SAPIENZA
UNIVERSITÀ DI ROMA

Stopping the Noise: A Review of Early Stopping Techniques for Learning with Noisy Labels

Faculty of Information Engineering, Informatics, and Statistics
Master in Engineering in Computer Science

Davood Sheibani
ID number 2126056

Advisor
Prof. Fabrizio Silvestri

Co-Advisors
Dr. Maria Sofia Bucarelli
Dr. Farooq Wani

Academic Year 2024/2025

Thesis defended on 23 July 2025
in front of a Board of Examiners composed by:

Prof. Fabrizio Silvestri (chairman)
Prof. Danilo Comminiello
Prof. Lenti Simone
Prof. Paolo Liberatore
Prof. Roberto Navigli
Prof. Laura Palagi
Prof. Paolo Russo
Prof. Ivan Visconti

Stopping the Noise: A Review of Early Stopping Techniques for Learning with Noisy Labels
Master Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: sheibani.2126056@studenti.uniroma1.it

To all the curious minds.

Acknowledgements

I would like to express my deepest gratitude to Prof. Fabrizio Silvestri, Dr. Maria Sofia Bucarelli, and Dr. Farooq Wani. Their guidance, support, and belief in my work have been fundamental throughout this journey. Each of them, in their own way, has shaped my understanding, challenged my thinking, and encouraged me to grow as a researcher. I am truly thankful for their time, patience, and inspiration.

Abstract

Deep learning has revolutionized the landscape of artificial intelligence, achieving state-of-the-art performance across a wide range of applications such as computer vision, natural language processing, and speech recognition. A key factor behind this success is the availability of large-scale datasets with human-annotated labels that enable deep neural networks (DNNs) to learn complex patterns. However, in real-world scenarios, the quality of labels is often compromised due to factors such as human error, crowd-sourced annotation, weak or automated labeling strategies, and even adversarial interference. These limitations make it infeasible to consistently collect perfectly labeled data at scale, resulting in label noise—incorrect or corrupted annotations—that can significantly degrade model performance by leading to overfitting and poor generalization.

Learning from noisy labels, also known as robust training, has therefore emerged as a critical challenge in modern deep learning. DNNs have a strong capacity to memorize even incorrect labels, which leads to overfitting to noise and hinders their ability to generalize to unseen data. Addressing this challenge requires a thorough understanding of how label noise affects model training and a systematic investigation of robust learning techniques. This thesis begins by formalizing the problem of learning with label noise from a supervised learning perspective, where noisy labels are considered part of the training set and strategies are developed to mitigate their harmful effects.

The core contribution of this work is a comprehensive analysis of state-of-the-art methods that incorporate early stopping or related strategies to avoid memorization of noisy labels. These methods are examined through a unified lens that considers specific design features—such as network architecture, regularization strategies, learning dynamics, and dependence on auxiliary information—captured in a structured feature-based taxonomy.

In addition to architectural and algorithmic features, this thesis introduces a complementary empirical cue analysis. This second layer of evaluation focuses on the behavioral signals that methods rely on during training—such as small-loss heuristics, prediction consistency over time, or feature-space regularities—to identify or mitigate the effect of noisy labels. The integration of both structural and empirical perspectives provides a richer understanding of how robust training methods operate, and why they succeed under different conditions.

The reviewed methods are further analyzed to identify recurring design decisions and key divergences, with the aim of understanding which combinations of features and cues are most effective under varying noise regimes. This dual analysis supports a principled comparison of approaches, offering practical insights that go beyond static taxonomies and laying the foundation for evaluating and improving robust learning methods.

To support and validate this analysis, the thesis incorporates a practical experimental framework designed for systematic benchmarking of selected robust training methods. The framework emphasizes extensibility and reproducibility, allowing new models to be integrated and evaluated consistently. It also includes essential components such as noise simulation, evaluation metrics, and benchmark datasets.

Finally, the thesis outlines open research directions, such as integrating early stopping with self-supervised or semi-supervised learning, and designing adaptive strategies that respond dynamically to learning signals. The overall goal is not only to consolidate current understanding but to provide a conceptual and practical foundation for developing more robust, generalizable, and interpretable deep learning systems in the presence of label noise.

Contents

Chapter 1

Introduction

Deep learning has become a transformative force in artificial intelligence, enabling breakthroughs in tasks such as image classification, speech recognition, and natural language processing [29, 20, 52, 14, 6, 13, 92]. This success is largely attributed to the combination of large-scale annotated datasets and deep neural networks (DNNs), which possess a strong capacity to learn complex hierarchical patterns from raw input data. However, this dependence on vast amounts of high-quality labeled data introduces a significant challenge in real-world applications, where annotations are often noisy or unreliable.

In many practical scenarios, data labeling is outsourced to non-experts via crowdsourcing platforms such as Amazon Mechanical Turk [71], generated through automated heuristics or weak supervision [50, 15, 38], or obtained through web scraping, where labels are inferred indirectly [73, 84]. These strategies are cost-effective but inherently error-prone. The resulting label noise—defined as incorrect or corrupted class annotations—can lead to severe degradation in the generalization performance of deep networks [99, 3, 68]. Despite the use of regularization techniques like data augmentation [75], dropout [72], and weight decay [30], DNNs tend to memorize noisy labels over time, compromising their ability to learn meaningful patterns [3].

This growing concern has led to the emergence of robust learning from noisy labels as a dedicated research field. Robust training methods aim to maintain generalization performance despite the presence of label noise. A wide array of strategies have been proposed, including loss correction techniques [58], robust loss functions [48], sample selection methods [20], and hybrid approaches incorporating semi-supervised learning [32].

This thesis contributes to the evolving field of robust learning by focusing on a specific subset of strategies: those that utilize early stopping or related mechanisms to avoid memorization of noisy labels. The central objective is to analyze how these methods exploit the temporal dynamics of deep learning—such as the early-learning phenomenon or prediction consistency—to halt training before overfitting to noise occurs. To achieve this, the analysis goes beyond traditional method categorization and adopts a dual-framework approach. The first layer is a feature-based taxonomy, which examines each method across dimensions such as architectural design, regularization techniques, learning dynamics, and dependence on auxiliary

supervision. Complementing this is a second layer of empirical cue analysis, which identifies the behavioral heuristics these methods rely on—such as small-loss selection, confidence signals, or consistency across augmentations—to infer label correctness during training. Together, these two perspectives provide a comprehensive, fine-grained understanding of what makes early-stopping-based methods effective and how their design choices influence robustness to label noise.

1.1 Background and Motivation

The availability of large annotated datasets has long been considered a cornerstone of modern supervised learning. However, the process of acquiring accurate labels at scale is often infeasible due to resource constraints, subjective interpretation, or ambiguous class definitions. In practice, labels are frequently obtained through noisy processes such as heuristic-based auto-labeling, social tagging, or third-party annotation services [15, 38, 73]. The resulting label noise introduces systematic and random errors that compromise the reliability of the training signal.

Deep neural networks are particularly vulnerable to such noise due to their high model capacity and non-convex optimization behavior. Unlike traditional shallow models, which tend to average out minor inconsistencies, DNNs can memorize even large portions of corrupted labels [3]. This memorization behavior has been well documented in the literature [99, 68] and poses a major threat to generalization, especially when label noise rates are moderate to high.

Robust training is thus not merely a supplementary technique, but a necessity for reliable deployment of deep learning systems in realistic environments. Numerous empirical studies have shown that without noise-aware learning strategies, model performance deteriorates rapidly as the noise level increases [84, 20]. Furthermore, the type of label noise—whether symmetric, asymmetric, or instance-dependent—affects the training dynamics in fundamentally different ways [58, 20, 93].

Among the many robust training strategies, early stopping has emerged as a promising direction. It is motivated by the observation that DNNs tend to learn clean patterns in the early phase of training, while memorization of incorrect labels occurs later [3, 32]. Leveraging this temporal distinction, early stopping methods attempt to interrupt the training process before memorization dominates, thereby preserving the model's generalizability.

Given the diversity of robust learning methods, this thesis proposes a comparative framework that examines early stopping strategies through both a feature-based taxonomy and an empirical cue analysis. By dissecting architectural choices, regularization mechanisms, and behavioral patterns observed during training, it reveals common principles underlying effective approaches and offers practical guidance for designing noise-resilient learning systems.

1.2 The Challenge of Label Noise in Deep Learning

In the era of data-driven machine learning, the reliability of supervised learning algorithms is fundamentally tied to the quality of the labeled data used during training. While deep neural networks (DNNs) have shown impressive capabilities in

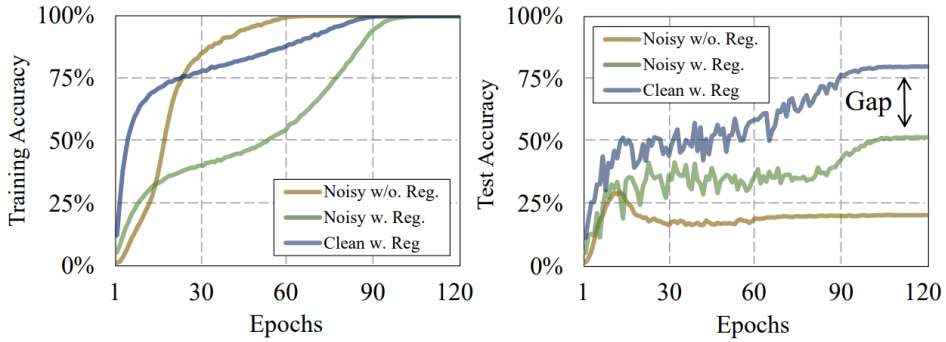


Figure 1.1. Convergence curves of training and test accuracy when training WideResNet-16-8 on CIFAR-100 with 40% symmetric label noise. "Noisy w/o. Reg." and "Noisy w. Reg." denote models trained on noisy data without and with regularization, respectively, while "Clean w. Reg." indicates training on clean data with regularization. Adapted from Song et al. (2022), “Learning from Noisy Labels with Deep Neural Networks: A Survey [70]

tasks such as image recognition [29], language understanding [14], and autonomous systems [6], these models depend heavily on large-scale datasets annotated with correct ground-truth labels. However, in practical settings, acquiring perfectly labeled data is rarely feasible. Human annotation is expensive and error-prone, and alternatives such as crowdsourcing [71], automated labeling [50], or heuristic methods [73] often introduce systematic or random label inaccuracies.

This phenomenon, known as label noise, has emerged as a key obstacle in the successful deployment of deep learning models. Label noise can take various forms—ranging from uniformly random noise, to asymmetric class-flipping, to highly complex instance-dependent noise, where the likelihood of mislabeling depends on the input features [58, 93]. Such noise not only dilutes the training signal but actively misguides the optimization process. As a result, learning becomes unstable, and the final model exhibits poor generalization on clean, unseen data.

What makes label noise particularly challenging in deep learning is the model’s ability to memorize. Deep networks are highly expressive and capable of fitting even completely random labels when trained long enough [3]. While early phases of training tend to prioritize clean examples, over time DNNs begin to fit noisy data points, leading to degraded performance [68, 99]. This memorization phenomenon undermines traditional regularization strategies such as dropout [72], weight decay [30], data augmentation [75], and batch normalization [26], which, although widely adopted, are often insufficient to prevent overfitting to corrupted labels.

Unfortunately, even when these regularization techniques are extensively applied, deep networks still exhibit a notable gap between training on clean versus noisy data. As shown in Figure 1.1, models trained on noisy labels achieve significantly lower test accuracy compared to those trained on clean labels, despite using regularization. Moreover, label noise has been observed to degrade model performance more severely than other forms of noise, such as input noise [84]. Thus, achieving strong generalization in the presence of label noise remains a major and unresolved challenge in deep learning.

Moreover, empirical studies have demonstrated that noise can significantly distort the learned feature representations within a model, especially when the noise rate is high [84]. This not only affects classification performance but also harms downstream applications such as transfer learning and semi-supervised learning.

A key difficulty in dealing with label noise is that it is typically unobservable—models must infer which labels are likely corrupted based only on indirect signals such as prediction confidence, loss trajectories, or neighborhood consistency. Strategies that attempt to identify clean data dynamically during training—such as small-loss sample selection [20]—have shown promise, but remain limited when the noise is complex or heavily instance-dependent.

These challenges motivate the need for learning algorithms that are inherently robust to noise or can adaptively identify and mitigate its effects during training. Among such strategies, early stopping has emerged as a simple yet effective approach: by terminating training before memorization occurs, the model retains the structure learned from cleaner signals [3, 32]. However, determining the right moment to stop—and doing so consistently without clean validation data—remains an open research question and is central to the methods explored in this thesis.

1.3 Objectives and Scope of the Thesis

The presence of label noise in training datasets remains one of the primary obstacles to achieving reliable and generalizable deep learning models. Despite substantial progress in understanding how deep networks learn [3, 94], mitigating the negative effects of corrupted labels continues to pose significant theoretical and practical challenges.

The primary objective of this thesis is to systematically study and analyze methods that leverage early stopping and related strategies to counteract the memorization of noisy labels. Recognizing that deep neural networks tend to first learn simple and clean patterns before eventually overfitting to noisy labels [3], this thesis aims to exploit this dynamic by identifying methods that prevent memorization while maintaining strong generalization capabilities.

Unlike previous surveys or reviews that categorize methods based on algorithmic strategies alone [16, 19], This thesis introduces a feature-based taxonomy (chapter 4) and an empirical cue analysis (chapter 5) to describe and compare robust training methods. This fine-grained perspective allows for a deeper comparison of how different methods tackle label noise, moving beyond surface-level classifications. By analyzing these underlying design choices, the study aims to reveal patterns that contribute to greater robustness against noisy labels.

The custom taxonomy proposed in this thesis defines eleven specific features, grouped under four main dimensions.

The feature-based view captures fundamental architectural, algorithmic, and data-dependence choices across robust training methods, enabling a more detailed comparison than traditional taxonomies. These features are analyzed in detail in subsequent chapters, offering a deeper insight into their role in shaping robust learning strategies under label noise.

To fulfill these objectives, the scope of the thesis encompasses several key com-

ponents:

- **Theoretical Analysis**: Formalizing the problem of supervised learning with label noise, characterizing different types of noise (symmetric, asymmetric, instance-dependent) [58, 93], and explaining the memorization dynamics of deep neural networks trained under noisy supervision [3, 94].
- **Method Review and Dual-Lens Analysis**: Conducting an in-depth review and analysis of state-of-the-art robust training techniques, selected for their relevance to early stopping and noise-robust learning. Each method is systematically examined through both a feature-based taxonomy and an empirical cue framework, highlighting their structural design choices and training-time behavioral signals.
- **Experimental Framework Development**: Building a modular, extensible benchmarking framework for consistent evaluation of selected methods under synthetic noise settings.
- **Empirical Evaluation and Feature Impact Analysis**: Performing experiments across different noise types and analyzing how specific design features impact performance.
- **Identification of Open Challenges and Research Opportunities**: Discussing the limitations of current methods and proposing future directions, such as integrating self-supervised learning or adaptive early stopping mechanisms [32, 32].

The goal is not only to deepen the understanding of robust learning techniques but also to offer practical insights and guidance for designing noise-resilient deep learning systems that can be deployed in real-world environments where label quality cannot be guaranteed.

1.4 Structure of the Thesis

This thesis is organized into eight chapters, each designed to progressively build a comprehensive understanding of the challenges and solutions associated with learning from noisy labels in deep neural networks.

- **Chapter 1: Introduction**
  Provides an overview of the problem of noisy labels, outlines the objectives, scope, and contributions of the thesis, and introduces the feature-based taxonomy that underpins the analysis.
- **Chapter 2: Background and Related Work**
  Reviews the fundamentals of supervised learning, the impact of label noise, classical approaches to noise-robust learning, early stopping strategies in deep learning, and recent surveys on the subject.

- **Chapter 3: Problem Definition and Methodological Foundations**
  Formalizes the problem of learning under noisy supervision, defines key types of label noise (symmetric, asymmetric, instance-dependent), and explains theoretical insights into the early-learning and memorization behavior of deep neural networks.
- **Chapter 4: Feature-Based Taxonomy for Robust Training Methods**
  Introduces and motivates the custom feature-based taxonomy. Each feature is clearly defined and justified,
- **Chapter 5: Empirical Cue Analysis**
  Introduces a complementary perspective based on training-time behavioral cues such as small-loss criterion, prediction consistency, augmentation robustness, and gradient coherence. Cues are defined and illustrated.
- **Chapter 6: In-Depth Analysis of Selected Methods**
  Reviews and compares robust learning methods using both the feature-based taxonomy and empirical cue analysis. For each method, structural traits and training cues are identified, compared, and tabulated.
- **Chapter 7: Experimental**
  Describes the design and implementation of a modular experimental framework. Experimental protocols, covering dataset selection, noise rate estimation protocols, and evaluation metrics.
- **Chapter 8: Conclusions and Future Work**
  Summarizes key insights, and outlines future directions.

Chapter 2

Preliminaries and Related Work

2.1 Fundamentals of Supervised Learning and Label Quality

Supervised learning aims to learn a function $f : X \to Y$ that maps input data $x \in X$ to target outputs $y \in Y$, given a labeled training set:

$$D = \{(x_i, y_i)\}_{i=1}^{n}$$

where (x_i, y_i) pairs are assumed to be independently drawn from an underlying distribution $P(X, Y)$. A core assumption underpinning the success of supervised deep learning models [29, 60, 14] is the availability of reliable, accurately annotated labels.

In practical applications, however, this assumption is frequently violated. Factors such as human annotation errors [46], variability in crowdsourced labeling [57, 51], automated labeling processes [35], and adversarial label flips [85] contribute to corrupted supervision. As a result, the training set is better represented as:

$$\tilde{D} = \{(x_i, \tilde{y}_i)\}_{i=1}^{n}$$

where $\tilde{y}_i$ denotes a noisy version of the true label y_i.

Modeling label noise is central to robust learning. Two predominant noise structures are often considered:

- **Class-Conditional Noise (CCN)**: The noise process depends only on the true class label, independent of the instance features.
- **Instance-Dependent Noise (IDN)**: The noise probability varies as a function of the input instance itself [83].

The presence of label noise is particularly detrimental because deep neural networks (DNNs), due to their high capacity, can memorize corrupted labels without difficulty [3, 94]. This memorization behavior contrasts with robustness against input perturbations [65, 26], where standard regularization techniques are often sufficient. In contrast, label noise directly distorts the learning signal, posing a more fundamental challenge [101].

Understanding the mechanisms by which label noise affects training dynamics is essential for developing noise-robust algorithms. The subsequent sections build upon these definitions to survey existing techniques and identify key research directions in learning from noisy supervision.

2.2 Impact of Label Noise on Deep Neural Networks

Deep neural networks (DNNs) have demonstrated extraordinary capacity for modeling complex data distributions, achieving outstanding performance in domains such as computer vision, natural language processing, and recommendation systems [29, 60, 96, 56, 54, 23, 14, 64]. However, their high expressiveness also makes them particularly vulnerable to the presence of label noise during supervised training.

Unlike classical models, which often underfit noisy data, DNNs are capable of memorizing even completely random labels when trained for sufficient epochs [3, 94]. This behavior significantly compromises generalization performance. While in the early stages of training networks tend to learn meaningful patterns, prolonged exposure to noisy supervision eventually causes the models to overfit to corrupted labels, leading to performance degradation on clean unseen data [3].

Standard regularization techniques—such as data augmentation [65], weight decay [30], dropout [72], and batch normalization [26]—can alleviate overfitting in clean-label settings. However, when label noise is present, these methods are generally insufficient on their own [101]. The challenge lies in the fact that label noise corrupts the target supervision signal itself, making it fundamentally more damaging than input perturbations.

Moreover, label noise can severely distort the optimization landscape, making it harder for gradient-based learning algorithms to converge toward generalizable minima. Even small fractions of incorrect labels can mislead the optimization process, particularly in overparameterized models with high capacity.

Given these vulnerabilities, handling label noise effectively has become a crucial focus in modern deep learning research. A wide range of specialized robust training strategies have been proposed—including noise-tolerant loss functions [17], label correction and reweighting mechanisms [58], and sample selection methods [20]—to counteract the adverse effects of noisy supervision. The understanding of how DNNs behave under label noise serves as a foundation for the methodological innovations discussed in subsequent chapters.

2.3 Classical Methods for Learning with Noisy Labels

Despite the recent dominance of deep neural networks (DNNs), the challenge of learning from noisy labels has long been studied in classical machine learning. Early approaches developed strategies to mitigate the impact of corrupted supervision, many of which laid the foundations for modern robust learning techniques.

Formally, we assume a supervised learning setup where we have access to a noisy training dataset:

$$D = \{(x_i, \tilde{y}_i)\}_{i=1}^{n}$$

where $x_i \in X$ represents the input feature, and $\tilde{y}_i \in Y$ is the observed (potentially corrupted) label associated with x_i.

A fundamental model for label noise assumes that clean labels y are stochastically flipped into noisy labels $\tilde{y}$ according to a noise transition matrix $T \in [0,1]^{C \times C}$:

$$T_{ij} = P(\tilde{y} = j \mid y = i)$$

where C is the number of classes. If T is known or can be estimated, it is possible to correct for label noise during training.

Noise-tolerant loss functions are among the earliest approaches developed to handle corrupted labels. Conventional losses such as cross-entropy tend to amplify the effect of incorrect labels because they assign large penalties to misclassified examples. To address this, alternative loss functions like Mean Absolute Error (MAE) were explored for their noise robustness properties [17]. Later innovations, including the generalized cross-entropy (GCE) loss [99] and the symmetric cross-entropy loss [79], aim to balance robustness and learning effectiveness.

Another major direction involves noise modeling and loss correction. By estimating the transition matrix T, loss correction techniques adjust the learning objective to account for the label noise. Specifically, if T is known, the corrected empirical risk is minimized as:

$$\hat{R}_{\text{corrected}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{C} T^{-1}_{\tilde{y}_i, j} \, \ell(f_\theta(x_i), j)$$

where f_θ is the model with parameters θ, and ℓ is the loss function. Forward and backward correction techniques based on this principle have been proposed to compensate for corrupted supervision [58].

In addition to loss correction, sample selection and instance reweighting techniques have been developed. These methods assign reliability weights $w_i \in [0,1]$ to each training instance based on heuristics or learned confidence estimates:

$$\hat{R}_{\text{weighted}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} w_i \, \ell(f_\theta(x_i), \tilde{y}_i)$$

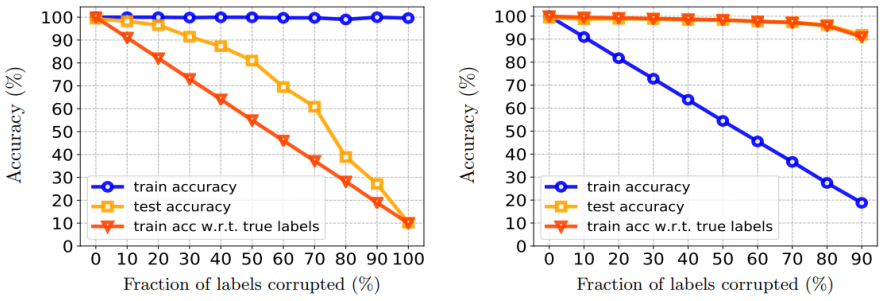
Instance weighting strategies aim to emphasize clean examples while suppressing the influence of noisy or uncertain samples [62, 28].

Classical noise mitigation approaches also include pre-processing methods that attempt to identify and remove noisy labels before training [7, 1], as well as data cleaning based on clustering, ensemble learning, or instance profiling.

To illustrate the types of label noise typically considered in this literature, we defer detailed examples and formal definitions to Chapter 3.2 (“Types of Label Noise”), where symmetric, asymmetric, and instance-dependent noise configurations are discussed in depth.

2.4 Early Stopping Strategies in Deep Learning

Deep neural networks (DNNs) are known for their powerful ability to fit complex data patterns. However, this expressiveness comes at the cost of a tendency to



(a) Trained model after many iterations (b) Trained model with early stopping

Figure 2.1. Impact of early stopping under noisy labels. Training and test accuracy curves for a 4-layer neural network trained on MNIST with varying amounts of random label corruption. (a) Without early stopping, the network memorizes noisy labels, harming test performance. (b) With early stopping, generalization is preserved despite high noise rates. (Adapted from Li et al., "Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks," 2019 [31]).

overfit, particularly in the presence of noisy labels. Early stopping has emerged as a simple yet highly effective regularization technique to counteract overfitting in deep learning [3, 94].

In classical supervised learning, early stopping involves monitoring performance on a separate validation set and halting training when the validation loss stops improving. This approach implicitly regularizes the model by preventing it from fully minimizing the training loss, thus favoring solutions that generalize better to unseen data.

When dealing with noisy labels, early stopping becomes even more crucial. Research shows that deep networks tend to fit clean examples first, memorizing the true patterns during the initial training epochs. As training progresses, the network eventually starts to memorize noisy labels, leading to degradation in generalization performance [3, 94]. By stopping training early—before memorization dominates—the model can capture meaningful structures in the data without overfitting to corrupted supervision.

