of our approach in terms of performance, time efficiency, and GPU memory usage, underscoring its potential for practical applications.

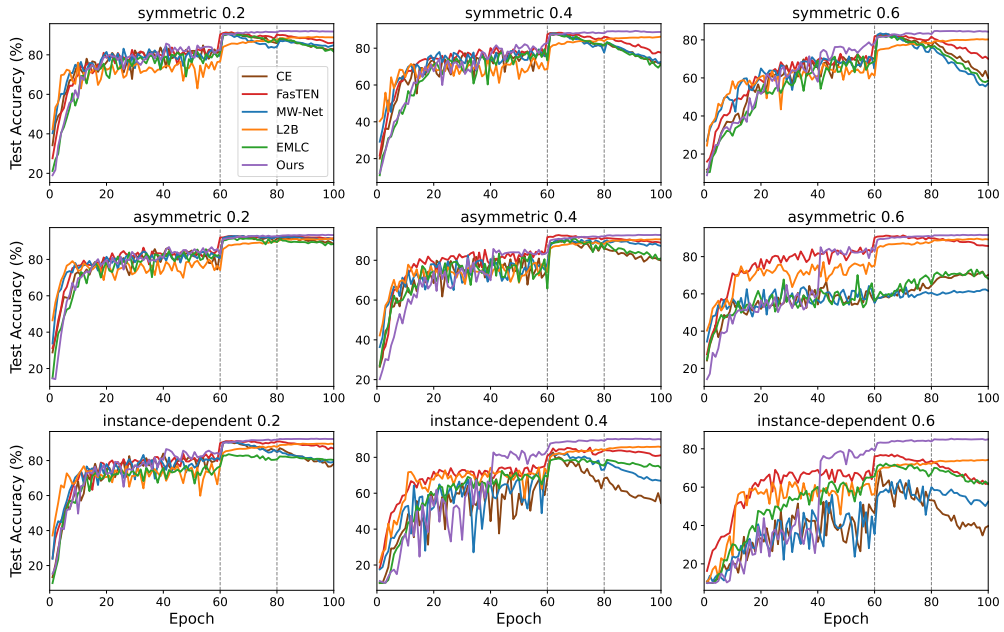

Figure 6: Training dynamics of all methods on CIFAR-10 under different noise scenarios.

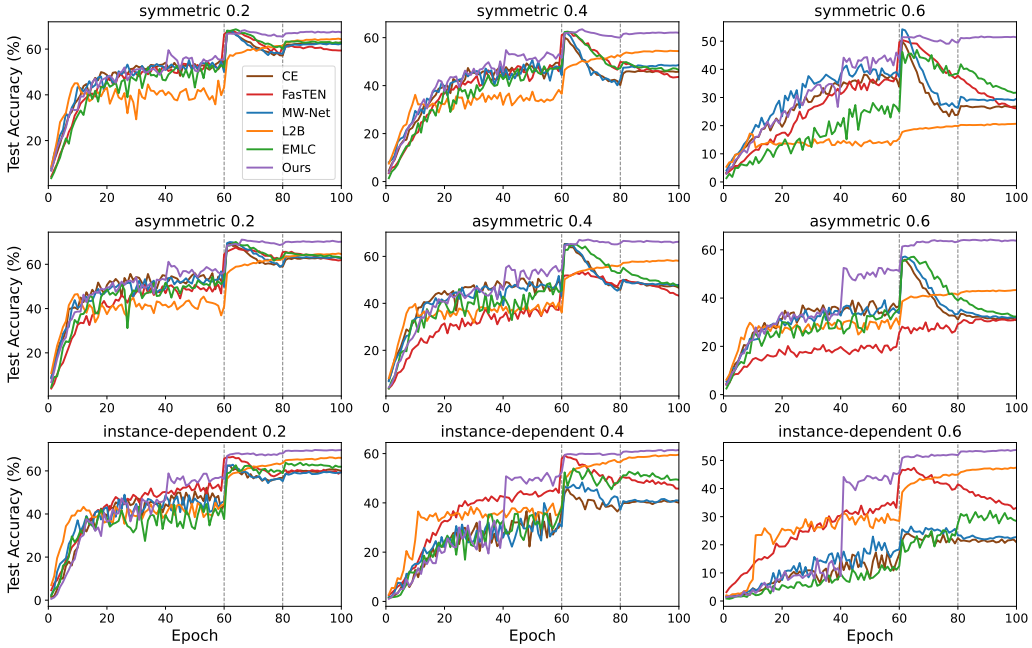

Figure 7: Training dynamics of all methods on CIFAR-100 under different noise scenarios.