2.4.1 Theoretical Foundations

Several theoretical studies [43, 31] have attempted to explain why early stopping is beneficial in the noisy label regime. These works suggest that:

- Early phases of training correspond to fitting the dominant, low-noise components of the data distribution.
- Later phases increasingly fit high-variance noise or adversarially mislabeled points.
- The inductive bias of stochastic gradient descent (SGD) favors learning simpler patterns first, providing a natural window for stopping before harmful memorization occurs.

Thus, early stopping can be seen as an implicit form of complexity control that protects models from overfitting to unreliable data.

2.4.2 Practical Strategies for Early Stopping under Label Noise

In practice, applying early stopping in noisy-label settings raises several challenges:

- **Validation Set Contamination**: If the validation set itself contains noisy labels, traditional early stopping criteria based on validation loss or accuracy become unreliable [22].
- **Alternative Signals**: Researchers have proposed using training dynamics (e.g., rate of loss decrease [3]) or trusted small clean datasets [22] to guide early stopping decisions more robustly.
- **Noise-Aware Stopping**: Some approaches dynamically estimate when the model transitions from fitting clean data to memorizing noise, using indicators such as sharp increases in loss variance or overfitting curves [59].

Overall, while early stopping is straightforward conceptually, its practical implementation in noisy environments requires careful design.

2.4.3 Mathematical View

We can express the idea of early stopping mathematically:

Let the model parameters at epoch t be θ_t. Suppose $L_{\text{val}}(\theta_t)$ is the validation loss at epoch t.

Classical early stopping selects:

$$t^* = \arg\min_t L_{\text{val}}(\theta_t)$$

In the presence of noisy labels, L_{val} may itself be noisy, thus requiring additional mechanisms or more robust criteria for choosing the optimal stopping point t^*.

A deeper theoretical analysis of early learning, memorization dynamics, and their implications for early stopping strategies is provided in Chapter 3.4.

2.5 Related Surveys and Comparative Studies

The challenge of learning from noisy labels has attracted significant attention, leading to several comprehensive surveys and comparative studies over recent years. These works provide valuable perspectives on the taxonomy, methodological trends, and empirical evaluation of noise-robust learning techniques.

One influential survey by Frenay and Verleysen [16] offers a classical perspective, focusing on early strategies for classification under label noise, including noise modeling, robust loss functions, and instance weighting. While foundational, this survey predates the deep learning revolution and thus does not cover modern neural architectures or large-scale learning under noisy supervision.

More recent surveys, such as the one by Han et al. [19], concentrate on label-noise representation learning in the deep learning era. They categorize methods

into four broad groups: noise transition modeling, loss correction, robust training strategies, and semi-supervised learning extensions. Their review provides a detailed categorization but primarily emphasizes the high-level algorithmic class rather than underlying architectural and training dynamics.

Song et al. [70] provide a highly influential and up-to-date survey, systematically reviewing 62 state-of-the-art robust training methods for deep neural networks under noisy supervision. They propose a five-category taxonomy (robust architectures, robust regularization, robust loss functions, loss adjustment, and sample selection) and compare methods based on six evaluation properties, including flexibility and noise robustness. Song et al.'s survey sets a solid foundation for understanding the current landscape of robust learning but focuses more on method grouping than on detailed feature-level analysis.

In contrast to these traditional taxonomies, this thesis adopts a feature-based taxonomy that dissects methods along fundamental architectural and algorithmic dimensions — such as network structure (e.g., single vs. multiple networks), regularization types (e.g., KL divergence regularization, embedding regularization), learning dynamics (e.g., reliance on early learning signals), and dependence on auxiliary data (e.g., semi-supervised losses). (As introduced in Chapter 1.3 and summarized in Table 4.1.)

This feature-oriented view enables a finer-grained comparative analysis, highlighting recurring design patterns and divergences across different approaches that may not be apparent under broader categorical groupings.

Several comparative experimental studies [58, 19, 2] have also highlighted the difficulty of evaluating methods consistently due to differences in noise simulation settings, dataset choices, and evaluation protocols. To address this, the thesis develops a modular benchmarking framework (described in Chapter 6) that systematically implements, configures, and evaluates selected state-of-the-art methods under consistent and reproducible experimental conditions.

In summary, while valuable foundations have been established by previous surveys, this thesis builds upon and extends them by offering:

- A new feature-based taxonomy tailored to architectural and dynamic aspects of robust training methods.
- A systematic empirical evaluation framework addressing inconsistencies found in prior comparative studies.
- A deeper analysis of early-stopping-related techniques as a central strategy for mitigating label noise.

Together, these contributions aim to advance both the understanding and the practical deployment of noise-robust learning systems in deep learning.

Chapter 3

Problem Definition and Methodological Foundations

3.1 Formalization of Learning under Label Noise

In the standard supervised learning setting, we assume access to a training dataset

$$\mathcal{D} = \{(x_i, y_i^*)\}_{i=1}^N,$$

where $x_i \in \mathbb{R}^d$ is a data point and $y_i^* \in \{1, 2, \ldots, K\}$ is its corresponding true label. However, in real-world scenarios, datasets often suffer from label noise. This means that the training data becomes

$$\tilde{\mathcal{D}} = \{(x_i, \tilde{y}_i)\}_{i=1}^N,$$

where $\tilde{y}_i$ may differ from the ground-truth label y_i^*. Learning under such noise requires care, as deep neural networks (DNNs) can easily memorize incorrect labels if training continues unchecked [3, 94].

Training is typically performed using mini-batch stochastic gradient descent. At training step t, a mini-batch

$$B_t = \{(x_j, \tilde{y}_j)\}_{j=1}^b$$

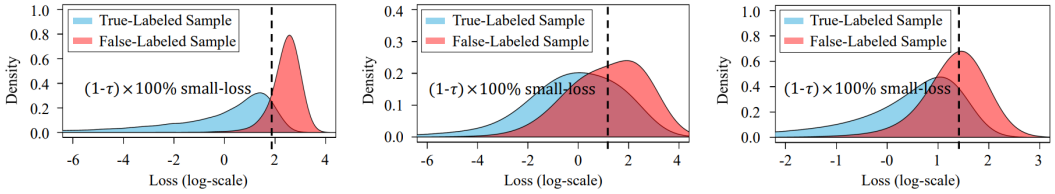
is sampled, and the model parameters θ_t are updated as:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \left(\frac{1}{|B_t|} \sum_{(x,\tilde{y}) \in B_t} \mathcal{L}(x, \tilde{y}; \theta_t) \right),$$

where α is the learning rate, and $\mathcal{L}$ is a loss function such as cross-entropy.

To study and control the behavior of DNNs in the presence of label noise, it is important to understand the notion of *memorization*. A sample is considered memorized if, across recent training epochs, the model repeatedly predicts the noisy label assigned to it. Let $\hat{y}_t = \Phi(x \mid \theta_t)$ be the model's prediction at epoch t, and define the prediction history over the most recent q epochs as

$$\mathcal{H}_x^q = \{\hat{y}_{t-q+1}, \ldots, \hat{y}_t\}.$$



(a) Symmetric Noise. (b) Pair Noise. (c) Real-World Noise.

Figure 3.1. Loss distributions at a training accuracy of 50%. (a) and (b) show results on CIFAR-100 with 40% synthetic noise: (a) uses symmetric noise, where each label is flipped to a random class with equal probability; (b) uses pair noise, where each label is flipped to a specific incorrect class. (c) shows results on FOOD-101N with real-world noise at a rate of 18.4%. Adapted from Song et al. [69].

The empirical prediction probability is given by [69]:

$$P(y \mid x, t; q) = \frac{1}{q} \sum_{j=1}^{q} [\hat{y}_{t-j+1} = y],$$

where the Iverson bracket $[\cdot]$ equals 1 if the condition is true, and 0 otherwise.

A sample x with noisy label $\tilde{y}$ is considered memorized if the most frequently predicted label in the last q epochs is equal to $\tilde{y}$:

$$\arg\max_{y} P(y \mid x, t; q) = \tilde{y}.$$

To evaluate memorization, *memorization precision* and *memorization recall* are proposed in [69]. Let $\mathcal{M}_t$ be the set of memorized samples at epoch t. Then:

$$\mathrm{MP}_t = \frac{|\{(x, \tilde{y}) \in \mathcal{M}_t : \tilde{y} = y^*\}|}{|\mathcal{M}_t|}, \quad \mathrm{MR}_t = \frac{|\{(x, \tilde{y}) \in \mathcal{M}_t : \tilde{y} = y^*\}|}{|\{(x, \tilde{y}) \in \tilde{\mathcal{D}} : \tilde{y} = y^*\}|}.$$

As illustrated in Figure 3.1, the point in training where memorization precision and recall intersect is considered optimal for early stopping. Beyond this point, precision drops rapidly as the model begins to memorize noisy labels—a phase described as the "error-prone period".

Identifying this early stopping point is challenging in practice, since ground-truth labels y^* are not available. To overcome this, [69] introduces two practical heuristics:

- **Validation Heuristic:** Stop training at the epoch with lowest error on a small clean validation set.
- **Noise-Rate Heuristic:** Stop when training accuracy reaches $(1 - \tau) \times 100\%$, where τ is the known noise rate.

These lightweight strategies approximate the optimal early stopping point without requiring full access to clean labels.

3.2 Types of Label Noise

Label noise refers to the phenomenon where the assigned class label $\tilde{y}$ does not correspond to the true underlying class y for a given training example x. In supervised

learning, especially with deep neural networks, such noise can significantly affect training dynamics, leading to overfitting and poor generalization. Understanding the types of label noise is crucial to developing robust learning algorithms and evaluating their effectiveness.

In the literature, label noise is typically categorized into three major types based on the underlying noise generation mechanism:

- **Symmetric Noise**: Every label has an equal chance of being flipped to any other class, regardless of the input features or semantics.
- **Asymmetric Noise**: Labels are flipped in a structured way, usually between semantically similar classes, simulating real-world confusion.
- **Instance-Dependent Noise**: The probability of label corruption depends on the input features themselves, making it the most challenging and realistic type.

This taxonomy provides a foundation for both theoretical analysis and empirical evaluation in robust learning. These noise types are widely used in benchmarks and have been formalized in multiple robust learning studies, including [16, 53, 99].

Each of these types is discussed in detail in the following subsections, with mathematical formulations and practical implications.

3.2.1 Symmetric Noise

Symmetric label noise (also known as uniform noise) refers to the scenario where each class label in the dataset is randomly flipped to any of the other classes with equal probability, independent of the input features or the class semantics. This type of noise assumes that all classes are equally likely to be confused with one another, and that the corruption process is unbiased and class-agnostic.

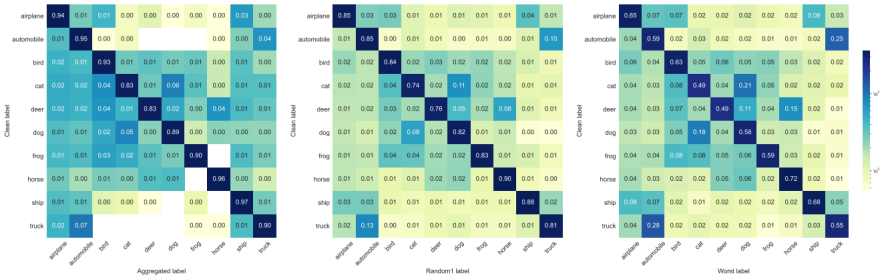
Formally, let C denote the number of classes. For a given true label $y \in \{1, \ldots, C\}$, the noisy label $\tilde{y}$ is generated according to:

$$P(\tilde{y} = j \mid y = i) = \begin{cases} 1 - \eta, & \text{if } j = i \\ \frac{\eta}{C-1}, & \text{if } j \neq i \end{cases}$$

Here, $\eta \in [0, 1]$ is the noise rate, representing the probability of a label being corrupted. When $\eta = 0$, the labels are completely clean. When $\eta = 1$, labels are maximally corrupted and all labels are uniformly random.

Symmetric noise is often used in experimental settings due to its simplicity and ease of simulation. It is considered a worst-case uniform corruption, as it provides no structure for the model to exploit. Despite being somewhat unrealistic in real-world applications, it serves as a useful benchmark for evaluating the robustness of algorithms under high uncertainty.

This noise model is commonly used in robust learning studies and benchmarks, including in [99, 53], and is implemented in many synthetic noisy-label datasets.



(a) Aggregated noisy labels (b) Random noisy labels (c) Worst noisy labels

Figure 3.2. Transition matrix of CIFAR-10N noisy labels (color bar is log-norm transformed). Adapted from Wei et al. [82].

3.2.2 Asymmetric Noise

Asymmetric label noise (also known as class-dependent noise) refers to structured label corruption where certain classes are more likely to be misclassified into specific, semantically similar classes. Unlike symmetric noise, where any label might flip uniformly to another, asymmetric noise reflects realistic annotation errors — such as confusing a "cat" with a "dog" or an "automobile" with a "truck" — which are more common in real-world human-labeled datasets.

Formally, let C be the number of classes. Given a true label $y \in \{1, \ldots, C\}$, the noisy label $\tilde{y}$ is generated as:

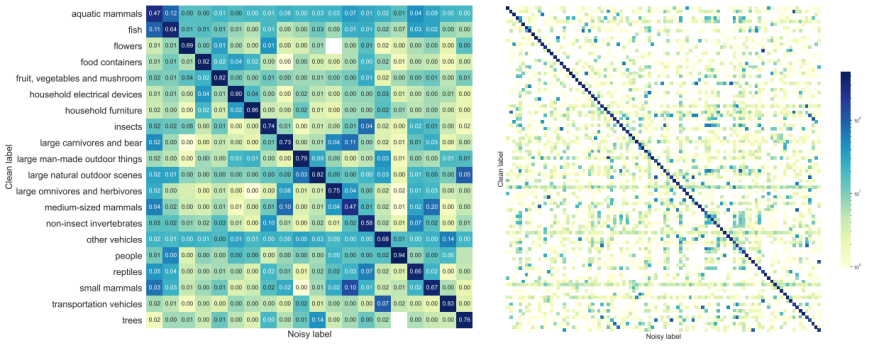
$$P(\tilde{y} = j \mid y = i) = \begin{cases} 1 - \eta, & \text{if } j = i \\ \eta_{ij}, & \text{if } j \neq i \text{ and } j \in N(i) \\ 0, & \text{otherwise} \end{cases}$$

Here, η_{ij} defines the probability of class i being flipped to class j, and $N(i) \subseteq \{1, \ldots, C\} \setminus \{i\}$ represents the confusion set — i.e., the classes most likely to be confused with class i. This noise structure is often derived from real-world data, such as annotation statistics or confusion matrices.

Asymmetric noise is more difficult to detect and correct than symmetric noise, because the corrupted labels are still semantically reasonable. Furthermore, the transition probabilities η_{ij} are not uniform and are often not observable, making modeling and learning under such noise particularly challenging.

As shown in Figure 3.2, the transition matrices from CIFAR-10N under various noise conditions (e.g., aggregated human labels, synthetic noise, worst-case labels) exhibit strong asymmetric patterns — such as frequent mislabeling between visually similar classes like "truck" and "automobile", or "bird" and "airplane". These patterns illustrate the non-uniformity and semantic bias of asymmetric noise.

This type of noise better reflects real-world annotation issues and is widely used in benchmarks such as CIFAR-10N [82], WebVision [35].



(a) Noisy coarse labels (b) Noisy fine labels

Figure 3.3. Transition matrix visualization for CIFAR-100N showing fine-grained label flip patterns. Each class tends to confuse with a small subset of others, reflecting structured, asymmetric label noise. Adapted from Wei et al. [82].

To further highlight this, Figure 3.3 presents the asymmetric noise patterns in CIFAR-100N, showing the high-frequency confusion between classes in the same superclass. This behavior reinforces the class-dependent nature of such label noise.

Asymmetric label noise is therefore a more accurate proxy for real-world noise and is widely used in benchmarking robust learning methods. Prominent datasets like CIFAR-10N [82], WebVision [35], and Clothing1M [87] all feature class-dependent noise, making this setting critical for evaluating the performance of noise-robust algorithms.

3.2.3 Instance-Dependent Noise

Instance-dependent label noise is the most general and realistic type of label corruption, where the likelihood of mislabeling a sample depends on the specific input features $\mathbf{x}$ rather than just the true class y. In this setting, two samples belonging to the same class may be labeled correctly or incorrectly with different probabilities, depending on their feature representations.

Unlike symmetric or asymmetric noise, which rely on fixed class-wise transition probabilities, instance-dependent noise reflects the variability in annotator behavior or the inherent ambiguity of samples. For instance, images that are blurry, occluded, or lie near class boundaries are more likely to be mislabeled. This complexity makes instance-dependent noise particularly challenging to handle, as it invalidates many assumptions underlying simpler noise models.

Formally, for a given input $\mathbf{x}$ with true label y, the noisy label $\tilde{y}$ is sampled from a distribution $P(\tilde{y} \mid \mathbf{x})$, which is no longer conditionally independent of the input. That is:

$$P(\tilde{y} \mid \mathbf{x}, y) \neq P(\tilde{y} \mid y)$$

Moreover, Figure 3.4 from the same study visually confirms this phenomenon. By dividing feature representations (extracted from a trained ResNet34) into clusters, the authors show that noisy label assignments vary significantly depending on which cluster an instance belongs to. Even within the same true class, some clusters

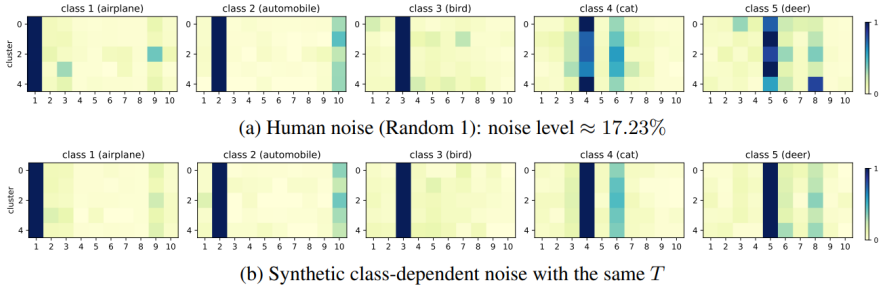


Figure 3.4. Visualization of noise transitions under human-level label noise vs. synthetic noise. Feature representations from ResNet34 are clustered using k-means; the class-conditional confusion varies between clusters, showing that noise depends on input features. (Adapted from Wei et al. [82]).

experience more severe or systematic mislabeling than others. This clear correlation between input features and corruption pattern validates the instance-dependent noise model.

Instance-dependent noise is especially difficult to handle using traditional correction techniques because it does not follow a global or static noise transition matrix. The dynamic, input-sensitive nature of this noise often demands advanced solutions, such as instance-level confidence modeling, feature-aware reweighting, or meta-learning-based sample selection.

Because of its complexity and its relevance to real-world applications, instance-dependent noise has become a key focus in the development and evaluation of noise-robust deep learning methods.

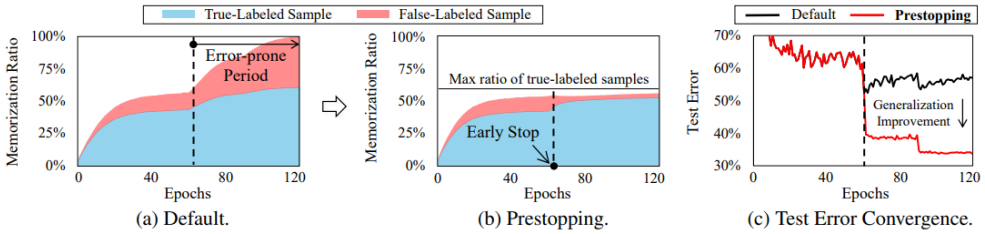
3.3 Learning Dynamics of Deep Neural Networks

Deep neural networks (DNNs) exhibit unique learning dynamics, particularly when trained on datasets containing label noise. One of the most widely observed behaviors is the *memorization effect*, where networks first learn clean and simple patterns before eventually fitting noisy or mislabeled examples [3, 94].

[69] provides a detailed investigation of this phenomenon. The study demonstrates that DNNs go through two distinct phases during training: an early phase, dominated by learning clean-label samples, and a later phase, characterized by increasing memorization of noisy labels. The transition between these phases has critical consequences for model generalization.

The rate at which a model memorizes noisy labels depends on the type of label noise. In the case of symmetric noise—where each label is randomly flipped to any other class—the separation between clean and noisy samples (in terms of loss or prediction behavior) is relatively large. In contrast, with pair noise, where labels are flipped to specific incorrect alternatives (e.g., “cat” to “dog”), the separation is smaller and memorization of noisy labels begins earlier. This makes learning dynamics more subtle and prone to error, as the model finds it harder to distinguish between clean and corrupted data [69].

To analyze these dynamics quantitatively, the memorization ratio is introduced,



(a) Default. (b) Prestopping. (c) Test Error Convergence.

Figure 3.5. Memorization dynamics under noisy labels. The memorization ratio of false-labeled samples rises sharply in the error-prone phase, while memorization of true-labeled samples plateaus. (Adapted from Song et al., [69])

which tracks how many clean versus noisy samples the model memorizes over time. Let $M_{\text{true}}(t)$ and $M_{\text{false}}(t)$ denote the number of memorized samples with correct and incorrect labels at epoch t, respectively. As training progresses, we typically observe:

$$\frac{d}{dt}M_{\text{false}}(t) \gg 0, \quad \frac{d}{dt}M_{\text{true}}(t) \approx 0 \text{ or } < 0.$$

This inequality reflects that after a certain point in training, the model stops learning useful information and starts memorizing label noise. This shift marks the entrance into what the study terms the *error-prone regime*—a period during which continuing training results in degraded generalization due to the absorption of mislabeled examples.

Figure 3.5 visualizes this behavior by plotting the memorization ratios of clean and noisy samples as training progresses. The plot shows that memorization of noisy samples accelerates in later epochs under standard training, while memorization of clean samples plateaus. This contrast clearly indicates the onset of overfitting.

To counteract this harmful phase, the study compares standard training with an early stopping strategy. Under early stopping, training is halted before the memorization of noisy labels intensifies. As demonstrated in Figure 3.5(c), models trained with early stopping achieve better generalization performance on the test set compared to those trained without interruption. The consistent performance across datasets and noise types validates early stopping as a simple yet effective means of mitigating the negative impact of noisy labels [69].

These insights into the temporal dynamics of memorization suggest that robust learning from noisy data requires more than just loss minimization—it demands time-aware control over the training process. Early stopping provides such control, by terminating training precisely when useful generalization reaches its peak and before memorization degrades it.

3.4 Early Learning versus Memorization: Theoretical Insights

The generalization ability of deep neural networks (DNNs) in the presence of label noise has motivated extensive theoretical analysis. While overparameterized models are capable of fitting even completely random labels [94], their optimization dynamics

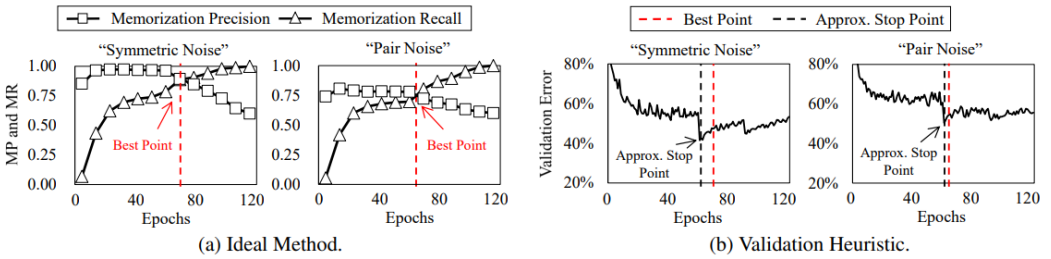


Figure 3.6. Memorization dynamics under noisy labels. The memorization ratio of falsely labeled samples rises sharply in the error-prone phase, while memorization of truly labeled samples plateaus. Adapted from Song et al. [69].

suggest that learning from clean data precedes the memorization of noisy examples. This temporal separation provides a theoretical foundation for early stopping as a regularization strategy.

[69] synthesizes both empirical and theoretical evidence to explain why early stopping improves generalization under noisy supervision. The core idea is that gradient-based optimization algorithms, such as stochastic gradient descent (SGD), tend to minimize the loss for easier, more consistent samples—typically those with correct labels—before fitting more difficult, often noisy examples. This results in an implicit prioritization of clean data early in training [3].

Let us denote the overall loss on a noisy dataset as the sum of losses over clean and noisy subsets:

$$L(\theta) = L_{\text{clean}}(\theta) + L_{\text{noise}}(\theta),$$

where θ represents the model parameters. In the early phases of training, the gradient of the clean part dominates the update:

$$\theta_{t+1} = \theta_t - \alpha \left(\nabla L_{\text{clean}}(\theta_t) + \nabla L_{\text{noise}}(\theta_t) \right), \quad \text{with } \|\nabla L_{\text{clean}}\| \gg \|\nabla L_{\text{noise}}\|.$$

However, as training continues, the contribution of noisy gradients increases, eventually corrupting the optimization path and leading the model to memorize incorrect labels.

Building on this insight, [69] introduces the concept of a *maximal safe set*—a collection of training samples confidently learned before memorization begins. These are identified using the memorization precision and recall metrics. The optimal stopping point is defined as the intersection of these two metrics, representing the best balance between generalization and noise avoidance. This point is visualized in Figure 3.6, which shows that memorization precision declines after recall saturates, indicating the onset of overfitting.

The theoretical basis for early stopping is further reinforced by results from Li et al. [34], who show that gradient descent with early stopping is *provably robust* to label noise in overparameterized networks. Their analysis demonstrates that if optimization is terminated before noisy gradients dominate, the resulting model converges to a solution that generalizes well to the clean data distribution.

Oymak et al. [55] contribute additional theoretical justification by analyzing the Jacobian structure of neural networks during training. They show that early in training, the learned representations tend to lie in a low-rank subspace, which is

more stable and generalizable. As training progresses, this low-rank structure breaks down—coinciding with the memorization of noise—further motivating the need to stop training early.

In a related direction, Hendrycks et al. [21] show that early stopping also improves robustness and uncertainty calibration, especially when combined with pretraining. Their findings imply that early-stopped models are less sensitive to label corruption and are better calibrated in terms of output confidence.

Taken together, these theoretical insights explain why early stopping is effective: it acts as an implicit form of regularization that leverages the natural order in which DNNs fit data. By halting training before memorization sets in, the network retains its generalization ability without requiring architectural changes or noise-aware loss functions.

While early stopping is theoretically well-justified as a form of implicit regularization, a range of practical methods have been developed to operationalize this principle in the presence of label noise. These approaches aim to determine the optimal stopping point by analyzing training dynamics, loss behavior, or prediction consistency over time. They offer concrete strategies for mitigating memorization and improving generalization without requiring access to clean labels. A detailed examination of these methods, their underlying mechanisms, and comparative effectiveness will be presented in Chapter 6.

Chapter 4

Feature-Based Taxonomy for Robust Training Methods

4.1 Motivation for a Feature-Oriented Taxonomy

Training deep neural networks on datasets with noisy labels has inspired a plethora of robust learning methods in recent years. Researchers have approached the label noise problem from varied angles—designing noise-tolerant loss functions, filtering out suspected mislabeled samples, adding correction layers, applying semi-supervised learning, and more [58, 20, 32, 48].

Traditional surveys and reviews typically organize these techniques into broad categories based on their primary strategy (e.g., robust loss functions, sample selection, label correction, etc.) [16, 19]. While such categorizations provide a high-level overview, they often fall short in capturing how modern state-of-the-art methods combine multiple strategies. In fact, many leading approaches today are hybrid methods that simultaneously incorporate several techniques [32].

For example, the method **DivideMix** blends small-loss sample selection with semi-supervised consistency training and data augmentation, rather than fitting neatly into a single category. This convergence of ideas means that a simple, flat list of methods or one-dimensional grouping can oversimplify important differences and commonalities among algorithms.

To address these gaps, we propose a **feature-based taxonomy** that classifies noisy-label learning methods along multiple dimensions of design, rather than assigning each method to one exclusive class. The key idea is to break down each algorithm into a set of fundamental features, enabling a multi-faceted comparison. This granular view is crucial because many robust training methods span more than one traditional category [19, 32].

By describing methods in terms of their features, we can systematically highlight which techniques they share and how they differ, without losing nuance. Prior comparative studies have started to recognize the need for such analysis. For instance, Song et al. [70] grouped methods by five general strategies and evaluated each group on criteria like required supervision and noise resilience. However, that approach still treated each method as belonging to one group, with any cross-category innovation labeled as a "hybrid" outlier.

Table 4.1. Feature-Based Taxonomy of Robust Learning Methods

Feature Name	Description
Architectural Structure	
Using One Network	Method uses a single deep neural network without duplication.
Using Two or Multiple Networks	Method employs two or more networks jointly (e.g., Co-teaching, DivideMix).
Regularization Mechanisms	
Class-Balanced Regularizer	Loss function is adjusted to correct class imbalance induced by label noise.
KL Divergence Regularizer	A KL divergence penalty enforces consistency in prediction distributions.
Embedding Regularization	Regularization is applied directly to feature representations.
Adding Additional Learnable Parameters	Extra trainable modules (e.g., noise adaptation layers) are added.
Learning Dynamics	
Loss Regularization Only	Loss function is modified without changing network architecture or data.
Add Contrastive Loss	Contrastive learning objectives are introduced to resist memorization.
Dependence on Auxiliary Information	
Needs Clean Validation Dataset	A small trusted clean set is required for tuning or early stopping.
Add Semi-Supervised Loss	Semi-supervised learning components are integrated.
Meta-Learning Framework	Meta-learning strategies are used for dynamic sample reweighting.

In contrast, our feature-level taxonomy treats the important aspects of an algorithm—architecture, loss/regulation techniques, training dynamics, and use of extra information—as first-class characteristics. This approach avoids pigeonholing complex methods into single bins and directly addresses the shortcoming of existing surveys: the lack of a unified framework to compare methods that integrate multiple components.

Ultimately, a feature-based taxonomy provides deeper insight into why certain methods excel and helps identify combinations of ideas that have been under-explored in the literature.

We identify four major dimensions that collectively characterize robust training methods under label noise. Each dimension corresponds to a question about the method's design, and within each dimension as summarized in Table 4.1.we enumerate specific features that methods may or may not possess.

This fine-grained, feature-oriented perspective enables several advantages:

- It supports more detailed comparisons across methods with similar high-level labels but different internal assumptions.
- It facilitates modular benchmarking, where one can analyze the impact of individual features (e.g., co-training vs. single-network) across various settings.
- It allows identifying combinable features that may yield more powerful hybrid strategies.

The taxonomy introduced here is not meant to be exhaustive but rather expressive enough to cover the key design axes of state-of-the-art techniques. It also serves as a practical framework for organizing the literature review in Chapter 6 and interpreting experimental.

In summary, this feature-based taxonomy offers a systematic framework to classify and compare noisy-label learning methods. By examining algorithms along the four dimensions of architecture, regularization, dynamics, and auxiliary requirements, we move beyond superficial groupings and toward a deeper insight into their inner workings.This structured perspective not only clarifies how current state-of-the-art techniques (like SOP, ELR, CORES, NES, etc.) relate to each other, but also helps identify open areas—combinations of features not yet fully explored—that could inspire future research.

In the following sections, we delve into each taxonomy dimension in detail, using it to analyze a wide range of robust training methods under label noise.

4.2 Architectural Structure

An important way to categorize robust learning methods under label noise is by the number of neural networks (models) they employ. Some approaches use a single network, relying on that model's internal mechanisms to handle noisy labels, whereas others use two or more networks in tandem, leveraging interactions between peer models to improve noise robustness. This design choice influences how the method mitigates label noise, affects training stability (e.g., preventing divergence or overfitting), and determines what interaction strategies are possible during training. We discuss these categories below, focusing on how the number of networks relates to robustness and the dynamics of training under noisy labels.

4.2.1 Single-Network Methods

Methods in this category train a single neural network on the noisy dataset, so all noise-handling must be done within one model's training process. Without any peers to cross-check its predictions, a single network approach typically relies on specialized training tricks to remain robust to incorrect labels. Common techniques include:

- **Robust Loss Functions and Regularization:** These methods replace standard loss functions with noise-tolerant alternatives (e.g., MAE, GCE) or apply explicit regularization (e.g., weight decay, dropout) to reduce sensitivity to outliers and prevent overfitting [61, 49].

- **Sample Weighting and Curriculum Learning:** The model prioritizes examples with lower losses (assumed to be clean), either by reweighting or filtering them. This self-paced learning process gradually includes harder examples, helping the model focus on reliable data during early training [20, 28].

- **Self-Correction and Label Refinement:** Without a peer model, the network may iteratively adjust training targets based on its own predictions. For example, bootstrapping techniques blend noisy labels with model-inferred pseudo-labels to guide training:

$$L_{\text{bootstrap}} = (1-\alpha)\,\ell(f_\theta(x), \tilde{y}) + \alpha\,\ell(f_\theta(x), y^*),$$

where $y^* = \arg\max_c p_\theta(c \mid x)$ is the predicted label and $\alpha \in [0, 1]$ controls reliance on the model's belief [61].

- **Early Stopping and Monitoring:** Training is halted before the network begins memorizing noise, typically using a clean validation set or heuristics. Some methods include a restart phase: after early stopping, a cleaner subset is selected and used for retraining, further improving robustness [69, 32, 81].

Single-model approaches benefit from their simplicity—only one model to train—and can be quite effective when combined with the above techniques. However, the lack of a peer means the model has no external check on its decisions. Thus, one challenge is avoiding the model falling into a trap of trusting its own erroneous predictions (confirmation bias) without correction. Methods mitigate this by internal heuristics (like using loss thresholds or statistical models of the loss) to guess which labels are likely wrong [28, 81]. In practice, single-network methods may perform well at moderate noise rates, especially with strong regularization, but they can struggle under extremely noisy conditions (e.g., when a large fraction of labels are random) because the model might eventually start fitting the noise. This limitation motivates the use of multiple networks, which we discuss next, to provide mutual error correction.

4.2.2 Two or Multiple Network Methods

In this category, the learning algorithm trains two (or more) networks simultaneously and exploits their interaction to achieve greater robustness to noisy labels. The core idea is that having a second opinion (or several) can help validate or filter the supervision signal: peer networks can detect each other's mistakes or agree on predictions, thereby reducing the chance that all models are misled by the same noisy label.

Intuitively, if each network is initialized differently and learns at its own pace, it will not memorize the noise in exactly the same way or at the same time as its partner. Thus, networks can catch each other's errors before those errors propagate. This approach significantly addresses the *confirmation bias* issue—the tendency of a solo model to reinforce its own errors. In fact, by training two networks to filter errors for each other, one can effectively avoid the confirmation bias that would plague self-training [49].

Empirically, multi-network methods have shown the ability to withstand very high noise rates (e.g., 50%+ corrupt labels) much better than single-model methods [20, 32].

Interaction Strategies Multi-network methods can be categorized by how the models interact during training. Several interaction strategies have been explored in the literature:

- **Mutual Data Selection (Peer Teaching):** In this strategy, each network selects a subset of training samples it considers reliable—typically those with low loss—and shares them with its peer for training. This mutual selection mechanism helps prevent a model from reinforcing its own potential errors, as

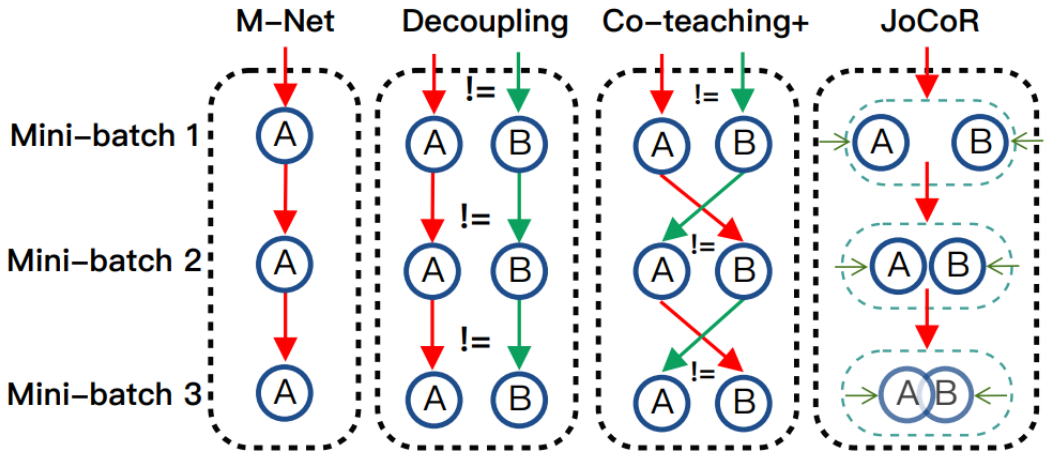


Figure 4.1. Comparison of error flow in MentorNet, Decoupling, Co-teaching+, and JoCoR. **Left:** MentorNet uses a single network (A), which may reinforce its own incorrect predictions. **Middle-left:** Decoupling updates two networks (A and B) only when their predictions disagree (!=), reducing mutual confirmation. **Middle-right:** Co-teaching+ selects small-loss samples with prediction disagreement and exchanges them between networks. **Right:** JoCoR jointly optimizes both networks with a co-regularized loss, aligning predictions to both ground-truth and peer outputs. (Adapted from Wei et al. [81].)

training data is filtered through the lens of the other model. As illustrated in Figure 4.1 (middle-right), some advanced variants of this approach (e.g., Co-teaching+) further refine the process by retaining only those low-loss examples on which the two networks disagree in their predictions, thereby adding an additional robustness filter. This crisscross exchange of trusted and informative samples between networks helps isolate noisy labels and encourages more stable and noise-resilient learning trajectories.

- **Peer Disagreement (Update by Disagreement):** Here, the networks are updated only on samples where their predictions differ. By skipping samples they agree on (which may include memorized noisy labels), the method emphasizes ambiguous or hard-to-learn examples. As shown in Figure 4.1 (middle-left), this decoupling strategy helps preserve diversity in the networks' learning paths and mitigates simultaneous memorization of noise, since updates occur only when uncertainty or conflict exists..

- **Output Coupling and Agreement Enforcement:** This strategy explicitly encourages prediction alignment between two networks by coupling their outputs through a joint loss function (e.g., KL divergence or mean squared error between predicted distributions). As shown in Figure 4.1 (right), both networks are optimized together on the same mini-batches, with a loss that enforces agreement between them in addition to fitting the ground-truth labels. This alignment helps the models converge to a stable consensus, ideally focusing on clean data. To avoid jointly memorizing incorrect labels, this strategy is often paired with sample selection mechanisms that prioritize likely correct

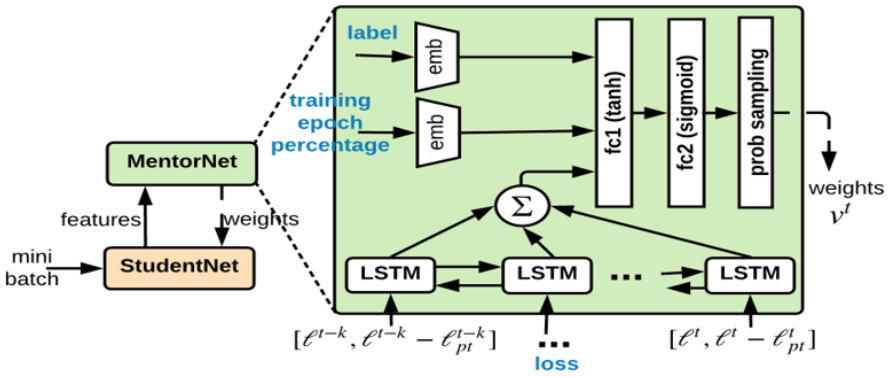


Figure 4.2. Teacher–student training framework where the mentor network guides sample weighting. The student network learns from the data, while the mentor (either pre-trained or co-trained) provides instance-level weights to prioritize clean or informative samples during training. (Adapted from Jiang et al. [28].)

examples.

- **Asymmetric Teacher–Student Roles:** One model (the teacher) is fixed or updated conservatively and supervises the training of a second model (the student) by providing soft labels, confidence weights, or sample-specific loss adjustments. The teacher may be a pre-trained model, a moving average of the student, or trained on a small clean subset. This asymmetry stabilizes training, as the teacher provides guidance unaffected by the noise currently influencing the student. The student, in turn, can focus on learning from examples that are more likely to be clean. As shown in Figure 4.2, this framework can be implemented via a separate MentorNet that dynamically learns a curriculum to weight the training samples for the StudentNet, encouraging the student to focus on trustworthy labels during training. This dynamic and data-driven approach is particularly effective in preventing the overfitting of deep models to corrupted labels.

- **Ensembles and Committees:** Extending beyond two networks, ensembles aggregate predictions from multiple models to dilute the influence of noisy labels. Consensus voting, majority filtering, or averaging predictions can identify unreliable examples. Though computationally expensive, ensemble strategies increase robustness by leveraging collective judgment across independently trained networks.

In summary, using additional networks provides multiple lenses on the noisy data, which greatly helps in identifying and mitigating label noise. Peer networks can agree on truly clean examples and flag or filter out the suspicious ones by either not selecting them or by indicating disagreement [69]. This generally leads to more stable training: one model's momentary lapse is unlikely to derail the entire learning process, since the other model(s) can compensate or at least not reinforce the error.Consequently, two-network methods have demonstrated significantly improved resilience against heavy noise, often achieving higher accuracy on clean test data

than any single model approach in identical settings [20, 32]. The trade-off, however, is computational. Maintaining two deep networks doubles the number of parameters and iterations, making the training roughly twice as slow and more memory intensive. There is also a risk that both networks could still converge to the same wrong solution if not designed properly.To counter this, strategies like the disagreement update or maintaining initial diversity are crucial to keep the models complementary. When successful, multi-network approaches exemplify the adage "two heads are better than one"—by interacting through co-training, peer review, or consensus-building, they achieve noise-robust learning outcomes that single networks would struggle to attain on their own.

4.3 Regularization Mechanisms

When training with noisy labels, adding appropriate regularization can mitigate overfitting to incorrect labels. Several types of regularization mechanisms have been explored to make learning robust against label noise. This section discusses four such strategies: (1) class-balanced regularization, (2) Kullback–Leibler (KL) divergence-based regularization, (3) embedding or manifold regularization, and (4) introducing additional learnable parameters to explicitly model noise. Each subsection outlines the concept, provides a mathematical formulation if applicable, and notes an example method (to be detailed in the next chapter). Figures from recent studies are suggested for illustration where appropriate.

4.3.1 Class-Balanced Regularizer

One issue in learning with noisy labels is that models may become biased toward classes with more (or less noisy) samples. A class-balanced regularizer addresses this by encouraging the model's predictions to match a desired class distribution (often the uniform or known prior distribution over classes). In practice, one can estimate the mean predicted probability for each class:

$$\bar{f}_c = \frac{1}{N} \sum_{i=1}^{N} f_c(x_i; \theta),$$

and penalize deviation from the prior p_c for class c. For example, a regularization term can be defined as a KL divergence between the prior and the average model prediction:

$$R_{\mathrm{CB}} = \sum_{c=1}^{C} p_c \log \frac{p_c}{\bar{f}_c},$$

which reaches its minimum (0) when $\bar{f}_c = p_c$ for all classes.

If the true class frequencies are equal (class-balanced data), $p_c = \frac{1}{C}$ yields an entropy-based penalty that prevents the network from concentrating predictions on a few classes. By adding such a term to the loss (with some weight λ), the training process is biased toward treating classes more evenly despite noise. This helps because under label noise, certain classes (especially minority classes) might otherwise be neglected or misclassified disproportionately.

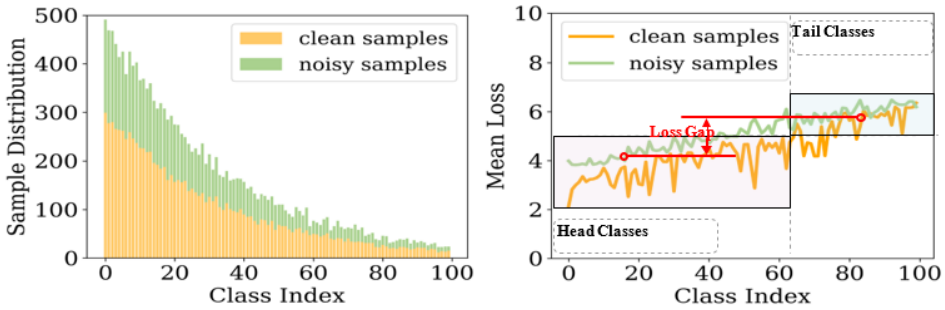


Figure 4.3. The sample distribution (left) and the mean loss variation (right) on noisy and imbalanced CIFAR-100 (noise rate is 0.4 and imbalance factor is 20). We can find: (1) both tail class samples and noisy samples exhibit large losses; (2) losses of some clean samples belonging to tail classes are even larger than losses of some noisy ones from head classes. Accordingly, existing low-loss-based sample selection methods tend to fail when distinguishing clean and noisy samples. This inspires us to develop a class-balanced sample selection method to combat noisy and imbalanced labels. (Adapted from Liu et al. [41])

A well-known use of class-balanced regularization in noisy label learning is the work of Tanaka et al. [74], who included a "class entropy" term to spread out predictions according to class priors when training on noisy datasets. In their joint optimization framework, this regularizer guided the model to not collapse into predicting only the majority classes. As a result, even if some classes had fewer clean samples, the model still allocated capacity to them due to the regularization.

Figure 4.3 in "Learning with Imbalanced Noisy Data by Preventing Bias in Sample Selection" [41] vividly illustrates the motivation: it shows that on a noisy, class-imbalanced dataset, the minority-class samples can exhibit higher loss than some noisy majority-class samples, confusing standard training. A class-balanced regularizer combats this by ensuring the model doesn't simply learn the easiest (or majority) classes and ignore the rest.

In summary, class-balanced regularization leverages prior knowledge of class distribution to make learning more robust to noise-related bias.

4.3.2 KL Divergence Regularizer

Another effective strategy is to introduce a KL divergence regularizer into the loss, measuring the discrepancy between two distributions and penalizing it. KL divergence can be used in different ways: one is to modify the loss function itself to be more noise-robust, and another is to enforce consistency between multiple predictions or models.

An example of the former is adding a reverse KL term to the standard cross-entropy loss. For instance, the symmetric cross-entropy approach combines the ordinary cross-entropy (which is

$$H(p, q) = -\sum_i p_i \log q_i$$

for true distribution p and model prediction q) with a "reverse" cross-entropy

term

$$H(q,p) = -\sum_i q_i \log p_i.$$

This effectively adds a KL divergence from the model's predicted distribution to the one-hot label distribution (or a soft label), which is symmetric to the conventional term. Intuitively, this punishes overconfidence on any single class and makes the model robust to label noise by not blindly trusting potentially corrupted labels. Empirically, such KL-based regularization (e.g., the Symmetric Cross Entropy loss) has been shown to prevent deep networks from overfitting noisy labels by balancing fitting and regularization [79].

KL regularization is also used to enforce agreement between models or between successive predictions. For example, a technique known as co-regularization trains two neural networks simultaneously on the noisy dataset and adds a term to the loss that penalizes the KL divergence between the two networks' output probability distributions. This encourages the two models to produce similar predictions for each input, effectively regularizing each other.

The method JoCoR (Joint Training with Co-Regularization) employs this idea: it minimizes a joint loss consisting of the sum of each network's classification loss plus a co-regularization term (the KL divergence between the networks' outputs). By doing so, both networks are steered toward consensus on data points – a property that helps reduce the chance of one network wildly memorizing a wrong label since disagreement would incur a penalty. Over time, the networks agree on more outputs, which tends to happen on clean data rather than noisy data [81].

Figure 4.1 illustrates this paradigm by comparing it with earlier approaches – it shows that JoCoR feeds the same mini-batch to two models and aligns their predictions (via a KL regularizer) before selecting small-loss (likely correct) samples for updating.

In addition to these, KL divergence has been integrated in label correction frameworks. For instance, PENCIL uses predicted label distributions for each example (instead of fixed one-hot labels) and includes a KL divergence term in its objective to gradually match the model's predictions to these adjustable label distributions. Initially the label distribution is uniform or based on the given noisy label; as training progresses, the KL regularizer encourages the network output to get closer to the current label estimate, and vice versa (using a reverse KL), i.e.,

$$KL(f(x_i;\theta)\|y_i^d),$$

where y_i^d is the learnable label distribution. This effectively performs a soft consistency check between what the model believes and the labels, providing a regularizing effect that corrects noise [91].

While the details of such methods will be covered later, the key point is that adding a KL divergence-based term helps constrain the learning process: it can either temper the influence of noisy labels on the loss or synchronize multiple predictions, both of which improve robustness to label noise.

4.3.3 Embedding Regularization

Embedding regularization (also referred to as manifold or representation regularization) aims to enforce smoothness or structural constraints on the model's learned representations or on an auxiliary latent space, so that the presence of noisy labels does not arbitrarily distort the learned features. The core idea is that if two instances are similar (in input space or feature space), the model should treat them similarly despite potential label errors.

One way to implement this is by constructing a similarity graph (or affinity matrix) among training examples and adding a penalty term that keeps the model's outputs or latent features consistent with that graph structure. For example, we can define W_{ij} to be 1 if example i and j are deemed neighbors (e.g., their inputs are close and they share the same estimated clean label), and 0 otherwise. An embedding regularizer can then be formulated as:

$$R_{\text{emb}} = \sum_{i,j} W_{ij} \, \|z_i - z_j\|^2,$$

where $z_i = f(x_i; \theta)$ is the feature embedding of instance i (or any representation of i, such as a predicted label distribution or noise-transition vector). This is essentially a graph Laplacian regularization: it heavily penalizes the model if it maps two similar inputs to very different representations.

By minimizing R_{emb}, the model is encouraged to learn a representation manifold where local neighborhoods of data (believed to share the same true class) stay smooth and clustered, even if some labels are wrong. Conversely, one can also push dissimilar points (e.g., different classes) apart using a related term, but the general principle is to align the geometry of the learned representation with the presumed true labeling structure.

A concrete example is the method by Cheng et al. [11], which estimates an instance-dependent noise transition matrix for each sample and uses manifold regularization to constrain these matrices. They assume that "the closer two instances are, the more similar their corresponding transition matrices should be." To enforce this, they build an intrinsic graph connecting nearest neighbors of the same (noisy) class and add a regularizer:

$$\sum_{i,j} S_{ij}^{(I)} \|T(x_i) - T(x_j)\|^2,$$

(plus a similar term for an extrinsic graph of different classes), where $T(x)$ is the estimated noise transition for instance x, and $S_{ij}^{(I)}$ encodes neighborhood relations. Minimizing this encourages the noise model $T(x)$ to vary smoothly on the data manifold, which makes the estimation more robust and less prone to overfitting noise idiosyncrasies.

More generally, methods that incorporate embedding regularization (e.g., using pairwise constraints, contrastive losses, or triplet-based penalties on features) help the network learn discriminative yet noise-tolerant features. They exploit the intuition that the true underlying structure of the data (how images or samples relate to each other) changes more gradually than random label noise. Figure 4.4 in Cheng et al. [11], could be referenced here, as it schematically shows a neural network whose

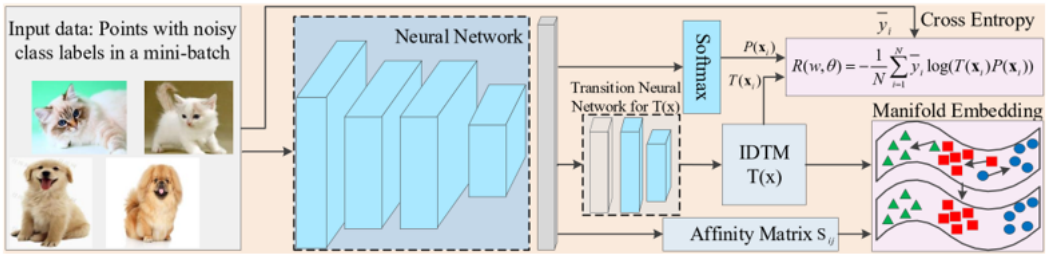


Figure 4.4. depicts a neural network whose noise estimates are regularized by an affinity graph (maintaining local similarity); and for additional noise parameterss. Adapted from Cheng et al. [11].

output (a per-instance noise transition matrix) is regularized by an affinity graph of the inputs, thereby aligning the learned noise parameters with the data manifold.

4.3.4 Adding Additional Learnable Parameters

A more explicit way to handle label noise is to introduce additional learnable parameters into the model architecture that are dedicated to modeling the noise. By increasing the model's capacity in a targeted way, the hope is that the extra parameters will absorb the label noise effects, allowing the main model to focus on learning the true underlying mapping.

One popular instantiation of this idea is the **noise adaptation layer**. In this approach, one augments the classification network with a final layer (often a fully connected layer with a softmax) that sits on top of the standard output layer. This noise layer is parameterized by a matrix $W \in \mathbb{R}^{C\times C}$ which represents a learned label transition matrix. It transforms the predicted clean-label probabilities $P(y|x;\theta)$ into noisy-label probabilities $P(\tilde{y}|x;\theta,W)$ via:

$$P(\tilde{y}=j \mid x) = \sum_{i=1}^{C} P(y=i \mid x;\theta)\cdot W_{i,j}$$

In other words, $W_{i,j}$ is the model's estimate of the probability that an example with true class i ends up with observed label j. These $W_{i,j}$ entries are treated as learnable parameters, updated along with the network weights θ during training. The effect is that the model can explicitly correct or account for systematic noise: the base network learns to predict the clean label distribution, and the noise layer maps it to fit the observed (noisy) labels. At test time, the noise layer can be removed or inverted if an estimate of true labels is needed.

This approach was introduced by Goldberger and Ben-Reuven [18], who optimized the network and noise-layer parameters via an EM-like procedure to maximize the likelihood of the noisy data. By adding this extra softmax layer, they achieved better generalization on noisy datasets, as the noise layer effectively modeled confusion between classes without corrupting the feature learning of the network.

Figure 4.5 (adapted from Song et al. [70]) depicts this idea: it shows a base DNN whose outputs are fed into an additional layer representing the label flipping probabilities, which, during training, learns to invert the noise process.

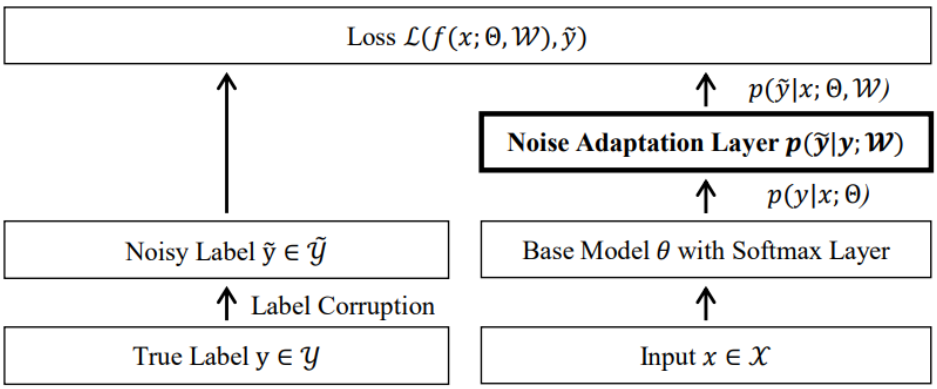


Figure 4.5. Illustration of a noise adaptation layer placed on top of a base network. It models the label corruption process by estimating a label transition matrix (adapted from Song et al. [70]).

Another way to add parameters for noise handling is to treat the labels or label confidences themselves as latent variables to be learned. In this paradigm, one introduces a set of auxiliary parameters $\{y_i^d\}$, one for each training sample, which represent the model's belief about the true label (often in the form of a probability distribution or continuous label embedding). These are sometimes called *soft labels* or *pseudo-label parameters*, and they are updated during training alongside the model weights.

For example, in the PENCIL method [91], each noisy example i is assigned a learnable label distribution $\{y_i^d\}$ (a vector of dimension C). The network is trained on these soft labels using a suited loss (including a KL divergence term), and the label parameters y_i^d themselves are gradually refined by gradient descent to better fit the model's predictions. This effectively adds $N \times C$ extra parameters (for N samples and C classes), significantly expanding the parameter space. However, those parameters are there to soak up noise – if a training label is incorrect, the model can adjust that sample's y_i^d away from the provided label, rather than having to fit it exactly.

By the end of training, the y_i^d may serve as corrected labels. This approach of label smoothing or label re-learning via additional parameters has proven robust, as it gives the model the flexibility to discount or change individual noisy labels in a principled way.

In summary, introducing dedicated parameters for noise is a form of regularization that acknowledges the noise explicitly in the model. Whether through a noise transition layer (which globally models how labels flip) or per-sample latent label variables (which individually adjust each sample's supervision), these methods prevent the base model from overfitting the noise. They act as a buffer or filter for the noise. The additional parameters are usually coupled with some constraints or regularizers (to keep them from degenerate solutions), but they significantly improve noise robustness when optimized correctly.

This idea will reappear in the next chapter's methods – many state-of-the-art noisy label algorithms either add small networks (teachers, mentors, or noise layers) or extra per-example parameters to manage noisy data.

4.4 Learning Dynamics

In addition to the static regularization mechanisms of Section 4.3, robust learning under label noise can exploit the learning dynamics of neural networks. This approach leverages how model behavior changes over time—for example, the empirical observation that deep networks tend to fit correct (clean) labels during early epochs before eventually memorizing noisy labels [3, 94].

By intervening in the training process based on time (e.g., early-phase vs. late-phase behavior), one can mitigate the impact of noisy data. In contrast to traditional regularization (which imposes fixed constraints like norm penalties or data augmentation throughout training), methods based on learning dynamics adapt when and how the model learns from each example.

Two key strategies in this vein are:

- **Time-aware loss functions** that capitalize on the early learning phase by penalizing divergence from initial predictions or stabilizing representations before memorization sets in (e.g., Early-Learning Regularization).
- **Contrastive or consistency-based objectives** that encourage agreement across time, models, or data views, making it harder for the model to latch onto noise without support from similar samples or prior beliefs.

We discuss each of these directions in the subsections that follow.

4.4.1 Loss Regularization Only

A well-established empirical observation is that deep neural networks tend to learn clean-label patterns first and only later begin to memorize noisy labels [3, 94]. This dynamic suggests that time-aware regularization strategies—which modulate learning based on the training stage—can improve robustness. Unlike static regularizers (e.g., weight decay), these methods adjust the loss or training process over time to align with the model's evolving behavior.

The most straightforward application of this idea is early stopping. By monitoring a small clean validation set or internal heuristic, training is halted at the point when generalization peaks—before the network begins memorizing incorrect labels. This simple strategy is surprisingly effective and has shown strong performance under various noise settings [69].

A more refined approach is Early-Learning Regularization (ELR) [43]. This method adds a temporal constraint to the loss function to discourage the model from deviating from its early predictions, which are likely influenced by clean labels. Let $p_i(t)$ denote the model's output distribution for sample i at epoch t, and let q_i be a stabilized estimate of the early prediction (e.g., an exponential moving average of p_i). The regularization term is defined as:

$$R_i = -\log(p_i(t) \cdot q_i)$$

The complete loss becomes:

$$L_i = L_{CE}(p_i(t), y_i) + \lambda R_i$$

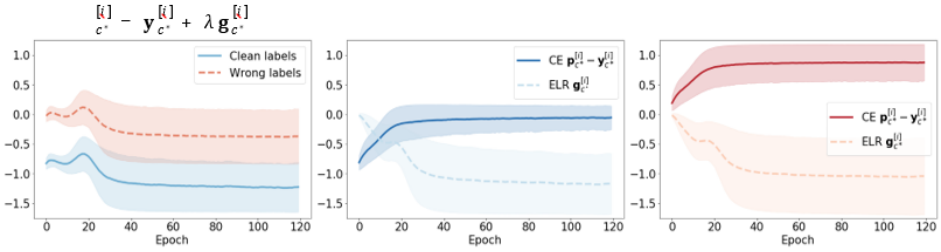


Figure 4.6. Under ELR, gradients for noisy samples decay to near zero, while gradients for clean samples remain strong—highlighting the regularizer's selective focus. (Adapted from Liu et al. [43])

where L_{CE} is the standard cross-entropy loss, and λ is a weighting coefficient. This formulation penalizes the model for shifting away from its early belief, thereby "locking in" likely-correct predictions and reducing the chance of overfitting noisy labels. ELR has been empirically validated to maintain higher gradients for clean samples while suppressing updates for noisy ones, effectively halting memorization after the early phase [43].

Another widely adopted technique is loss-based sample selection. Since clean samples usually incur smaller losses in the early epochs, methods based on curriculum learning dynamically filter or reweight training data. At each step, only a subset of samples with the lowest losses—assumed to be more trustworthy—is used for backpropagation. This principle underpins several robust learning frameworks. For example, the training set may initially include all samples, then gradually narrow to those consistently showing small loss values [20].

Importantly, this process is adaptive: clean samples with initially high loss can re-enter training once their loss decreases, while persistently high-loss (likely noisy) examples are filtered out. This allows for a self-correcting training loop that continually refines its notion of which data is reliable. Unlike fixed regularizers, this method evolves with the model's confidence and learning trajectory [32].

In summary, loss regularization based on learning dynamics offers an effective and interpretable approach to improving robustness. Whether through explicitly penalizing deviation from early predictions or dynamically selecting reliable samples, these strategies exploit the natural progression of learning to suppress overfitting to noise without requiring additional networks or external supervision.

4.4.2 Add Contrastive Loss

While early-loss regularization focuses on the temporal dynamics of model outputs, contrastive learning introduces a relational perspective—shaping the model's internal representations by comparing samples directly. This technique, which has gained prominence in self-supervised and semi-supervised learning, can be adapted to noisy label scenarios to enforce representation consistency and resist overfitting to incorrect supervision.

In contrastive learning, the goal is to pull together embeddings of similar instances (positive pairs) and push apart dissimilar ones (negative pairs). This is formalized using a contrastive loss such as the InfoNCE loss:

$$L_{\text{contrast}}(i,j) = -\log\left(\frac{\exp\left(\text{sim}(z_i, z_j)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k\neq i]} \exp\left(\text{sim}(z_i, z_k)/\tau\right)}\right)$$

Here, z_i and z_j are embeddings of a positive pair (e.g., two augmentations of the same input), $\text{sim}(\cdot)$ is cosine similarity, τ is a temperature hyperparameter, and the denominator includes all other examples as negatives. This loss encourages the model to learn feature representations that are stable and invariant, independent of noisy labels.

One way contrastive learning is applied is via unsupervised pretraining: models are first trained with a self-supervised contrastive loss (e.g., SimCLR) on unlabeled or label-ignored data. This encourages clustering of semantically similar samples and reduces spurious correlations, producing a feature space aligned with true class structure [9, 63]. When fine-tuned with noisy labels, these pretrained features help delay memorization and improve robustness, especially in high-noise regimes [89, 32].

Beyond initialization, contrastive losses can be integrated during supervised training. For instance, adding a consistency loss between two augmentations of the same input helps enforce stable predictions:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{contrast}}$$

This loss does not rely on labels—it penalizes disagreement between views, identifying unstable predictions as potentially noisy [90]. Samples showing inconsistent outputs can be down-weighted, while consistent ones are used for supervised updates. This idea is also extended to model agreement, where predictions from two networks (or two augmentations) are compared to estimate label reliability [90, 81].

As illustrated in Figure 4.7, contrastive learning introduces representation-level consistency that helps mitigate label noise. On the right, a self-supervised contrastive (SelfCon) objective aligns two augmented views of the same input (green and red triangles), forming tight and stable clusters in embedding space. On the left, supervised contrastive learning (SupCon) extends this idea by pulling together samples from the same class (e.g., blue squares for class 1, blue circles for class 2) while separating unrelated examples (red triangles). By maintaining representation consistency for similar inputs in this way, contrastive losses act as a strong regularizer. The model is encouraged to map samples to stable clusters defined by inherent similarity rather than fitting them arbitrarily to noisy labels. As a result, predictions for each cluster become more reliable and less prone to fluctuate or follow noise.

Some methods use supervised contrastive learning selectively: samples identified as clean via the small-loss criterion or inter-model agreement are pulled together based on their labels, while uncertain samples are either excluded or handled via unsupervised contrastive loss [97, 90]. This hybrid (semi-supervised) approach ensures structure preservation for trustworthy data while still learning from unlabeled or noisy examples.

In summary, contrastive losses guide the network to learn stable, similarity-based representations rather than memorizing noisy labels. This enhances robustness and generalization, especially when combined with the early-learning dynamics. Together, these strategies offer a principled framework for learning from noisy datasets.

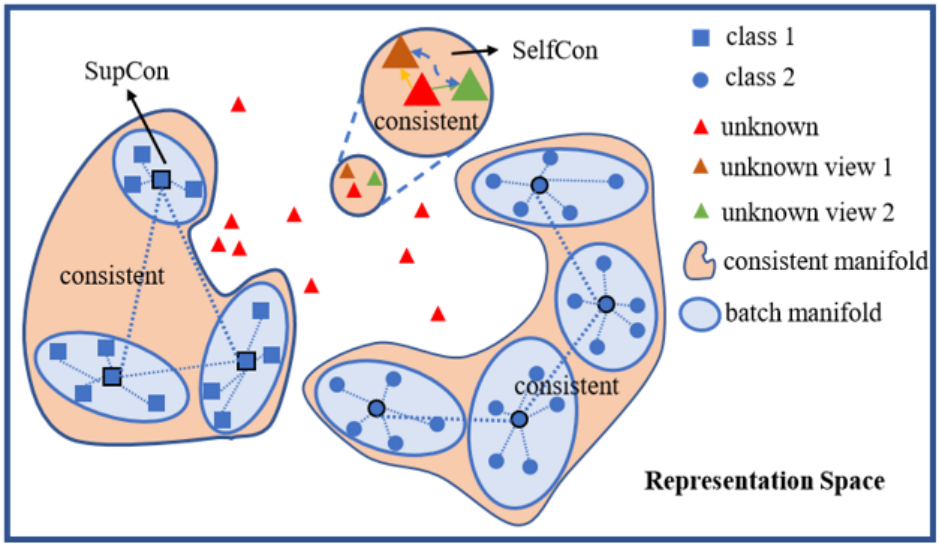


Figure 4.7. An example on how contrastive learning introduces extra consistency regularization. Models maintain consistent predictions for representations in orange manifolds. Both SelfCon and SupCon encourage representation space to maintain consistency in the batch. Given data batches drawn from the dataset, SelfCon only learns a small consistent manifold around each sample based on 'self-supervision'. SupCon can learn a more continuous manifold as samples with the same label can be clustered together. (Adapted from Zhang et al. [98])

4.5 Dependence on Auxiliary Information

Robust learning under label noise often benefits from additional auxiliary information beyond the raw noisy training set. Such information can guide the learning process to mitigate the impact of incorrect labels. This section discusses three common forms of auxiliary information and the role each plays in noisy label training.

4.5.1 Needs Clean Validation Dataset

One popular form of auxiliary data is a small clean validation dataset disjoint from the noisy training set. This is a set of examples with reliable labels (assumed free of noise) that can be used to monitor and guide the training. Its role is crucial in preventing overfitting to corrupted labels. Studies have shown that over-parameterized neural networks can memorize even completely random labels given enough training time, leading to poor generalization. By providing an unbiased evaluation signal, a clean validation set enables strategies like early stopping—halting training before the model starts fitting the noise [69].

In fact, stopping training once validation performance deteriorates can "avoid overfitting to the noisy labels." The idea is to exploit the network's tendency to learn simple patterns first and delay memorizing outliers. With a clean validation set, one can detect when the model transitions from learning generalizable patterns to fitting mislabeled data, and intervene by stopping or adjusting training at that point.

Beyond early stopping, a clean validation set can be used explicitly in the training

objective as a form of meta-feedback. Because this set represents the ground-truth distribution, it can guide the model to focus on genuine signal. For example, one can frame a bilevel optimization where the model's parameters are primarily learned from the noisy training data, but an upper-level objective evaluates performance on the clean set [62, 66]. In practice, this often means dynamically re-weighting or selecting training examples based on their contribution to validation loss.

A meta-learning algorithm by Ren et al. [62] implements this by assigning each training sample a weight that is adjusted via a meta-gradient to minimize loss on the clean validation set. Intuitively, samples that hurt validation accuracy (likely those with incorrect labels) get down-weighted, whereas those that help validation performance receive higher weights.

Another way to leverage a clean subset is through a teacher-student or curriculum learning framework. For instance, a "Mentor" network can be trained on the trusted data to learn a curriculum (i.e., a strategy for weighting or ordering training instances) and then guide the main network (the student) during training on noisy data. The MentorNet method [28] is an example: it uses a small set of true-labeled images to learn a data-driven curriculum so that the student focuses on samples likely to have correct labels.

This approach effectively filters or reduces the influence of suspected noisy examples using the mentor's guidance derived from clean data. Empirically, incorporating a clean validation set in such ways leads to significantly more robust models.

The clean data feedback allows the model to attain zero training error (fitting even the noisy labels) without a spike in validation error, indicating that overfitting to wrong labels has been avoided by virtue of the auxiliary clean set signal.

4.5.2 Add Semi-Supervised Loss

Another form of auxiliary information is unlabeled data (or using the noisy data as if unlabeled when appropriate). Techniques from semi-supervised learning are brought in to help learn from the input distribution even when labels are unreliable. In semi-supervised learning (SSL), the training objective includes an extra loss term on unlabeled examples to regularize the model [32]. This loss does not require ground-truth labels; instead it may encourage consistency (the model should give similar predictions for an input under different perturbations) or entropy minimization (the model should make confident predictions on unlabeled data) [66, 28, 62].

Incorporating such an SSL loss in the noisy label scenario can significantly improve robustness. The basic idea is to leverage the information in the feature distribution of all examples, including those that might have incorrect labels, by not trusting the given labels fully and instead learning from the structure of the data itself. Formally, let D_L be a set of training samples for which we use the given labels (presumed mostly correct), and D_U be the set of samples for which we ignore the labels and treat them as unlabeled. A typical training objective can be written as a combination of supervised and unsupervised terms:

$$\mathcal{L}_{\text{total}}(\theta) = \frac{1}{|D_L|} \sum_{(x_i,y_i)\in D_L} \ell(f_\theta(x_i), y_i) + \lambda \cdot \frac{1}{|D_U|} \sum_{x_j \in D_U} \ell_{\text{unsup}}(f_\theta(x_j))$$

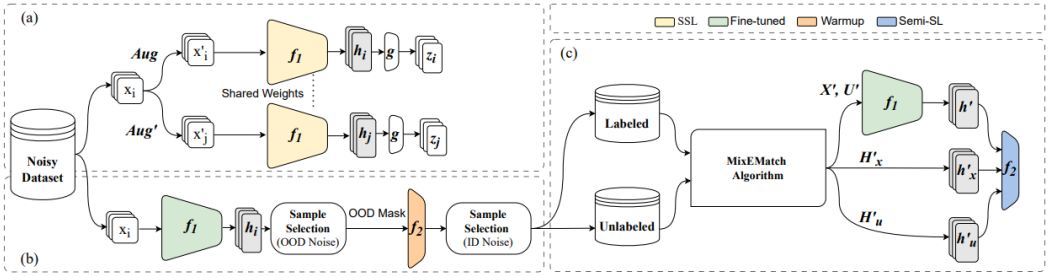


Figure 4.8. Semi-supervised training under noisy labels. After initial representation learning and filtering, the model separates the dataset into labeled (likely clean) and unlabeled (likely noisy) subsets by analyzing classification loss distributions. A semi-supervised algorithm is then applied, combining supervised loss on the clean samples with unsupervised consistency loss on the unlabeled samples. This structure-aware learning enables the model to utilize all data while reducing overfitting to incorrect labels. (Adapted from the DivideMix framework [32])

Here ℓ is the usual supervised loss (e.g. cross-entropy) on an example (x_i, y_i), and ℓ_{unsup} is a semi-supervised loss term that does not require a true label for x_j [66]. For instance, ℓ_{unsup} could measure consistency between predictions of f_θ on two random augmentations of x_j, penalizing differences to enforce prediction stability. It could also include an entropy term to avoid overly confident arbitrary predictions on unlabeled inputs [28].

By tuning the trade-off parameter λ, the model is guided to learn not just from potentially noisy labels, but also from the intrinsic structure of the input data. In the context of label noise, one can obtain D_L and D_U by partitioning the original noisy dataset based on some criterion of label reliability. A common approach is to use the network's behavior to estimate which data points are likely clean. For example, small training losses often indicate that a sample fits the model's current hypothesis well and is thus likely correctly labeled [69]. Approaches like DivideMix leverage this by modeling the loss distribution with a mixture model and separating the dataset into a "probably clean" subset and a "noisy" subset at each epoch [32]. The clean subset D_L is used with the standard supervised loss, while the noisy subset is treated as D_U and a semi-supervised loss is applied [32].

During training, the model then optimizes $\mathcal{L}_{total}$ as above, effectively recycling the noisy examples as unlabeled data rather than discarding them. This procedure regularizes the network: it can still learn from features of samples with incorrect labels (through the unsupervised term) without being misled by their wrong labels. The unsupervised consistency loss encourages the model to learn a smooth, clustered representation of data, which helps it resist fitting random label noise. By adding an SSL component, robust learning algorithms can substantially improve performance under high noise rates. They avoid relying solely on potentially corrupted supervision and instead extract signal from the input distribution. Many state-of-the-art noisy-label training methods employ this strategy. For instance, DivideMix demonstrated that using a semi-supervised objective in conjunction with noise separation yields excellent results on CIFAR-10/100 with synthetic noise and real-world noisy datasets [77, 32].

Figure 4.8 in the DivideMix paper provides an overview of this approach, showing how the data is split and fed into a semi-supervised training scheme. In summary, the use of an auxiliary unsupervised loss helps the model maintain correct predictions on clean test data by not overfitting to wrong labels – the model learns from data points even when their labels are suspect, but in a way that emphasizes consistency and cluster structure rather than the noisy label itself.

4.5.3 Meta-Learning Framework

A third category of methods relies on a meta-learning framework to handle noisy labels, which inherently uses auxiliary information (often a clean validation set, as in Section 4.5.1, or additional parameters) in a more sophisticated training paradigm. In meta-learning approaches, one treats certain aspects of the training process as learnable and optimizes them using a meta-objective. In the context of label noise, this usually means there is an inner training loop that learns the model on the noisy training data, and an outer loop that adjusts some meta-parameters by evaluating the model on a small clean dataset. The auxiliary clean set plays the role of a meta-trainer: it provides a feedback signal to learn how the model should handle noisy data. In effect, the method learns to learn from noisy labels. Most meta-learning solutions for noisy labels are formulated as a bilevel optimization problem [77]. Let θ denote the model's parameters (to be learned on noisy data) and let ϕ represent meta-parameters that govern the training (these could be per-sample weights, a noise correction mapping, or even the network initialization). We have two levels of objectives:

Inner level – train the model on the noisy training set D_{train} (with perhaps weighted or corrected labels defined by ϕ). This yields model parameters $\theta(\phi)$ after one or several training steps. For example, if ϕ assigns weights to samples, $\theta(\phi)$ would be the result of minimizing the weighted training loss [66, 62]:

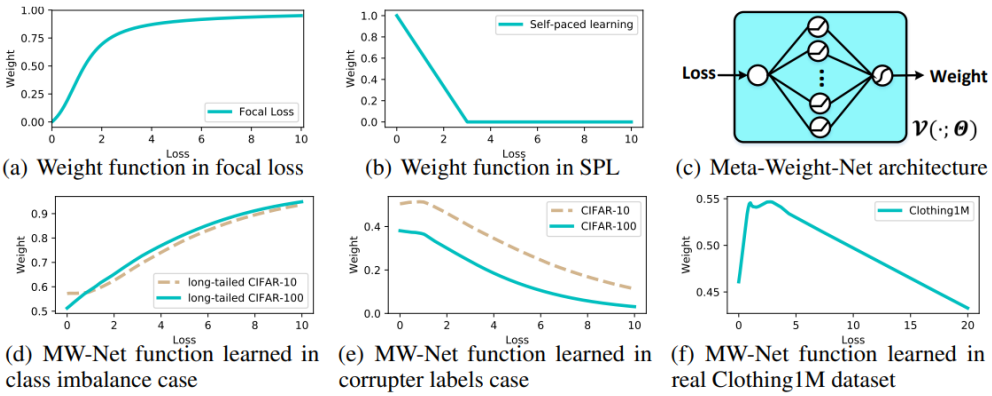
$$\min_{\theta} \sum_{(x_i,\tilde{y}_i)\in D_{\text{train}}} \phi_i \cdot \ell(f_\theta(x_i), \tilde{y}_i)$$

Outer level – evaluate the model on the clean validation set D_{val} and update ϕ to minimize the validation loss. In formula form, ϕ is optimized as:

$$\min_{\phi} \frac{1}{|D_{\text{val}}|} \sum_{(x_j,y_j)\in D_{\text{val}}} \ell(f_{\theta(\phi)}(x_j), y_j)$$

Here $D_{\text{val}} = \{(x_i^{\text{meta}}, y_i^{\text{meta}})\}_{i=1}^{M}$ is the small unbiased meta-dataset ($M \ll N$) used for meta-learning [66]. The meta-parameters ϕ thus receive gradients from the validation set, telling us how we should have trained on the noisy set.

By iterating between these two levels (or by differentiating through the inner optimization in a single step), the algorithm learns how to train on noisy data. In practical terms, the meta-learning framework automatically adjusts how much trust to place on each training example or how to adapt its label. This auxiliary mechanism can take various forms. One form is meta sample weighting: the method learns a function $w_\phi(\cdot)$ that maps a training sample's attributes (e.g., its loss) to



(a) Weight function in focal loss (b) Weight function in SPL (c) Meta-Weight-Net architecture

(d) MW-Net function learned in class imbalance case (e) MW-Net function learned in corrupter labels case (f) MW-Net function learned in real Clothing1M dataset

Figure 4.9. Task-adaptive weighting functions learned by Meta-Weight-Net under different training scenarios (adapted from [66], Figure 1). (d) In the class imbalance setting, the model assigns higher weights to high-loss samples, recognizing that they may belong to underrepresented classes and are important for generalization. (e) Under 40% uniform label noise, the model learns a sharply decreasing function, assigning near-zero weight to high-loss (likely noisy) samples and high weight to small-loss (likely clean) ones. (f) On the real-world noisy Clothing1M dataset, the learned function is flatter but still decreasing, indicating softer reweighting in the presence of naturally complex and ambiguous noise.

a weight, and ϕ (the parameters of this function) are updated so that a weighted training step yields better performance on D_{val} [62, 66].

Another form is meta label correction, where the approach treats the true label of each example as a latent variable/hyper-parameter to be optimized. The model's predicted "corrected" labels (or an auxiliary label-cleaning network) are refined by the outer loop such that the model trained on these corrected labels performs well on the validation set [77].

In both cases, the small clean set is the crucial source of truth that drives the meta-optimization. It provides the "signal" for what the network should output, thereby indirectly telling the model which training examples or labels to trust. The role of the auxiliary clean data here is similar to Section 4.5.1 but integrated more tightly into training: it explicitly shapes the learning algorithm. This meta-learning paradigm has yielded state-of-the-art results in learning with noisy labels [77]. By taking noisy labels as variables to tune (instead of fixed erroneous inputs), the method can markedly reduce their negative impact. For example, a meta-learning based reweighting method was shown to automatically learn to give lower weights to high-loss samples (which likely have incorrect labels) and higher weights to low-loss samples, emulating the effect of small-loss selection but in a data-driven way [66].

In Figure 4.9 of the Meta-Weight-Net paper, for instance, the learned weighting function under 40% label noise is indeed a decreasing function of the loss – assigning near-zero weight to examples with large losses (presumably mislabeled) and higher weight to those with small loss [66]. Such approaches essentially learn a robust training strategy: the auxiliary meta-knowledge (the clean validation set and possibly assumptions of a clean label distribution [66]) is used to adjust the training process on the fly. Over time, the model focuses on patterns that generalize (because that's

what improves validation loss) and ignores or corrects patterns that don't. As a result, the final classifier is much less influenced by the noisy labels it was trained on. The meta-learning framework thus leverages auxiliary information to learn to overcome label noise, rather than relying on a fixed heuristic. This flexible, learnable approach has proven effective in numerous recent works [77] and underscores the importance of even a small amount of clean data when dealing with large-scale noise.

4.6 Summary of Feature-Based Taxonomy

In this chapter, we proposed a feature-based taxonomy to systematically categorize robust training methods under label noise. Unlike prior taxonomies that primarily group methods by superficial structure or chronology, our framework decomposes methods along four orthogonal dimensions, each capturing a fundamental design decision in learning from noisy data. Each of these dimensions comprises concrete features such as "uses contrastive loss," "requires clean validation data," or "adds learnable label parameters," enabling a granular comparison across state-of-the-art techniques.

This taxonomy is not merely descriptive—it provides an analytical lens through which we can assess both common patterns and critical trade-offs in robust training strategies. It also prepares the foundation for Chapter6, where we apply this taxonomy to a curated set of representative methods. Each method will be mapped across the feature space, allowing us to contrast their design philosophies, strengths, and limitations in a principled way.

Chapter 5

Empirical Cue Analysis in Noisy Label Learning

5.1 Introduction: Why Empirical Cues Matter

Beyond architectural choices and regularization strategies, many recent methods for learning with noisy labels rely on observable patterns that emerge during training, often referred to as *empirical cues*. These cues are not encoded directly into the model architecture, but rather emerge from the characteristic behaviors of deep neural networks (DNNs) when exposed to noisy data. Understanding and leveraging these patterns can significantly enhance robustness to label noise.

Empirical cues capture behavioral signals—such as the tendency of DNNs to memorize clean samples before noisy ones (small-loss criterion), or the stability of predictions under augmentation—that provide practical heuristics for identifying and mitigating label noise. These cues are listed and described in Table 5.1, which summarizes the key types of training-time signals used across robust learning methods.

Table 5.1. Summary of Empirical Cues in Noisy-Label Learning

Empirical Cue	**Short Description**
Small-Loss Criterion	Prioritizes low-loss samples assumed to be clean, based on the memorization dynamics of DNNs.
Curriculum via Dynamic Thresholding	Gradually includes harder samples using adaptive thresholds, mirroring the memorization order of learning.
Temporal Consistency of Predictions	Encourages or tracks prediction stability over time, penalizing sudden changes in class probability.
Noisy Validation Early-Stopping	Stops training when noisy validation accuracy peaks, avoiding overfitting to corrupted labels.
Normalized Loss Metrics	Identifies noisy samples by comparing their loss to class-wise or global average loss.
Layer-Wise Memorization Rates	Leverages the differing learning rates of network layers—early layers generalize, deeper ones overfit.
Model Confidence on Labels	Uses prediction confidence as a proxy for label correctness.
Soft Label Refinement	Replaces noisy hard labels with soft targets derived from prediction distributions or latent structure.
Inter-Network Agreement	Uses agreement between two networks (or subnetworks) to reinforce reliable predictions.
Frequency-Domain Cues	Detects noise by analyzing robust phase vs. noise-prone amplitude components in feature representations.
Sparsity of Noise (Label Errors as Outliers)	Models label errors as rare outliers and applies sparse regularization techniques to isolate them.
Gradient Coherence	Exploits the fact that gradients on clean examples are coherent, while those on noisy ones are divergent.
Augmentation for Robustness	Applies input transformations (e.g., Mixup, cropping) to dilute the effect of noise.
Ensembling for Robustness	Uses prediction averaging (e.g., SWA, temporal ensembles) to reduce overfitting and improve stability.
Consistency under Augmentation	Measures how consistent predictions are across augmented versions of the same input.

These cues represent a distinct analytical dimension that complements, but

remains orthogonal to, the feature-based taxonomy introduced in Chapter 4. While the taxonomy categorizes design decisions (e.g., use of regularizers, multiple networks, or clean validation sets), empirical cues instead highlight how training-time dynamics and signals are used to infer sample reliability.

In this chapter, we formally introduce and define the most commonly used empirical cues in noisy-label learning. For each cue, we provide a concise explanation, its underlying assumption, and illustrative examples from state-of-the-art methods. This behavioral analysis provides a richer and more precise lens through which to interpret and compare the practical effectiveness of robust learning methods. It also sets the stage for the comparative synthesis presented later in Chapter 6, where both structural features and empirical behavior are jointly considered.

In the next sections, we provide a focused explanation of each empirical cue listed in Table 5.1 to clarify their role, underlying assumptions, and how they are leveraged in state-of-the-art noisy-label learning methods.

5.2 Explanation of Core Empirical Cues

5.2.1 Small-Loss Criterion

One of the most widely observed empirical behaviors in deep learning with noisy labels is the *memorization effect*—the tendency of deep neural networks to fit cleanly labeled data before fitting noisy labels. This observation underlies the *small-loss criterion*, a heuristic that assumes samples with lower training loss are more likely to be correctly labeled. By identifying and prioritizing these small-loss samples, various robust training methods attempt to reduce the influence of label noise during learning.

The small-loss phenomenon was first systematically investigated by Zhang et al. [94], who showed that overparameterized networks can perfectly fit random labels but do so only after fitting the clean patterns. Building on this, many methods explicitly exploit the early phase of training to identify clean examples. For instance, Co-teaching selects instances with smaller loss values in each mini-batch to exchange between networks during training [20], while DivideMix uses Gaussian Mixture Models to model the loss distribution and identify likely clean versus noisy samples [32].

The intuition is simple yet powerful: since clean samples are easier to fit, they yield smaller losses early in training, and models can use this loss signal to guide sample selection or weighting. The effectiveness of this criterion diminishes as training progresses and networks begin to memorize noisy labels. Therefore, the small-loss heuristic is often applied only during the early or mid-stages of training.

In some methods, such as PGDF (Prediction-Guided Divide and Filter) [36] and ProMix [86], the small-loss assumption forms the basis for building an initial clean subset. This subset is then used either for semi-supervised learning or to regularize the rest of the training process. Overall, the small-loss criterion has become a foundational empirical cue for robust training strategies under label noise.

5.2.2 Curriculum via Dynamic Thresholding

Core Idea Curriculum via dynamic thresholding is a training strategy that adapts the learning process based on the difficulty of individual training samples, mimicking a human-like curriculum where easier examples are learned first. The method starts by focusing on clean, easy-to-learn examples—typically those with low loss or high confidence—and gradually allows harder samples to influence training as the model matures. Importantly, the "difficulty threshold" for accepting a sample is not fixed but evolves dynamically during training based on the model's confidence and past performance.

How It Helps Against Noise This approach builds on the empirical insight that deep neural networks (DNNs) tend to learn clean patterns before memorizing noisy ones—a property known as the memorization effect. By prioritizing early-learned samples and only admitting harder ones later when the model is better calibrated, dynamic thresholding avoids learning from mislabeled or ambiguous data too early, which could otherwise destabilize training or propagate incorrect gradients.

Instead of relying on a global loss-based filter (as in the basic small-loss criterion), curriculum strategies use per-sample dynamics to determine whether a sample should be trusted at a given time, providing a more flexible and personalized pacing mechanism.

Implementation Example In CORES^2 [12], each sample is assigned a dynamically updated confidence score that tracks its "cleanliness" over rounds of training. During each round, only samples that pass a certain dynamic threshold—computed using a combination of prediction stability and regularized loss—are used to update the model.

Formally, if $L_{\text{reg}}(x_i)$ is the regularized loss for sample x_i, the threshold $\tau_t(x_i)$ at epoch t is adapted as:

$$\text{Include } x_i \text{ if } L_{\text{reg}}(x_i) \leq \tau_t(x_i)$$

where $\tau_t(x_i)$ increases gradually to reintroduce harder examples over time.

Practical Impact Dynamic thresholding allows for self-paced learning, where the curriculum is implicitly learned by the model from the data itself, rather than being hand-designed. It is particularly effective under instance-dependent label noise, where some samples are consistently difficult or mislabeled. Unlike fixed heuristics, dynamic thresholding adapts based on each sample's trajectory and the model's evolving trust in it.

Connection to Empirical Cues While closely related to the *Small-Loss Criterion*, curriculum learning introduces a time-evolving, individualized thresholding mechanism rather than a static loss cutoff. This makes it a more nuanced and potentially more robust alternative, especially when noise patterns are complex or correlated with sample features.

5.2.3 Temporal Consistency of Predictions

Core Idea Temporal consistency of predictions refers to the empirical observation that cleanly labeled samples tend to exhibit stable model predictions across training epochs. In contrast, noisy examples often induce fluctuating or inconsistent outputs over time. This cue assumes that prediction trajectories for clean data evolve smoothly, while noise introduces instability.

How It Helps Against Noise By tracking the prediction history of individual samples, several robust learning methods identify label noise indirectly. Samples that frequently change their predicted class or exhibit high prediction entropy across epochs are considered less trustworthy. Conversely, examples with consistent predictions are treated as more reliable and can be weighted higher or used to guide learning. This approach allows models to mitigate memorization of noisy labels without requiring clean supervision.

For instance, ELR and ELR+ integrate exponential moving averages of past predictions as a form of soft target regularization. This mechanism penalizes significant deviations from prior predictions, thereby enforcing temporal stability during training. Such strategies implicitly assume that early, stable outputs are more trustworthy indicators of the true label [43].

Formulation Example Let $p_t(x) \in \mathbb{R}^C$ denote the predicted class probabilities for sample x at epoch t. Temporal consistency can be enforced by minimizing the divergence between $p_t(x)$ and a moving average of past predictions $\bar{p}(x)$:

$$\mathcal{L}_{\text{consistency}} = \text{KL}(p_t(x) \,\|\, \bar{p}(x))$$

where the moving average is defined recursively as:

$$\bar{p}(x) \leftarrow (1 - \beta)\bar{p}(x) + \beta p_t(x),$$

with $\beta \in (0, 1)$ controlling the update rate. This penalizes abrupt shifts in prediction, especially for clean examples.

Impact in Practice Temporal consistency is a lightweight yet powerful signal used in ELR [43], ELR+, and other methods relying on stable supervision. It enables implicit filtering of noisy samples and helps maintain generalization by discouraging overfitting to label corruption.

5.2.4 Noisy Validation Early-Stopping

Noisy Validation Early-Stopping is an empirical strategy for halting training at an optimal point without requiring a clean validation set. This cue is grounded in the observation that even when validation data is noisy—i.e., drawn from the same noisy distribution as the training set—it can still reveal a critical signal: the point at which the model begins to memorize label noise.

Core Idea. In standard supervised learning, early stopping is performed using a clean validation set to detect the point at which generalization performance peaks. However, in the presence of label noise, such clean validation sets are often unavailable. Surprisingly, studies show that performance on a noisy validation set follows a similar trajectory: it improves early in training, peaks, and then declines as the model begins to overfit the noisy labels.

Thus, the noisy validation accuracy curve can be used to stop training at the peak, avoiding the memorization phase and preserving generalization.

Mathematical Description. Let $\mathcal{D}_{\text{val}}^{\text{noisy}}$ be a noisy validation set and f_θ the model at epoch t. We monitor the noisy validation accuracy:

$$\text{Acc}_{\text{val}}(t) = \frac{1}{|\mathcal{D}_{\text{val}}^{\text{noisy}}|} \sum_{(x_i,\tilde{y}_i)\in\mathcal{D}_{\text{val}}^{\text{noisy}}} \mathbb{1}[\arg\max f_\theta(x_i) = \tilde{y}_i]$$

Training is stopped at epoch t^*, where:

$$t^* = \arg\max_t \text{Acc}_{\text{val}}(t)$$

This point approximately corresponds to the onset of overfitting to label noise.

Application in Robust Learning. This strategy is used by Noisy Early Stopping (NES) [76], which demonstrates that using noisy validation performance as a stopping signal yields surprisingly strong generalization, nearly matching methods that rely on clean validation data. NES operates under minimal assumptions and can be applied as a plug-in mechanism to most training pipelines.

Unlike methods that require careful modeling of the noise distribution or rely on clean reference samples, NES simply tracks model performance on held-out but noisy data and uses it to prevent memorization.

Noisy Validation Early-Stopping provides a practical and effective method to prevent overfitting in noisy-label settings. It turns the model's accuracy curve—on data drawn from the same corrupted distribution—into a reliable early-stopping signal, eliminating the need for clean supervision or external noise estimation.

5.2.5 Normalized Loss Metrics

Normalized Loss Metrics are empirical cues designed to enhance the detection of noisy samples by contextualizing a sample's loss relative to the broader distribution of training data. Instead of using the absolute loss, these metrics consider how atypical a sample's loss is—globally or within its predicted class—providing a more robust and adaptive signal for identifying label noise.

Core Idea. Raw loss values alone are often insufficient to flag mislabeled samples because losses vary across classes and model confidence levels. However, noisy samples tend to exhibit losses that are consistently higher than those of their clean counterparts, especially as training progresses. To make this distinction more meaningful, loss normalization computes a relative loss score by dividing each sample's loss by a reference loss value:

- **Global-Normalized Loss (GNL)**: Measures each sample's loss relative to the average loss over all training samples.
- **Class-Normalized Loss (CNL)**: Measures each sample's loss relative to the average loss among samples of the same predicted class.

This normalized signal enhances outlier detection and helps methods isolate likely noisy samples.

Mathematical Formulation. Let $\mathcal{D} = \{(x_i, \tilde{y}_i)\}_{i=1}^{N}$ be the noisy training set, and $\ell_i = \mathcal{L}(f_\theta(x_i), \tilde{y}_i)$ the training loss for sample i. Then:

Global-Normalized Loss (GNL):

$$\mathrm{GNL}_i = \frac{\ell_i}{\frac{1}{N}\sum_{j=1}^{N} \ell_j}$$

Class-Normalized Loss (CNL):

$$\mathrm{CNL}_i = \frac{\ell_i}{\frac{1}{|C_k|}\sum_{j \in C_k} \ell_j}$$

where $C_k = \{j \mid \arg\max f_\theta(x_j) = k\}$ is the set of samples predicted to belong to class k, and $k = \arg\max f_\theta(x_i)$ is the predicted class for sample i.

These normalized losses can then be used to assign weights (e.g., lower weights to high-normalized-loss samples), to rank or filter samples, or as input features for meta-learning modules that predict label correctness.

Normalized Loss Metrics offer a statistically principled way to flag suspicious training instances. By measuring how atypical a sample's loss is—either globally or within its predicted class—these cues provide valuable empirical information for robust learning, especially in meta-learning frameworks and sample weighting strategies [39]

5.2.6 Layer-Wise Memorization Rates

The *Layer-Wise Memorization Rates* cue arises from the empirical observation that different layers of deep neural networks exhibit distinct sensitivities to label noise over the course of training. Specifically, lower (earlier) layers tend to learn generalizable low-level features (e.g., edges, textures in images), while higher (deeper) layers are more prone to memorizing label noise due to their higher capacity and proximity to the loss signal [94].

Motivation and Empirical Findings. Studies such as PES (Progressive Early Stopping) [45] and PADDLES [47] have demonstrated that later layers in a deep network begin to overfit noisy labels earlier than the shallow layers. This asymmetry can be leveraged to design training strategies that protect the robust feature extraction layers by halting or delaying training in more noise-sensitive layers.

Empirical findings suggest that gradients propagated to deeper layers are often more noise-corrupted. Let a neural network be decomposed into L layers, and let

$\theta^{(l)}$ denote the parameters at layer l. The update rule for each layer can be generally written as:

$$\theta^{(l)} \leftarrow \theta^{(l)} - \eta \cdot \nabla_{\theta^{(l)}} \mathcal{L}(x, y; \theta)$$

As training progresses, the gradient $\nabla_{\theta^{(l)}} \mathcal{L}$ for deeper layers (larger l) tends to reflect label-specific signals more directly and thus becomes more corrupted by noise.

Implementation in Robust Methods. To combat this, robust training strategies employ techniques such as:

- **Staged Training:** Initially train shallow layers and freeze them early, adding deeper layers gradually.
- **Progressive Freezing or Early Stopping:** Allow shallow layers to train longer, but stop training deeper layers once they begin overfitting.

PES [45], for instance, defines different stopping schedules for different layers. Shallow layers are updated throughout the training cycle, while deeper layers are introduced progressively and trained for fewer epochs. This staged regime mitigates overfitting by reducing the noise exposure of the final classification layers.

Similarly, PADDLES [47] proposes a frequency-based disentanglement of phase and amplitude components in the feature space, implicitly separating robust and noise-prone signals. Amplitude learning is stopped earlier—a concept analogous to freezing deeper layers based on their memorization tendency.

This cue illustrates that noise robustness is not uniform across the network depth. Leveraging this asymmetry—through selective freezing or layer-specific learning rates—enables methods to maximize generalizable knowledge extraction while minimizing noise fitting.

5.2.7 Model Confidence on Labels

One of the most widely used empirical cues in noisy-label learning is the model's confidence in its predicted label. This cue leverages the intuition that during training, cleanly labeled samples typically elicit high-confidence predictions from the model, whereas mislabeled or ambiguous samples tend to receive lower confidence scores or incorrect high-confidence predictions. As such, the confidence level of a model's prediction can serve as a practical proxy for assessing label reliability.

Confidence-Based Filtering and Reweighting. Many methods apply confidence thresholds to either filter or down-weight samples with uncertain predictions. For example, if the model predicts a class c for input x with softmax output $p_c = \max_k p(y = k \mid x)$, then:

- A sample $(x, \tilde{y})$ is considered clean if $p_c \geq \tau$, for some threshold $\tau \in [0, 1]$.
- It may be discarded or reweighted if $p_c < \tau$.

This strategy is commonly implemented as part of sample selection, pseudo-labeling, or loss attenuation mechanisms in robust training pipelines.

Confidence Regularization. Other methods directly incorporate confidence regularization into the loss function to encourage the model to produce confident predictions only when warranted. For example, CORES^2 introduces a term to penalize low-entropy softmax outputs unless the model is sufficiently confident, thereby creating sharper, more trustworthy predictions [12].

In general, confidence-based cues operate under the assumption that clean labels correlate with high-certainty predictions, while noise induces uncertainty. However, it is important to note that this assumption can break down in cases of systematic or adversarial noise, where the model may become confidently wrong.

5.2.8 Soft Label Refinement

Core Idea Soft label refinement refers to the practice of replacing potentially corrupted hard labels (i.e., one-hot vectors) with refined soft labels—probability distributions over classes that reflect uncertainty or learned structure. This approach assumes that training with soft, corrected targets reduces overfitting to noise and allows the model to generalize better in the presence of mislabeled data.

How It Helps Against Noise In noisy-label scenarios, hard labels can be incorrect and overly rigid. Soft labels, in contrast, convey partial class information and allow for more flexible learning. These labels may be derived from model predictions (e.g., moving averages over time), feature-based clustering (e.g., centroid similarity in latent space), or jointly estimated latent label distributions.

By supervising the model with soft targets rather than relying solely on given labels, training is regularized and noisy supervision is smoothed out. This reduces the harmful impact of corrupted labels and helps preserve useful patterns in the data.

Formulation Example Let $\tilde{y}_i \in \{0,1\}^C$ be a noisy hard label for sample x_i, and let $\hat{y}_i \in [0,1]^C$ be a soft label (e.g., model-predicted, centroid-refined, or smoothed distribution). The training loss can be formulated as:

$$\mathcal{L}_{\text{soft}} = -\sum_{c=1}^{C} \hat{y}_{i,c} \log p_\theta(x_i)_c$$

where $p_\theta(x_i)_c$ is the predicted probability for class c. This replaces the traditional cross-entropy with a soft-label version that encourages alignment with a refined distribution.

Impact in Practice Soft label refinement is widely used in robust learning approaches. For example:

- **ELR and ELR+** use temporally averaged predictions as soft supervision [43].
- **NCOD+** refines labels by measuring similarity in feature space and assigning pseudo-probabilities based on class centroids [80].

These methods show that learning from distributions rather than discrete labels can mitigate the impact of noise, especially when guided by empirical cues or feature structure.

5.2.9 Inter-Network Agreement

Core Idea Inter-network agreement refers to a training heuristic used in multi-network architectures, where the predictions or behaviors of two (or more) networks are compared and used to guide each other. The key assumption is that disagreement between networks signals uncertainty or noise, while agreement indicates reliable supervision. This cue is especially useful in methods where networks are trained jointly but independently, such as co-teaching or dual-network regularization frameworks.

How It Helps Against Noise When multiple networks are exposed to the same noisy data, their independent learning trajectories offer a redundancy that can be exploited to mitigate noise. If two networks agree on a prediction for a sample—despite possible label noise—it is likely to be correct. Conversely, samples with high disagreement can be down-weighted, ignored, or targeted for correction.

This idea is operationalized through:

- **Cross-network consistency loss**, e.g., penalizing divergent predictions.
- **Peer loss strategies**, where networks select training samples for each other based on agreement confidence.
- **Implicit alignment**, as seen in ELR+, where two networks regularize each other's early predictions over time.

Formulation Example Let $p_\theta(x)$ and $p_{\theta'}(x)$ be the predicted class distributions from two networks. A simple agreement regularization can be expressed using Kullback–Leibler divergence:

$$\mathcal{L}_{\text{agreement}} = \text{KL}(p_\theta(x) \parallel p_{\theta'}(x)) + \text{KL}(p_{\theta'}(x) \parallel p_\theta(x))$$

Alternatively, a symmetric form using mean squared error can be applied:

$$\mathcal{L}_{\text{agreement}} = \|p_\theta(x) - p_{\theta'}(x)\|^2$$

Minimizing this term encourages the networks to produce consistent outputs.

Impact in Practice Inter-network agreement is a core mechanism in methods such as:

- **Co-teaching and Co-teaching+** [20], where networks teach each other using small-loss samples selected independently.
- **ELR+** [43], where two networks regularize each other using exponential moving averages of past predictions, fostering mutual stability without sample selection.

This cue strengthens robustness by leveraging model diversity and shared knowledge, reducing reliance on noisy supervision alone.

5.2.10 Frequency-Domain Cues

The *Frequency-Domain Cues* stem from recent insights into how deep neural networks learn and represent signals of different frequency components over time. This cue specifically focuses on the idea that certain frequency components of the data—particularly low-frequency amplitude information—are more susceptible to overfitting label noise, while high-frequency phase components tend to remain more robust and structurally informative throughout training [88].

Motivation and Core Assumption. The frequency perspective is motivated by the decomposition of signals into amplitude and phase spectra, a concept well-established in signal processing. In the context of deep learning, representations learned by convolutional neural networks (CNNs) can also be viewed through this lens. Given a feature map $F \in \mathbb{R}^{H \times W}$, its frequency representation via the 2D Fourier transform is:

$$\hat{F}(u, v) = \mathcal{F}\{F(x, y)\}$$

where the transformed signal $\hat{F}(u, v)$ can be decomposed as:

$$\text{Amplitude: } A(u, v) = |\hat{F}(u, v)|, \qquad \text{Phase: } \phi(u, v) = \angle \hat{F}(u, v)$$

Empirical findings suggest that amplitude encodes global intensity variations, which can overfit to noisy labels, while phase captures local, structural details that are more resilient to noise [47].

Implementation in Robust Methods. The method **PADDLES** (Phase-Amplitude Disentangled Early Stopping) [47] is a representative example of utilizing this cue. It proposes to disentangle the learning of amplitude and phase components in intermediate feature maps and introduces differential stopping strategies:

- **Amplitude learning is stopped early**, preventing the model from encoding unstable low-frequency signals that often align with noisy labels.
- **Phase learning continues**, allowing the model to retain robust spatial details critical for clean label prediction.

This selective training is governed by a dynamic schedule that monitors when the amplitude spectrum begins to diverge—indicating overfitting to noise—and then halts further updates to amplitude-specific parameters.

The overall loss function is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{phase}} + \mathbf{1}_{t<T_a} \cdot \mathcal{L}_{\text{amplitude}}$$

where T_a is the epoch after which amplitude learning is stopped.

Frequency-domain cues allow robust training methods to filter out noisy label effects by focusing on the phase spectrum of learned representations. This perspective introduces a novel axis of regularization—orthogonal to time- or loss-based heuristics—by directly controlling the spectral dynamics of feature learning.

5.2.11 Sparsity of Noise (Label Errors as Outliers)

The *Sparsity of Noise* cue is based on a common assumption in noisy-label learning: label errors are sparse in the dataset and behave as statistical outliers in both input and feature space. That is, while most training samples are correctly labeled and form coherent clusters, mislabeled samples appear as exceptions—either in their feature embeddings, loss values, or prediction dynamics. This insight leads to strategies that explicitly model label noise as a sparse variable and separate it from the clean label learning process.

Core Idea and Mathematical Formulation. To formalize this cue, consider a training set of inputs $\{(x_i, \tilde{y}_i)\}_{i=1}^N$, where $\tilde{y}_i$ may be a corrupted label. Some methods model the true label distribution as:

$$y_i = \tilde{y}_i + \eta_i$$

where η_i is a noise correction term—ideally zero for clean samples and non-zero for corrupted ones. Under the sparsity assumption, only a small number of η_i should be active. This leads to the following optimization objective with a sparsity-inducing regularizer (e.g., ℓ_1 norm):

$$\min_{\theta,\eta} \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), \tilde{y}_i + \eta_i) + \lambda \|\eta\|_1$$

Here, f_θ is the model, $\mathcal{L}$ is the loss function (e.g., cross-entropy), and λ controls the trade-off between fitting the data and enforcing sparsity. This formulation ensures that label error is attributed to a small number of highly deviating examples, rather than being diffusely modeled across the dataset.

The sparsity-of-noise cue assumes that only a limited subset of labels are incorrect, and those noisy samples stand out in loss magnitude or embedding space. By explicitly modeling noise with sparse constraints, methods such as SOP (Sparse Over-parameterization) are able to isolate and suppress the impact of label noise without requiring clean supervision or external guidance [33].

5.2.12 Gradient Coherence

Core Idea Gradient coherence refers to the empirical observation that gradients computed from clean-labeled samples tend to align more consistently with each other, whereas gradients from noisy-labeled samples tend to be more random and uncorrelated. This phenomenon is leveraged in certain robust learning methods to distinguish between clean and noisy samples based on the geometric properties of their gradient vectors during training.

How It Helps Against Noise Under clean supervision, similar examples typically induce gradients that point in compatible directions in parameter space, leading to coherent optimization steps. In contrast, noisy labels produce gradients that often oppose or diverge from those of clean samples, causing optimization instability. By measuring and promoting gradient coherence—i.e., how well a sample's gradient

aligns with those of other samples—the model can selectively trust and amplify clean signals while suppressing noisy ones.

This strategy transforms gradient statistics into a noise-resistance signal, operating independently of prediction confidence or loss magnitude.

Formulation Example Let $\nabla\mathcal{L}_i$ and $\nabla\mathcal{L}_j$ be the gradients of the loss with respect to parameters θ for samples i and j. The cosine similarity between gradients measures coherence:

$$\text{Coherence}(i,j) = \frac{\langle \nabla\mathcal{L}_i, \nabla\mathcal{L}_j \rangle}{\|\nabla\mathcal{L}_i\| \cdot \|\nabla\mathcal{L}_j\|}$$

Averaging this across a batch gives an estimate of how consistently a sample aligns with the learning direction of the majority.

In **CORES**^{2}, this insight is integrated via a peer loss term that penalizes divergence from clean gradients by subtracting the expected loss over randomly sampled peers, thus enhancing signal alignment:

$$\mathcal{L}_{\text{peer}} = \mathcal{L}_i - \mathbb{E}_{j\sim B}[\mathcal{L}_j]$$

Impact in Practice Gradient coherence is most notably exploited in:

- **CORES**^{2}: The peer loss framework implicitly enforces gradient alignment by penalizing label inconsistency with a broader peer group.
- Other regularization strategies may approximate coherence through variance reduction or by aggregating gradients over multiple data views.

This cue provides a model-internal, training-dynamics-based signal that complements prediction-level heuristics like loss magnitude or confidence, offering a promising avenue for future gradient-aware robust optimization techniques.

5.2.13 Augmentation for Robustness

Core Idea Augmentation for robustness refers to applying transformations to training inputs—either label-preserving (e.g., flipping, cropping) or label-altering (e.g., MixUp)—to reduce the model's sensitivity to noise. These augmentations encourage the network to learn more general features and avoid overfitting mislabeled samples.

How It Helps Against Noise

- Label-preserving augmentations increase data diversity and regularize learning, indirectly improving robustness to noise.
- MixUp-like strategies create blended inputs and labels, softening the harmful impact of individual noisy labels by spreading them across multiple samples [95].

Formulation Example (MixUp) Given two samples (x_i, y_i) and (x_j, y_j), MixUp generates:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \quad \tilde{y} = \lambda y_i + (1 - \lambda)y_j, \quad \lambda \sim \text{Beta}(\alpha, \alpha)$$

Practical Impact These techniques are effective in label noise settings because they prevent the model from overfitting to mislabeled data by encouraging smoother decision boundaries.

5.2.14 Ensembling for Robustness

Core Idea Ensembling for robustness involves combining predictions from multiple models or checkpoints to reduce the effect of noise. Instead of relying on a single model's potentially overfitted predictions, ensembles average or smooth over multiple hypotheses, leading to more stable and less noise-sensitive outcomes.

How It Helps Against Noise Noise tends to affect individual models in inconsistent ways, especially when training data is corrupted. By aggregating outputs—whether across time (e.g., Stochastic Weight Averaging) or across models (e.g., bagging)—the ensemble reduces variance and filters out idiosyncratic responses to noisy samples.

Common Techniques

- **Stochastic Weight Averaging (SWA)** [27]: Averages model weights across epochs to obtain flatter minima and more generalizable predictions.
- **Snapshot Ensembles** [24]: Use different checkpoints during training as ensemble members.
- **Test-Time Ensembling**: Averages predictions over multiple augmentations of the same input.

Benefits These strategies make final predictions more robust, especially in scenarios with high label corruption, without needing explicit noise detection.

5.2.15 Consistency under Augmentation

A powerful empirical cue for detecting label noise is *consistency under augmentation*. This cue relies on a simple but effective assumption:

> If a sample is cleanly labeled, then the model's predictions should remain stable when that sample is transformed through standard data augmentations (e.g., flipping, cropping, or color jittering).

In contrast, noisy samples tend to produce unstable predictions under augmentation, since the model struggles to learn a coherent pattern from a mislabeled input. This behavior makes prediction inconsistency a useful signal for identifying potentially corrupted data.

Core Idea. The key idea is to compare predictions made on:

- the original input x
- an augmented version $\tilde{x}$

If the model produces consistent class probabilities for both inputs, the sample is likely clean. If not, the sample may be mislabeled or ambiguous.

Formally, this is captured using a consistency loss:

$$\mathcal{L}_{\text{consistency}} = \text{Divergence}(f(x), f(\tilde{x}))$$

where $f(x)$ is the model's output on the original input, and $f(\tilde{x})$ is the output on its augmented counterpart. The divergence can be defined as KL divergence, mean squared error, or cross-entropy between the predicted soft labels.

A high consistency loss across augmentations signals uncertainty or mismatch in the model's predictions, which can be indicative of a noisy label.

Use in Robust Learning. NCOD+ [80] builds upon the NCOD framework by introducing a *stability cue* that encourages the model to produce consistent predictions under augmentation. This consistency signal, when combined with centroid similarity and sample-level weighting, improves the model's ability to distinguish clean from noisy samples.

Consistency under augmentation thus enables self-supervised noise detection by observing how the model reacts to input perturbations, without needing ground-truth labels. It is especially powerful when used alongside other cues such as loss normalization or embedding regularity.

5.3 Summary of Empirical Cue Analysis

This chapter explored a complementary analytical perspective to architectural and algorithmic choices: **empirical cues**—observable behavioral patterns that arise during the training of deep neural networks with noisy labels. These cues, though not always explicitly modeled, serve as inductive signals that many robust training methods exploit to infer which labels are trustworthy and which are likely corrupted.

Each cue reflects a specific training-time signature of label noise, whether it be the tendency of clean samples to have lower loss, the instability of predictions on noisy examples, or the clustering of features in embedding space. Importantly, many state-of-the-art methods do not rely on a single cue, but rather combine multiple signals, such as loss dynamics, confidence trends, and consistency measures.

This behavioral layer of analysis complements the *feature-based taxonomy* from Chapter 4, offering deeper insight into how and why robust learning methods succeed in the presence of noise—not just what components they use. The integration of empirical cues with structural method design enables a more nuanced comparison across methods, paving the way for the comparative synthesis in Chapter 6.

In the next chapter, we bring together both axes of analysis—**feature taxonomy** and **empirical cues**—to assess each method holistically and highlight trade-offs, synergies, and innovation patterns across the noisy-label learning landscape.

Chapter 6

In-Depth Analysis of Selected State-of-the-Art Methods

6.1 Overview and Selection Criteria

The methods selected for in-depth analysis were chosen to cover the full range of the 11-feature taxonomy (Chapter 4) and to reflect recent, high-impact advances in learning with noisy labels. We prioritized diversity of approach (e.g., different noise models, learning paradigms, and tasks), recency and influence (methods from ∼2020–2024 in top venues), and empirical performance (state-of-the-art results on benchmarks). For example, many chosen methods achieve competitive accuracy under noise [69, 94], and appear in major conferences (e.g., ELR at NeurIPS 2020 [43], L2B and SURE at CVPR 2024 [100, 37]). In particular, we include both discriminative and generative strategies, various sample-weighting and regularization techniques, and applications from image classification to segmentation. For instance, ADELE and SURE report state-of-the-art results on challenging noisy segmentation and noisy classification tasks [42, 37], while NCOD and ProMix show significant accuracy gains on CIFAR-noise benchmarks [80, 86].

Together, the selected methods ensure broad coverage of taxonomy features (e.g., label assumptions, model architecture, learning objective) and encompass the latest trends in robust learning. Chapter 5 will use this set to conduct a feature-based comparative analysis of the methods' design and behavior.

To ensure a robust and comprehensive comparison, each method in this chapter is evaluated along two complementary axes: Feature-Based Taxonomy Analysis, as introduced in Chapter 4, and Empirical Cue Analysis, detailed in Chapter 5. This dual perspective enables both a structural understanding of each method's components and a behavioral interpretation of how it responds to label noise during training.

Together, this two-layered analysis—formal structure and empirical strategy—offers a richer understanding of how each method functions and why it performs well. It also facilitates the discovery of recurring design heuristics, shared assumptions, and potential synergies, laying a foundation for future developments in noise-robust learning. In the sections that follow, we apply this integrated analysis to a diverse set of representative methods, including:

- **ADELE** (Adaptive Early-Learning Correction, CVPR 2022): This segmentation method extends the early-learning phenomenon to semantic segmentation [42]. ADELE detects the onset of memorization per category and adaptively corrects noisy pixel labels, achieving state-of-the-art results on PASCAL VOC 2012 under synthesized annotation noise. It exemplifies a unique task (segmentation) and uses category-wise early stopping, thus covering taxonomy features not seen in plain classification. ADELE is also recent (2022) and highly regarded, adding diversity in domain and strong benchmark performance.

- **CORES** (Confidence Regularized Sample Sieve, ICLR 2021): CORES^2 tackles instance-dependent label noise, a challenging noise model. It progressively "sieves" out corrupted examples without knowing noise rates [12]. CORES demonstrates strong empirical performance on synthetic CIFAR-10/100 and real-world Clothing1M noise, and comes with theoretical guarantees for filtering out noise. Its inclusion ensures coverage of the instance-dependent-noise dimension of the taxonomy and of sample-selection strategies.

- **ELR / ELR+** (Early-Learning Regularization, NeurIPS 2020): ELR is a seminal method that leverages early-learning in classification tasks. It uses a regularization term that prevents overfitting to noisy labels by guiding the model toward pseudo-targets obtained in the early phase [43]. ELR (and its improved variant ELR+) achieves robustness on standard benchmarks, yielding accuracy comparable to the state of the art.

- **GNL** (Partial Label Supervision for Agnostic Generative Learning, 2023): This recent approach combines generative modeling with partial-label supervision [40]. GNL proposes a single-stage optimization that approximates image generation via classifier outputs and introduces a Partial Label Supervision (PLS) mechanism to capture label uncertainty. It reports state-of-the-art results on vision and NLP benchmarks while reducing computation.

- **ILL** (Imprecise Label Learning, NeurIPS 2024): ILL is a unified framework for various imprecise label types (noisy, partial, multiple candidates, etc.) [8]. It uses an EM-based model to treat precise labels as latent variables and optimizes over the full distribution of possible labels. Notably, ILL surpasses specialized methods in each setting, marking the first unified framework with robust and effective performance across various challenging settings.

- **L2B** (Learning to Bootstrap, CVPR 2024): L2B uses meta-learning to balance real and pseudo labels and to weight instances dynamically [100]. It learns a new objective that implicitly relabels data, guiding bootstrapping more effectively. L2B yields strong robustness improvements on both natural and medical tasks under synthetic and real noise, often requiring little to no clean validation data.

- **NCOD** (Noisy Centroids Outlier Discounting, arXiv 2023): NCOD combines learnable sample weighting with class-centroid distances [80]. It discounts the loss of samples far from their class centroids, under the intuition that

distant samples are more likely mislabeled. This approach outperforms prior techniques on benchmark classification tasks with noise.

- **NES** (Noisy Early Stopping, arXiv 2024): NES is a theoretical and practical early-stopping method that shows a noisy validation set can suffice for early stopping [76]. The authors prove that near-optimal stopping can be achieved by monitoring performance on a noisy set instead of requiring a clean validation set.

- **PADDLES** (Phase-Amplitude Disentangled Early Stopping, arXiv 2022): PADDLES proposes stopping CNN training at different times for amplitude vs. phase features of internal layers [25]. Empirically, it outperforms other early stopping methods and achieves state-of-the-art performance on noisy datasets.

- **PES** (Progressive Early Stopping, NeurIPS 2021): PES splits the network into parts and applies early stopping progressively [5]. Despite its simplicity, PES yields more stable results than conventional early stopping and achieves high accuracy under noise.

- **PGDF** (Prior Guided Denoising Framework, IJCAI 2022): PGDF generates per-sample priors to guide which samples are treated as clean or noisy [10]. It integrates these priors into sample splitting and semi-supervised labeling, improving pseudo-labels and benchmark performance.

- **ProMix** (IJCAI 2023): ProMix expands the set of clean samples using high-confidence selection, then applies a balanced SSL framework [86]. It significantly improves accuracy on multiple noisy benchmarks.

- **SOP** (Sparse Over-parameterized Training, ICML 2022): SOP treats label noise as a sparse over-parameterized component separate from the clean model [44]. It jointly trains a network and sparse noise model, achieving state-of-the-art test accuracy on various real datasets.

- **SURE** (SUrvey REcipes, CVPR 2024): SURE is an ensemble of techniques (regularization, calibration, etc.) for improving reliability and uncertainty estimation [37]. It achieves state-of-the-art performance on real-world noisy-label datasets without task-specific tuning.

In summary, the selected methods collectively span diverse taxonomy axes, are drawn from the latest research, and exhibit competitive or state-of-the-art performance on benchmarks. This careful selection sets the stage for Chapter 5, whose goal is to apply the 11-feature taxonomy to these methods and conduct a comparative, feature-based analysis of their mechanisms and behaviors.

6.2 Method-by-Method Review and Feature & Empirical Analysis

This section presents an in-depth examination of each selected method introduced in Section 5.1. The goal is to analyze and compare these approaches through the lens of both the feature-based taxonomy established in Chapter 4 and a complementary layer of empirical cue analysis. This dual perspective enables a more comprehensive understanding of each method's design rationale, operational behavior, and underlying assumptions about how label noise manifests and can be mitigated during training.

Each subsection below is dedicated to a single method and follows a structured three-part format:

Overview: A concise summary of the method's core motivation, high-level intuition, and primary contributions toward learning robustly under noisy labels. This section highlights how the method positions itself within the broader landscape of noisy-label learning.

Feature-Based Analysis: A detailed breakdown of how the method maps to the 11-feature taxonomy, covering four key dimensions: architectural structure, regularization mechanisms, learning dynamics, and dependence on auxiliary information. For each relevant feature, we discuss its presence or absence, and explain how the method operationalizes that component in practice.

Empirical Cue Analysis: Beyond architectural choices, many methods implicitly rely on observable patterns during training—referred to as *empirical cues*—to guide training in the presence of label noise. These cues include, for instance, small-loss heuristics, confidence thresholds, feature-space clustering, and prediction consistency across augmentations. In this section, we identify the specific cues each method leverages, explain how these are encoded or utilized within the training process, and highlight the method's empirical assumptions about noise.

This layered analysis—both structural (via the taxonomy) and behavioral (via empirical cues)—provides a richer and more precise basis for comparing state-of-the-art methods. It not only reveals shared design principles and innovation trends but also supports a grounded discussion in the upcoming comparative summary, where cross-method patterns and trade-offs will be synthesized.

6.2.1 ADELE

Overview of ADELE

ADELE (Adaptive Early-Learning Correction for Noisy Segmentation) [42] prevents segmentation networks from memorizing annotation errors by (1) detecting when early learning ends *per class*, and (2) stabilizing its corrective predictions via multiscale consistency.

First, for each class, ADELE tracks the IoU between the model's predictions and the noisy labels, denoted $IoU_m(t)$. It fits the smooth curve

$$f(t) = a(1 - e^{-bt^c})$$

and monitors its slope $f'(t)$. When

$$\frac{|f'(1) - f'(t)|}{|f'(1)|} > 0.9,$$

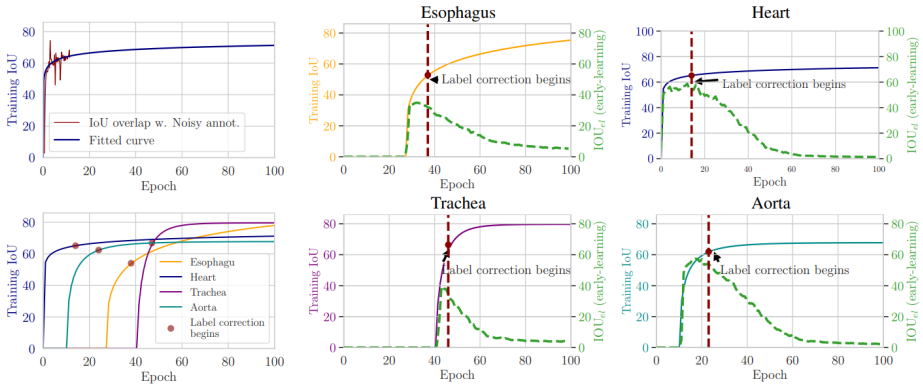


Figure 6.1. Curve-fitting on IoU between predictions and noisy labels (top left) detects per-class memorization onset. Label correction is triggered when the slope drops (bottom left) and aligns with the early-learning IoU peak for each organ (right). (Adapted from ADELE [42])

early learning is deemed over and memorization has begun. At that epoch, ADELE refurbishes the noisy labels by replacing them—only where model confidence ≥ 0.8—with the network's own predictions. As shown in Figure 6.1, this per-class correction aligns precisely with the end of the early-learning phase.

Second, to ensure these predictions are reliable, ADELE enforces multiscale consistency: it feeds each image at three scales ($0.7\times$, $1\times$, $1.5\times$) and obtains outputs $p_k(x)$. Defining the mean

$$q(x) = \frac{1}{3} \sum_k p_k(x),$$

it adds the KL-divergence loss

$$L_{MS} = -\frac{1}{3} \sum_{k=1}^{3} KL(p_k(x) \,\|\, q(x))$$

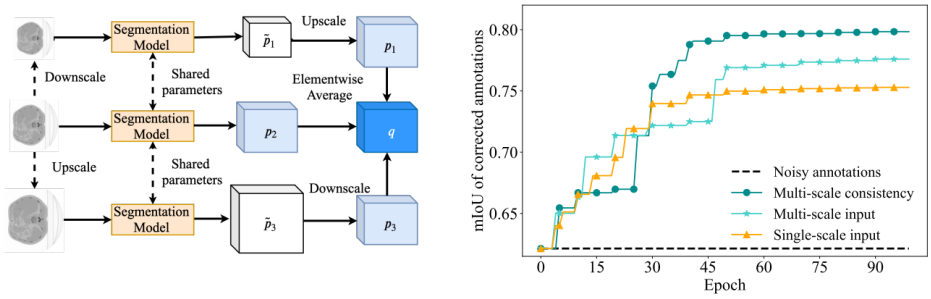


Figure 6.2. (Left) Multiscale-consistency regularization: three scaled inputs (0.7×, 1×, 1.5×) produce predictions p_k, which are aligned via KL divergence to their average q. (Right) The regularization yields sharper, more accurate corrected labels on SegTHOR. (Adapted from ADELE [42])

whenever $\max_i q_i(x) \geq 0.8$. Figure 6.2 illustrates how this term sharpens and stabilizes averaged predictions, enabling ADELE to correct labels robustly.

Together—adaptive label correction and multiscale regularization—ADELE requires no pre-training, no clean validation, and uses all samples, delivering significant mIoU improvements on both SegTHOR (medical CT) and PASCAL VOC (weakly supervised) benchmarks.

Feature-Based Taxonomy of Robust Learning Methods for ADELE

- **Architectural Structure**
  - **Using One Network:** ✓
    ADELE operates on a single segmentation model (e.g., U-Net or DeepLab) without requiring any duplicate or peer network.
  - **Using Two or Multiple Networks:** ✗
    Unlike co-teaching or DivideMix, ADELE does not employ ensembles or multiple networks; all corrections come from the same model.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    ADELE does not introduce any class-imbalance term; it corrects labels based on learning dynamics rather than rebalancing losses.
  - **KL Divergence Regularizer:** ✓
    The multiscale-consistency loss $L_{MS} = -\frac{1}{s}\sum_k KL(p_k \,\|\, q)$ explicitly uses KL divergence to align predictions across scales (Equation 3).
  - **Embedding Regularization:** ✗
    ADELE's regularization acts on output distributions, not on intermediate feature embeddings.
  - **Adding Additional Learnable Parameters:** ✗
    There are no extra modules (e.g., noise adaptation layers); ADELE only adds loss terms and label corrections.

- **Learning Dynamics**
  - **Loss Regularization Only:** ∼
    ADELE augments the cross-entropy loss with a KL-based regularizer, but also dynamically modifies the training labels—so it's more than pure loss adjustment.
  - **Add Contrastive Loss:** ✗
    ADELE does not include any contrastive or representation-level objective.
- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    All decisions (when to correct labels) are based on training-time IoU trends and confidence thresholds; ADELE requires no clean hold-out set.
  - **Add Semi-Supervised Loss:** ✗
    Although label correction resembles self-training, ADELE does not explicitly incorporate an unsupervised or semi-supervised consistency term beyond multiscale KL.
  - **Meta-Learning Framework:** ✗
    ADELE's hyperparameters (slope threshold, confidence cutoff) are fixed rather than learned via a meta-learning loop.

Empirical Cues in Noisy-Label Learning for ADELE

- **Small-Loss Criterion:** ✗
  ADELE does not prioritize samples by their training loss. Instead, it monitors IoU trends on noisy labels to detect memorization
- **Curriculum via Dynamic Thresholding:** ●
  ADELE uses a dynamic correction schedule based on IoU slope, which resembles curriculum pacing, though it does not explicitly rank examples by difficulty.
- **Temporal Consistency of Predictions:** ✗
  ADELE enforces consistency across multiple scales within a single epoch, not across epochs over time.
- **Noisy Validation Early-Stopping:** ●
  Rather than stopping training globally, ADELE uses the IoU curve on noisy labels to trigger per-class label correction—akin to an early-stopping signal—but still continues training with corrected labels.
- **Normalized Loss Metrics:** ✗
  ADELE does not compare per-sample losses to averages to identify outliers.
- **Layer-Wise Memorization Rates:** ✗
  ADELE tracks only overall IoU decay and does not analyze memorization at specific layers.

- **Model Confidence on Labels:** ✓
  ADELE only refurbishes a noisy pixel label if the model's prediction confidence exceeds 0.8, directly using confidence as a reliability cue.
- **Soft Label Refinement:** ✗
  ADELE replaces noisy labels with hard predictions rather than soft probability distributions.
- **Inter-Network Agreement:** ✗
  ADELE operates with a single model and does not use outputs from multiple networks for agreement-based correction.
- **Frequency-Domain Cues:** ✗
  There is no spectral decomposition or analysis of frequency characteristics in ADELE.
- **Sparsity of Noise (Label Errors as Outliers):** ✗
  ADELE does not model label noise as sparse or treat noisy labels explicitly as statistical outliers.
- **Gradient Coherence:** ✗
  ADELE does not track or utilize the alignment of gradients across samples.
- **Augmentation for Robustness:** ✓
  Feeding rescaled copies of each image (0.7×, 1×, 1.5×) acts as a form of data augmentation to improve robustness.
- **Ensembling for Robustness:** ●
  ADELE performs ensemble-like averaging of predictions across input scales but does not use multiple model snapshots or checkpoints.
- **Consistency under Augmentation:** ✓
  The multiscale-consistency loss (L_{MS}) explicitly enforces that predictions remain stable across those augmented (rescaled) inputs.

Summary of ADELE (Adaptive Early-Learning Correction)

ADELE is a simple yet powerful method for improving segmentation when labels are noisy. It watches each class's training performance to spot when the model shifts from learning true patterns to memorizing errors. At that moment, ADELE replaces the bad annotations with the model's own high-confidence predictions, ensuring the network retains correct structure. To make those predictions trustworthy, it also enforces consistency by averaging outputs over multiple input scales and penalizing any disagreement. ADELE trains from scratch, uses all available data, and needs no clean validation labels—yet delivers large accuracy gains on both medical (*SegTHOR*) and weakly supervised (*PASCAL VOC*) benchmarks.

6.2.2 CORES

Overview of CORES

CORES (Confidence Regularized Sample Sieve) [12] In the presence of instance-dependent label noise, standard cross-entropy training overfits by memorizing corrupted labels. CORES^2 overcomes this by encouraging confident predictions and then sieving out likely-noisy examples in a multi-round procedure.

Confidence Regularization For each sample $(x_n, \tilde{y}_n)$, CORES^2 adds a "confidence" penalty to the usual cross-entropy:

$$\ell_{\mathrm{CR}}(f(x_n)) = -\beta\, \mathbb{E}_{\tilde{Y}\sim P(\tilde{Y})}[\,\ell(f(x_n), \tilde{Y})\,],$$

where $\ell(f(x), y) = -\ln f(x)[y]$ and $\beta > 0$. Minimizing

$$\ell(f(x_n), \tilde{y}_n) + \ell_{\mathrm{CR}}(f(x_n))$$

forces $f(x_n)$ to concentrate its mass on a single class (Theorem 1), thereby resisting noisy labels.

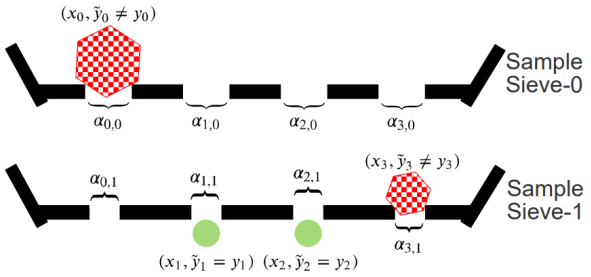


Figure 6.3. Dynamic sample sieves. Green circles are clean examples. Red hexagons are corrupted examples. Adapted from [12].

Sample Sieve Define a per-sample threshold α_n as the average (regularized) loss over all K classes under a frozen copy $\bar{f}$:

$$\alpha_n = \frac{1}{K}\sum_{y=1}^{K}\left[\ell(\bar{f}(x_n), y) + \ell_{\mathrm{CR}}(\bar{f}(x_n))\right].$$

We then introduce binary weights $v_n \in \{0, 1\}$, indicating whether to keep ($v_n = 1$) or discard ($v_n = 0$) sample n, by comparing its current regularized loss to α_n:

$$v_n(t) = \mathbb{1}\left\{\ell(f^{(t)}(x_n), \tilde{y}_n) + \ell_{\mathrm{CR}}(f^{(t)}(x_n)) < \alpha_n(t)\right\}.$$

Training alternates between:

$$f^{(t)} = \arg\min_{f}\sum_{n=1}^{N} v_n(t-1)\,[\ell(f(x_n), \tilde{y}_n) + \ell_{\mathrm{CR}}(f(x_n))]$$

and the sieve update above. This dynamic, multi-round filter safely removes corrupted labels without knowing noise rates. As illustrated in Figure 6.3, each iteration refines the "aperture" $\alpha_n(t)$ of the sample sieve.

CORES^2 requires no pre-training or external noise estimates, supports arbitrary architectures, and achieves strong robustness even under heavy, instance-dependent noise (e.g., 40% on CIFAR-10/-100).

Feature-Based Taxonomy of Robust Learning Methods for CORES^2

- **Architectural Structure**
  - **Using One Network:** ✓
    CORES^2 uses a single backbone network f, plus a frozen copy $\bar{f}$ for threshold computation, but does not duplicate or ensemble multiple full networks.
  - **Using Two or Multiple Networks:** ✗
    Unlike Co-teaching or DivideMix, CORES^2 maintains only one primary trainable model (with an auxiliary frozen snapshot), not two or more jointly trained networks.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    CORES^2's regularizer is confidence-based (penalizing prediction entropy), not designed to correct for class imbalance.
  - **KL Divergence Regularizer:** ✓
    In the consistency training phase (after sieving), CORES^2 applies a KL-divergence loss between $f(x)$ and $f(\text{aug}(x))$ on the "discarded" (noisy) examples.
  - **Embedding Regularization:** ✗
    Regularization is applied to the output probabilities (via ℓ_{CR}) or via consistency, not directly to the internal feature embeddings.
  - **Adding Additional Learnable Parameters:** ✗
    CORES^2 does not augment the network with dedicated noise-adaptation layers or extra modules—it only changes the loss and sample-selection logic.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✓
    The core of CORES^2 is the confidence regularizer $\ell_{\text{CR}}(f(x)) = -\beta\, \mathbb{E}_{\tilde{Y}}[\ell(f(x), \tilde{Y})]$ added to the cross-entropy, plus the sample-sieve thresholds. No architectural changes are required.
  - **Add Contrastive Loss:** ✗
    CORES^2 does not use contrastive objectives. Its unsupervised component is a consistency (KL) loss, not a contrastive loss.
- **Dependence on Auxiliary Information**

  - **Needs Clean Validation Dataset:** ✗
    All thresholds and regularization are computed solely from the noisy training set; no trusted clean set is used.
  - **Add Semi-Supervised Loss:** ✓
    The consistency (KL) loss on the sieved-out (noisy) samples is effectively a semi-supervised component—using their inputs (without trusting their labels) to improve generalization.
  - **Meta-Learning Framework:** ✗
    CORES² does not employ a meta-learning loop to adjust weights or hyperparameters; its sample weights v_n are updated via a fixed thresholding rule, not via learned meta-gradients.

Empirical Cues in Noisy-Label Learning for CORES²

- **Small-Loss Criterion:** ✓
  CORES² retains only those samples whose confidence-regularized loss falls below a per-sample threshold. In effect, it keeps the "small-loss" (low regularized loss) examples as "clean."
- **Curriculum via Dynamic Thresholding:** ✓
  CORES²'s per-sample threshold α_n is recomputed each round using the frozen model's average regularized loss. Early rounds admit only the easiest (lowest-loss) examples; later rounds allow progressively harder—but still clean—examples back into training. This self-paced inclusion mirrors curriculum learning.
- **Temporal Consistency of Predictions:** ✗
  CORES² does not explicitly enforce or track stability of a sample's class probabilities across epochs (beyond the consistency under augmentation). There's no penalty or criterion based on how a given example's prediction drifts over time.
- **Noisy Validation Early-Stopping:** ✗
  CORES² does not rely on monitoring validation accuracy (noisy or clean) for early stopping; it instead performs a fixed number of alternate training + sieving rounds.
- **Normalized Loss Metrics:** ●
  CORES² normalizes each sample's loss by comparing it to its own average regularized loss α_n across classes. This per-sample normalization echoes the idea of flagging outliers via relative (normalized) loss.
- **Layer-Wise Memorization Rates:** ✗
  CORES² does not analyze or exploit differences in how individual layers memorize clean vs. noisy examples; it works purely at the output-loss level.
- **Model Confidence on Labels:** ✓
  The core of CORES² is the confidence regularizer ℓ_{CR}, which explicitly encourages higher softmax confidence on the (presumed) correct label and uses that confidence to filter examples.

- **Soft Label Refinement:** ✗
  CORES^2 discards or down-weights noisy examples rather than replacing their labels with soft targets. It does not refine hard labels into soft ones via ensemble averages or latent structure.

- **Inter-Network Agreement:** ✗
  CORES^2 uses a single network (plus a frozen snapshot) rather than two or more networks whose mutual agreement would signal label reliability. No co-training or peer-network agreement mechanism is applied.

- **Frequency-Domain Cues:** ✗
  There is no decomposition of input features into phase/amplitude or frequency components—the method remains entirely in the standard data/feature domain.

- **Sparsity of Noise (Label Errors as Outliers):** ✗
  Although CORES^2 treats noisy samples as "outliers" to be sieved out, it does not explicitly model noise via sparse parameter matrices or an outlier-parameter term.

- **Gradient Coherence:** ●
  The confidence regularizer (peer-loss–derived) amplifies gradient signals when multiple clean examples agree and suppresses noisy gradients. Although CORES^2 does not measure coherence explicitly, its design leverages the empirical fact that gradients from clean data reinforce each other while noisy gradients tend to cancel out.

- **Augmentation for Robustness:** ✓
  After sieving out noisy examples, CORES^2 applies an unsupervised consistency loss (KL divergence) between each sample's prediction and its augmented counterpart, thereby leveraging data augmentation.

- **Ensembling for Robustness:** ✗
  CORES^2 does not average predictions across multiple models or checkpoints; it uses a single model and a frozen snapshot only for threshold calculation.

- **Consistency under Augmentation:** ✓
  The post-sieve phase enforces that the network's output for x and its augmented version $\text{aug}(x)$ remain consistent (via a KL-divergence penalty), explicitly leveraging this empirical cue.

6.2.3 ELR and ELR$^+$

Overview of (ELR) and ELR$^+$

Early-Learning Regularization (ELR) [43] is a robust training method designed to mitigate the memorization of noisy labels in deep neural networks. It is based on the empirical observation that during training, networks initially learn from clean labels (early-learning phase) before gradually overfitting to noisy labels (memorization phase). ELR leverages this early-learning phenomenon by encouraging the model to remain consistent with its earlier, more reliable predictions, thus providing resilience against label noise.

At the heart of ELR is a regularization term that penalizes deviation from target predictions $t^{(i)}$, which are computed as an exponential moving average of the model's previous outputs for each training sample:

$$t^{(i)}(k) = \beta \cdot t^{(i)}(k-1) + (1-\beta) \cdot p^{(i)}(k),$$

where $p^{(i)}(k)$ is the model's softmax output at iteration k, and $\beta \in [0, 1)$ is the momentum term.

The ELR loss function combines the standard cross-entropy loss with a regularization penalty:

$$\mathcal{L}_{\text{ELR}}(\Theta) = \mathcal{L}_{\text{CE}}(\Theta) + \frac{\lambda}{n} \sum_{i=1}^{n} \log(1 - \langle p^{(i)}, t^{(i)} \rangle),$$

where $\langle p^{(i)}, t^{(i)} \rangle$ measures the agreement between current predictions and the historical target. The regularization term penalizes large deviation from early beliefs, thus preserving clean signal and resisting memorization.

To understand the regularizer's effect, the authors analyze the gradient of the ELR loss:

$$\nabla \mathcal{L}_{\text{ELR}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\mathcal{N}_\Theta(x^{(i)})} \left(p^{(i)} - y^{(i)} + \lambda g^{(i)} \right),$$

where $g^{(i)} \in \mathbb{R}^C$ is the gradient correction term defined by:

$$g_c^{(i)} = \frac{p_c^{(i)}}{1 - \langle p^{(i)}, t^{(i)} \rangle} \sum_{k=1}^{C} (t_k^{(i)} - t_c^{(i)}) p_k^{(i)}.$$

This correction serves two purposes:

- For clean labels, it maintains significant gradient signal even after cross-entropy vanishes.
- For noisy labels, it dampens harmful gradients, suppressing memorization.

As shown in Figure 6.4, the model trained with standard cross-entropy initially learns correct labels but eventually overfits to noisy labels, as indicated by the rise in memorized wrong predictions (in red). In contrast, the ELR-trained model sustains performance on both clean and noisy examples by continuing to learn from clean labels and resisting memorization of the noise.

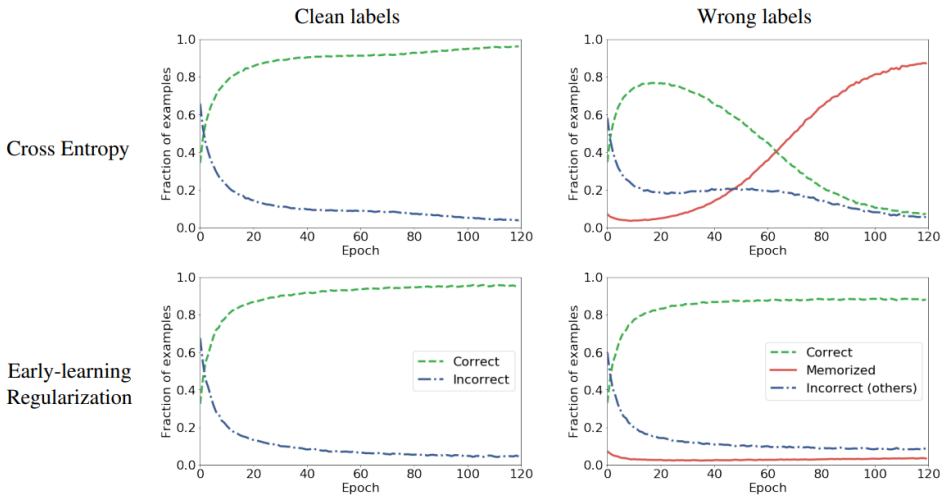


Figure 6.4. Figure 1: Comparison of a ResNet-34 trained with standard cross-entropy loss (top row) and with ELR (bottom row) on CIFAR-10 with 40% label noise. **Left:** Clean-label prediction accuracy (green: correct, blue: incorrect). **Right:** Noisy-label behavior (green: true class predicted, red: memorized noise, blue: incorrect prediction). The ELR model maintains high accuracy while suppressing memorization of noisy labels. adapted from "Early-Learning Regularization Prevents Memorization of Noisy Labels"[43]

ELR$^+$: Enhancing ELR with Semi-Supervised Techniques

ELR$^+$ extends ELR by incorporating additional mechanisms inspired by semi-supervised learning to further improve robustness, especially under high noise rates. The key enhancements in ELR+ are:

- **Weight Averaging:** Maintains a moving average of model weights during training to produce more stable predictions.
- **Dual Networks:** Uses two separate neural networks, where each network generates targets for the other (a co-teaching-inspired setup). This reduces confirmation bias.
- **Mixup Augmentation:** Applies mixup to both inputs and targets, blending random examples and labels. This encourages smoother decision boundaries and reduces overfitting.

Each of these components independently contributes to improved performance. As demonstrated in the ablation studies in the paper, ELR$^+$ consistently outperforms basic ELR and other state-of-the-art methods on noisy benchmarks such as CIFAR-10 and CIFAR-100, even under extreme noise conditions (e.g., 80%–90% symmetric or asymmetric noise).

Summary

ELR and its extension ELR$^+$ provide simple yet powerful regularization-based frameworks for learning from noisy labels. Unlike methods that rely on sample selection, loss reweighting, or label correction, ELR does not require any supervision, clean data, or prior knowledge of the noise structure. It can be seamlessly integrated

into any architecture, trained from scratch, and scaled to real-world noisy datasets such as Clothing1M and WebVision. By anchoring learning to the early predictions, ELR effectively balances clean signal preservation and noise suppression, offering a principled solution to the label noise challenge.

Feature-Based Taxonomy of ELR and ELR+

- **Architectural Structure**
  - **Using One Network:** ✓
    ELR operates with a single deep neural network. ELR+ may optionally use two networks, but this is an enhancement—not a requirement in the base method.
  - **Using Two or Multiple Networks:** ✓ (ELR+ only)
    ELR+ optionally uses two networks in a co-teaching-inspired setup, where each network generates targets for the other to reduce confirmation bias. ELR alone uses only one network.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    Neither ELR nor ELR+ includes any adjustment to compensate for class imbalance in the loss function.
  - **KL Divergence Regularizer:** ✗
    KL divergence is not used. Instead, ELR uses a custom log-based regularization term, which is designed to preserve early-learning predictions more effectively.
  - **Embedding Regularization:** ✗
    ELR regularizes predictions (output probabilities), not internal embeddings or feature representations.
  - **Adding Additional Learnable Parameters:** ✗
    No noise adaptation layers or extra modules are introduced. The method relies solely on modified loss functions.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✓
    ELR and ELR+ both fit this category. The method modifies the training objective by adding a regularization term to the standard cross-entropy loss without altering the network architecture.
  - **Add Contrastive Loss:** ✗
    Contrastive objectives are not used in either version of the method.
- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    ELR and ELR+ do not require a clean validation set or any ground-truth labels to detect noise or stop training.

- **Add Semi-Supervised Loss:** ✓ (ELR+ only)
  ELR+ incorporates ideas from semi-supervised learning, such as:
  * Temporal ensembling
  * Weight averaging
  * Mixup

  These additions help exploit unlabeled-style generalization dynamics under label noise.
- **Meta-Learning Framework:** ✗
  Neither ELR nor ELR+ involves meta-learning or dynamic sample reweighting based on learned gradients or external supervision.

Empirical Cue Analysis of ELR and ELR+

- **Small-Loss Criterion:** ✗
  ELR does not select or prioritize small-loss samples. Instead, it regulates all samples uniformly by anchoring predictions to earlier outputs, rather than using loss magnitude to infer clean examples.
- **Curriculum via Dynamic Thresholding:** ✗
  ELR/ELR+ does not select or filter samples during training based on dynamic thresholds or sample easiness.
- **Temporal Consistency of Predictions:** ✓
  ELR tracks moving averages of model predictions over time and penalizes deviations from this average. This stabilizes predictions and discourages abrupt class switches, particularly important for noisy labels. It directly encodes temporal consistency as part of the loss via the term $\log(1 - \langle p, t \rangle)$.
- **Noisy Validation Early-Stopping:** ✗
  ELR does not rely on validation performance to stop training. In fact, it is specifically designed to function without any clean validation set.
- **Normalized Loss Metrics:** ✗
  ELR does not compare per-sample loss to any global loss statistics. It does not use loss normalization or thresholding to detect noise.
- **Layer-Wise Memorization Rates:** ✗
  ELR does not analyze or exploit layer-specific learning behavior. It operates entirely at the output level (softmax predictions).
- **Model Confidence on Labels:** ✓
  ELR relies on the model's own historical predictions as targets, assuming that early predictions are more reliable. This is a form of using model confidence accumulated over time.
- **Soft Label Refinement:** ✓
  Rather than relying solely on hard one-hot labels (which may be wrong), ELR regularizes the model toward soft target vectors $t^{(i)}$, derived from earlier

predictions. This refines the supervision signal by integrating historical model beliefs and smooth distributions.

- **Inter-Network Agreement:** ✗
  Even though ELR+ uses two networks, it does not measure or encourage prediction agreement between them. Each network independently uses the other's outputs to update its own target, but there is no symmetry or agreement-based logic such as KL divergence, JS divergence, or disagreement filtering.

- **Frequency-Domain Cues:** ✗
  ELR does not involve any spectral analysis or frequency-based separation between noise and signal.

- **Sparsity of Noise (Label Errors as Outliers):** ✗
  ELR does not model noise as outliers nor does it use any sparse noise prior or dedicated parameters for that purpose.

- **Gradient Coherence:** ✓
  ELR encourages gradient updates to align with early-learning direction, which tends to originate from clean labels. By maintaining influence from earlier trusted predictions, ELR reduces the risk of diverging gradients caused by noisy labels. This leads to more stable, coherent gradient signals, even without explicitly modeling this property.

- **Augmentation for Robustness:** ✓ (ELR+ only)
  ELR+ incorporates Mixup augmentation, which helps smooth decision boundaries and reduce the impact of label noise.

- **Ensembling for Robustness:** ✓ (ELR+ only)
  ELR+ uses both:
  - Temporal ensembling of predictions for target estimation.
  - Optionally, weight averaging across training steps for more stable outputs.

- **Consistency under Augmentation:** ✓ (ELR+ only)
  With Mixup and dual-network setups, ELR+ promotes prediction stability under input perturbations and model variation, aligning with this empirical cue.

6.2.4 GNL

Overview of GNL: Generative Noisy Label Learning with Partial Label Supervision

Generative Noisy Label Learning (GNL) [40] is a robust training framework developed to handle noisy labels in both vision and language tasks. Unlike traditional generative methods that require variational latent representations (e.g., VAEs) or fixed causal assumptions, GNL introduces a single-stage, causality-agnostic framework. This approach bypasses the need to model latent image variables and flexibly supports both generative ($p(x|y)$) and discriminative ($p(y|x)$) learning under label noise.

The core objective of GNL is to infer the hidden clean label y from a noisy label $\tilde{y}$, while estimating the desired distribution $p(x|y)$ or $p(y|x)$. This is achieved using an EM-style training objective. When assuming the causal direction $Y \to X$, the variational bound is:

$$\mathbb{E}_{q(y|x)}[\log p(x|y)] = \mathbb{E}_{q(y|x)}[\log p(\tilde{y}|x,y)] - \mathrm{KL}[q(y|x)\|p(x|y)p(y)] + \mathrm{KL}[q(y|x)\|p(\tilde{y}|x,y)p(y)].$$

In the alternative causal direction $X \to Y$, the formulation becomes:

$$\mathbb{E}_{q(y|x)}[\log p(y|x)] = \mathbb{E}_{q(y|x)}[\log p(\tilde{y}|x,y)] - \mathrm{KL}[q(y|x)\|p(x|y)p(y)] + \mathrm{KL}[q(y|x)\|p(\tilde{y}|x,y)p(x)].$$

A key contribution of GNL is its **Partial Label Supervision (PLS)** mechanism, which builds an instance-specific soft prior $p(y)$. This prior combines the noisy label $\tilde{y}_i$, a confidence-weighted candidate label distribution c_i, and an uncertainty term u_i as follows:

$$p_i(y^{(j)}) = \frac{\tilde{y}_i^{(j)} + c_i^{(j)} + u_i^{(j)}}{Z},$$

where Z normalizes the distribution. Here, c_i is estimated from the moving average of model predictions, and u_i captures uncertainty by sampling from a uniform distribution scaled by prediction entropy.

To avoid costly generative modeling, GNL approximates $p(x|y)$ with classifier outputs:

$$p(x|y) \approx \frac{q(y|x)}{\sum_i q(y|x_i)}.$$

The overall training loss combines three terms: cross-entropy for fitting the noisy label, a KL divergence for matching the refined prior, and a KL divergence corresponding to the EM E-step:

$$L = L_{\mathrm{CE}} + L_{\mathrm{PRI}} + L_{\mathrm{KL}}. \tag{6.1}$$

As illustrated in Figure 6.5 of the original paper, the framework integrates the model components and training strategy in a unified pipeline, showing the flow from noisy labels and predictions to refined priors and the final classification output. The figure also visualizes example CIFAR-10 images with their associated soft partial labels under 40% instance-dependent noise, clearly demonstrating how GNL represents label uncertainty during training.

Through extensive experiments on synthetic and real-world noisy datasets—including CIFAR-10/100, Clothing1M, and AGNews—GNL achieves state-of-the-art

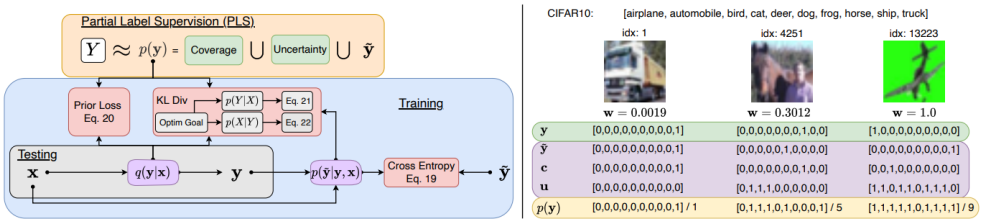


Figure 6.5. Overview of the GNL framework. Left: Model training structure based on EM optimization and Partial Label Supervision, where $q(y|x)$ is the variational posterior and $p(\tilde{y}|x, y)$ models label noise. The prior $p(y)$ is dynamically built using coverage and uncertainty. Right: Example CIFAR-10 images with soft partial labels under 40% instance-dependent noise. Adapted from [40].

performance, while requiring no clean supervision or pretraining. Its support for both causal directions and capacity to handle instance-dependent noise highlight its flexibility and robustness.

GNL achieves state-of-the-art performance on CIFAR-10/100, Clothing1M, and AGNews under high instance-dependent noise, requiring neither clean labels nor pretraining. Its flexibility in handling both $X \to Y$ and $Y \to X$ makes it a versatile solution for noisy-label learning.

Feature-Based Taxonomy of Robust Learning Methods for GNL

- **Using One Network:** ✓
  GNL is trained using a single neural network (e.g., ResNet, BERT), with no auxiliary or co-training models. All learning (clean label estimation and noise modeling) is performed internally using the same network.

- **Using Two or Multiple Networks:** ✗
  GNL does not use dual networks or mutual training frameworks like Co-teaching, DivideMix, or JoCoR. Optional ensembling at test time does not count toward this category, which refers to training-time architecture.

- **Class-Balanced Regularizer:** ✗
  GNL does not explicitly address class imbalance in the label distribution. There is no use of class-frequency-aware loss weighting or resampling techniques.

- **KL Divergence Regularizer:** ✓
  KL divergence is a core component of GNL's loss function. It is used in both the M-step and E-step of the EM-inspired training to align predicted distributions $q(y \mid x)$ with the soft label prior $p(y)$.

- **Embedding Regularization:** ✗
  GNL does not apply regularization to intermediate feature representations or embeddings. Its regularization operates at the level of output distributions.

- **Adding Additional Learnable Parameters:** ✗
  GNL does not introduce extra modules like noise adaptation layers, memory

banks, or external parameter blocks. All parameters belong to the standard classifier and noise transition module implemented within the network.

- **Loss Regularization Only:** ✗
  Although GNL uses loss terms involving KL divergence and cross-entropy, it modifies the label supervision itself (via PLS), not just the loss. Thus, it goes beyond "loss regularization only."

- **Add Contrastive Loss:** ✗
  GNL does not incorporate contrastive objectives (e.g., SimCLR-style losses, representation alignment, or pairwise separation) in its training.

- **Needs Clean Validation Dataset:** ✗
  GNL is fully self-supervised with respect to noise. It does not require any clean subset, clean validation set, or prior knowledge of the noise rate for hyperparameter tuning or early stopping.

- **Add Semi-Supervised Loss:** ✗
  GNL does not integrate semi-supervised components such as consistency regularization on unlabeled data, pseudo-labeling of unannotated inputs, or MixMatch-style training.

- **Meta-Learning Framework:** ✗
  There is no meta-learning loop or optimization of sample weights or learning rates through meta-gradients. GNL operates with standard backpropagation using soft label supervision.

Empirical Cues in Noisy-Label Learning for GNL

- **Small-Loss Criterion:** ✗
  GNL does not prioritize low-loss samples. It treats all samples equally but assigns them soft priors based on prediction confidence and uncertainty, rather than assuming small-loss means clean.

- **Curriculum via Dynamic Thresholding:** ●
  There is no explicit thresholding, but GNL's uncertainty component u_i acts as a soft curriculum: noisy or uncertain samples receive flatter priors and contribute less to training, allowing the model to gradually focus on cleaner samples.

- **Temporal Consistency of Predictions:** ✓
  GNL maintains a moving average of predictions $C_i(t)$, which emphasizes prediction stability over time. Samples with consistent predictions are considered more reliable and receive sharper priors.

- **Noisy Validation Early-Stopping:** ✗
  GNL does not rely on validation accuracy (clean or noisy) to stop training. Training runs for a fixed number of epochs, independent of validation trends.

- **Normalized Loss Metrics:** ✗
  GNL does not compare a sample's loss to the class or batch average. Instead, it estimates noise through the magnitude of prediction loss ℓ_i and builds uncertainty-aware priors.
- **Layer-Wise Memorization Rates:** ✗
  GNL does not monitor or exploit differences in memorization across layers. The method operates at the output level, without inspecting internal layer behavior.
- **Model Confidence on Labels:** ✓
  GNL heavily relies on model confidence via the softmax output $q(y \mid x)$ to determine how strongly a label should be trusted. Higher confidence results in sharper priors.
- **Soft Label Refinement:** ✓
  This is the core mechanism of GNL. It replaces noisy hard labels with soft label priors $p(y)$, combining the noisy label, prediction history, and uncertainty into a refined target.
- **Inter-Network Agreement:** ✗
  GNL is a single-network method. There is no use of co-training or agreement between multiple networks.
- **Frequency-Domain Cues:** ✗
  GNL does not analyze spectral components (e.g., amplitude vs. phase) in feature representations. It operates purely in the time/activation domain.
- **Sparsity of Noise (Label Errors as Outliers):** ✗
  GNL does not assume label noise is sparse or use sparsity-based regularization. It handles noise probabilistically across the entire dataset.
- **Gradient Coherence:** ●
  GNL does not measure gradient alignment directly, but samples with unstable predictions and high loss are treated as uncertain. This indirectly downweights samples with divergent gradients.
- **Augmentation for Robustness:** ●
  GNL applies basic data augmentations (e.g., random crop and horizontal flip) during training, especially on vision datasets like CIFAR-10/100. However, it does not incorporate advanced augmentation strategies such as Mixup, CutMix, or consistency-based augmentations.
- **Ensembling for Robustness:** ●
  In some experiments (e.g., on real-world datasets like Animal-10N), GNL uses a two-model ensemble at inference time for robustness. However, this is not part of training and is optional.
- **Consistency under Augmentation:** ✗
  GNL does not enforce consistency across augmented inputs. There is no loss term that aligns predictions over different views of the same input.

6.2.5 ILL

Overview of ILL: Imprecise Label Learning: A Unified Framework

Real-world datasets often come with imprecise labels—noisy annotations, only candidate label sets, or a mix of labeled and unlabeled data—making standard supervised learning infeasible. Prior work has addressed each scenario (partial labels, noisy labels, semi-supervision) with bespoke methods, which do not generalize when multiple forms of imprecision coexist.

Imprecise Label Learning (ILL) [8] treats the unknown true labels Y as latent and models the observed data X and imprecise information I via a single missing-data maximum-likelihood objective. Formally, letting:

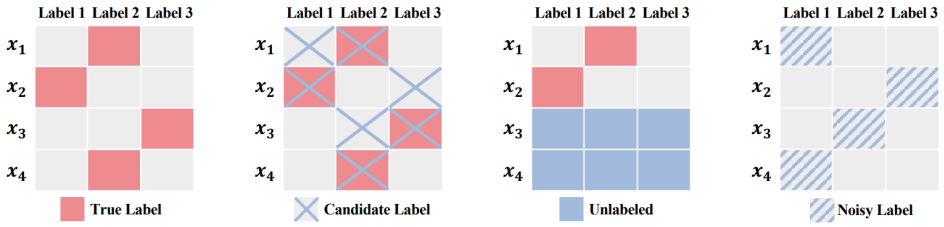
$$P(X, I; \theta) = \sum_{Y} P(X, Y, I; \theta),$$

ILL seeks:

$$\theta^* = \arg\max_{\theta} \log \sum_{Y} P(X, Y, I; \theta),$$

which is optimized using the Expectation-Maximization (EM) algorithm:

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{Y|X,I;\theta^{(t)}}[\log P(X, Y, I; \theta)] = \arg\max_{\theta} \mathbb{E}_{Y|X,I;\theta^{(t)}}[\log P(Y|X; \theta) + \log P(I|X, Y; \theta)].$$



(a) Full Label (b) Partial Label (c) Semi-Supervised (d) Noisy Label

Figure 6.6. Illustration of full and imprecise label configurations using a 4-sample, 3-class dataset. (a) Full label: one correct label; (b) Partial label: label is a candidate set containing the true class; (c) Semi-supervised: only part of the data is labeled; (d) Noisy label: observed label may be incorrect. Adapted from: "Imprecise Label Learning: A Unified Framework" [8]

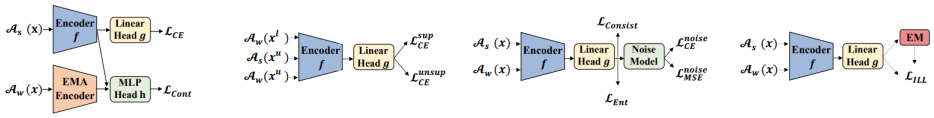
(As illustrated in Figure 6.6, ILL subsumes full, partial, noisy, and unlabeled settings under one EM pipeline.)

Special Cases

Partial Label Learning (PLL): Here, I is the candidate set $s \ni y$. Since $P(I|Y)$ is independent of θ, the EM loss reduces to a soft-target cross-entropy over s:

$$L_{\text{PLL}} = -\sum_{k \in s} P_t(y = k \mid x) \log P(y = k \mid x; \theta).$$

(See Figure 6.7a.)



(a) Partial Label (b) Semi-Supervised (c) Noisy Label (d) Imprecise Label

Figure 6.7. Baseline model pipelines under various imprecise label settings. (a) PiCO [78] for partial labels. (b) FixMatch [67] for semi-supervised learning. (c) SOP [44] for noisy labels. (d) Unified EM-based ILL framework that handles mixed imprecise labels. Adapted from: "Imprecise Label Learning: A Unified Framework" [8]

Semi-Supervised Learning (SSL): Given labeled data (X_L, Y_L) and unlabeled X_U, we recover:

$$L_{\text{SSL}} = -\sum_{(x,y)\in X_L} \log P(y \mid x;\theta) - \sum_{x\in X_U}\sum_{y} P_t(y \mid x)\log P(y \mid x;\theta),$$

i.e., a supervised cross-entropy plus a consistency term on pseudo-labels. (See Figure 6.7b.)

Noisy Label Learning (NLL): Observing $\hat{y}$ corrupted by a noise transition $T(\hat{y} \mid y;\omega)$, ILL yields:

$$L_{\text{NLL}} = -\sum_{y} P_t(y \mid x,\hat{y}) \log\left(\sum_{y'} P(y' \mid x;\theta)T(\hat{y} \mid y';\omega)\right) - \log T(\hat{y} \mid y;\omega),$$

which jointly fits the classifier and the noise model. (See Figure 6.7c.)

Mixed Imprecise Label Learning (MILL): When partial, noisy, and unlabeled data co-occur, one simply sums the above terms:

$$L_{\text{MILL}} = \underbrace{L_{\text{PLL}}}_{\text{partial terms}} + \underbrace{L_{\text{NLL}}}_{\text{noisy terms}} + \underbrace{L_{\text{SSL}}}_{\text{unlabeled terms}},$$

without any ad-hoc modification. (See Figure 6.7d.)

As illustrated in Figures 6.6–6.7, ILL provides a single EM-based pipeline that recovers—and often improves upon—prior state-of-the-art methods for each imprecise-label scenario.

Feature-Based Taxonomy of Robust Learning Methods for ILL

- **Using One Network:** ✓
  ILL formulates all imprecise-label scenarios in a single EM pipeline driven by one underlying classifier $P(y \mid x;\theta)$. There's no duplication of the backbone network across branches.
- **Using Two or Multiple Networks:** ✗
  Unlike Co-teaching or DivideMix, ILL does not rely on peer networks or mutual teaching; it uses posterior inference over one model.

- **Class-Balanced Regularizer:** ✗
  ILL's losses derive from EM expectations (soft-target CE, noise-transition likelihoods), but there's no explicit re-weighting term to correct for class imbalance in noisy labels.

- **KL Divergence Regularizer:** ✗
  Although the EM step enforces consistency between the model's distribution and the posterior $P_t(y \mid x, I)$, this appears as a cross-entropy or consistency loss—not an explicit KL penalty added to the loss.

- **Embedding Regularization:** ✗
  ILL does not apply any regularizer directly on the intermediate feature representations (e.g., norm penalties, distance constraints). All regularization is performed through label-level consistency objectives.

- **Adding Additional Learnable Parameters:** ✓
  In the noisy-label setting, ILL jointly learns a noise-transition model $T(\hat{y} \mid y; \omega)$, introducing extra parameters ω alongside the classifier's θ.

- **Loss Regularization Only:** ✓
  ILL's core innovation is to reinterpret various imprecise-label heuristics as EM-derived consistency losses—modifying only the loss without introducing architectural changes (except for the optional noise model).

- **Add Contrastive Loss:** ✗
  ILL does not incorporate contrastive or representation-learning losses; it remains fully within the probabilistic EM and soft-label consistency framework.

- **Needs Clean Validation Dataset:** ✗
  ILL does not require any clean validation set for hyperparameter tuning or early stopping. All forms of label imprecision are handled directly in the EM likelihood.

- **Add Semi-Supervised Loss:** ✓
  For unlabeled samples, ILL includes a standard SSL consistency term (soft-target cross-entropy on pseudo-labels), effectively recovering methods like FixMatch under the EM view.

- **Meta-Learning Framework:** ✗
  ILL does not involve meta-learning or higher-order optimization to reweight samples. It relies purely on closed-form EM updates and maximum-likelihood estimation.

Empirical Cues in Noisy-Label Learning for ILL

- **Small-Loss Criterion:** ●
  Although ILL does not hard-filter the bottom-loss samples, its E-step posterior

$$P_t(y = \hat{y} \mid x) \propto P(\hat{y} \mid y)\, P(y \mid x; \theta_t)$$

assigns higher weight to low-loss (high-confidence) examples. Thus, "clean" samples naturally dominate the M-step updates, while "noisy" ones are down-weighted rather than discarded.

- **Curriculum via Dynamic Thresholding:** ✗
  There is no curriculum schedule that gradually admits easier samples. All data points are processed in parallel per EM iteration, with posteriors updated continuously.

- **Temporal Consistency of Predictions:** ✗
  ILL does not explicitly track or penalize prediction drift over time. Temporal consistency arises only implicitly through the alternation of E- and M-steps.

- **Noisy Validation Early-Stopping:** ✗
  ILL fits its objective end-to-end using EM optimization and does not require a noisy validation set for early stopping or hyperparameter tuning.

- **Normalized Loss Metrics:** ✗
  No loss normalization is performed. ILL relies on posterior probabilities to manage noise rather than comparing individual losses to dataset statistics.

- **Layer-Wise Memorization Rates:** ✗
  ILL treats the model holistically and does not analyze differential learning speeds across layers to detect noisy patterns.

- **Model Confidence on Labels:** ✓
  Implicitly used: the posterior $P_t(y \mid x, \hat{y})$ reflects the model's belief in each class, up-weighting high-confidence labels in the EM loss.

- **Soft Label Refinement:** ✓
  Core to ILL: noisy hard labels $\hat{y}$ are replaced by soft posteriors $P_t(y \mid x, \hat{y})$ in the surrogate loss, refining supervision over time.

- **Inter-Network Agreement:** ✗
  ILL is a single-network framework (plus a transition model if needed), and does not rely on inter-model agreement mechanisms.

- **Frequency-Domain Cues:** ✗
  ILL does not perform frequency-domain analysis; it operates entirely in the standard feature and label spaces.

- **Sparsity of Noise (Label Errors as Outliers):** ✗
  While the EM posterior de-emphasizes unlikely labels, ILL does not apply a sparsity constraint or assume that most errors are outliers.

- **Gradient Coherence:** ✗
  Gradients are derived from maximizing the expected complete-data log-likelihood. There is no explicit mechanism to encourage alignment between per-sample gradients.

- **Augmentation for Robustness: ✗**
  ILL itself is agnostic to augmentation. While base networks may use augmentation during training, ILL does not introduce any augmentation-specific loss or logic.
- **Ensembling for Robustness: ✗**
  ILL does not use any model ensembling or checkpoint averaging. It trains a single model using EM.
- **Consistency under Augmentation: ✗**
  ILL does not incorporate any loss that enforces prediction consistency under input augmentation, unlike FixMatch or similar SSL frameworks.

Summary. In summary, *Imprecise Label Learning (ILL)* handles noisy labels primarily through two mechanisms: model confidence and soft-label refinement via its EM-derived posterior. While these form a powerful foundation for probabilistic supervision, ILL does *not* incorporate most of the specialized empirical cues used in other robust training methods—such as loss normalization, memorization dynamics, or augmentation consistency. Instead, it offers a principled and unified treatment of label imprecision grounded in likelihood maximization.

6.2.6 L2B and L2B-C2D

Overview of L2B: Learning to Bootstrap Robust Models for Label Noise and L2B-C2D

The *Learning to Bootstrap (L2B)* [100] framework introduces a novel meta-learning-based approach to robust training under label noise by adaptively balancing the influence of noisy labels and model-generated pseudo-labels. Unlike traditional bootstrapping, which uses fixed reweighting, L2B learns *sample-wise* weights to dynamically control the supervision mix per instance during training.

Given a noisy training set $D_{\text{tra}} = \{(x_i, y_i)\}_{i=1}^{N}$ and optionally a clean validation set D_{val}, the per-sample training loss is defined as:

$$L_i = \alpha_i \cdot \ell(f(x_i), y_i^{\text{real}}) + \beta_i \cdot \ell(f(x_i), y_i^{\text{pseudo}})$$

where $\alpha_i, \beta_i \geq 0$ are learnable weights and y_i^{pseudo} is the model-generated pseudo-label. L2B does not require $\alpha_i + \beta_i = 1$, allowing it to emphasize more reliable supervision sources.

These weights are optimized via a bi-level meta-learning objective. The learner updates θ with the weighted loss, while the meta-objective evaluates generalization on a clean set:

$$\min_{\alpha,\beta \geq 0} \frac{1}{M} \sum_{j=1}^{M} \ell(f(x_j^v; \hat{\theta}), y_j^v), \quad \text{where} \quad \hat{\theta} = \theta - \lambda \nabla_\theta \sum_i L_i$$

As shown in Figure 6.8, this process ensures that reweighting parameters (α, β) are optimized to guide robust model updates.

Importantly, L2B can operate without a clean validation set by estimating clean samples online via Gaussian Mixture Modeling (GMM) on per-sample losses, making it widely applicable.

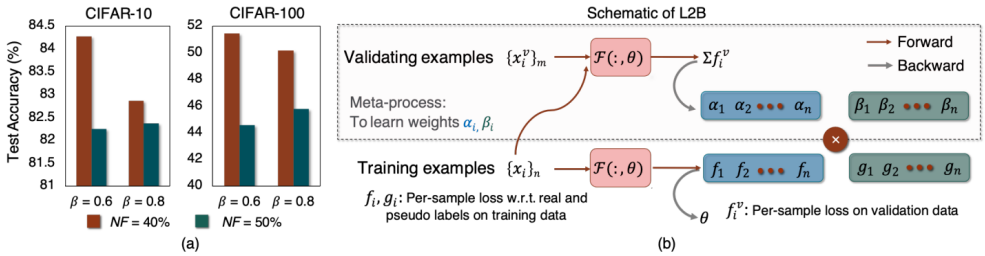


Figure 6.8. (a) The original bootstrapping loss is sensitive to the reweighting hyperparameter β, which varies with the noise level (NF = noise fraction). (b) Schematic overview of the L2B framework: reweighting parameters (α, β) are learned via a meta-process to balance noisy and pseudo labels dynamically. Adapted from [100].

L2B-C2D: Integrating L2B with Contrastive Pretraining

L2B-C2D enhances L2B by first applying self-supervised contrastive learning (*Contrast to Divide*, or C2D) to extract robust feature representations. These features allow early-stage division of training samples into clean and noisy subsets.

C2D contributes two core mechanisms:

- **Contrastive loss:** to improve representation quality and feature separability.
- **Semi-supervised consistency loss:** confident samples are used for supervised loss, while uncertain ones contribute to consistency-based objectives.

In this combined framework, L2B is applied after the C2D division step. It reweights losses for the now-refined data using its meta-learned α and β parameters. The synergy enables state-of-the-art results under both synthetic and real-world label noise—e.g., CIFAR-100 with 90% symmetric noise and Clothing1M.

The division of roles is clear: *C2D enhances features and sample partitioning, while L2B corrects supervision signals via meta-learned reweighting.*

Feature-Based Taxonomy of Robust Learning Methods for L2B and L2B-C2D

- **Architectural Structure**
  - **Using One Network:** ✓
    Used in both L2B and L2B-C2D. Both methods operate on a single main network throughout training. There is no co-teaching or ensemble setup.
  - **Using Two or Multiple Networks:** ✗
    Neither L2B nor L2B-C2D involves training multiple networks in parallel or alternating fashion.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    L2B does not implement any strategy to explicitly correct class imbalance in the loss function.
  - **KL Divergence Regularizer:** ✗
    There is no KL divergence penalty between distributions or outputs in L2B or C2D.
  - **Embedding Regularization:** ✓ (L2B-C2D only)
    C2D applies contrastive learning on feature representations during pretraining, which acts as an embedding regularizer.
  - **Adding Additional Learnable Parameters:** ✓
    L2B introduces sample-specific weights α_i, β_i, which are dynamically learned—an implicit form of additional trainable parameters.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✓
    Used in both L2B and L2B-C2D. L2B modifies the loss function directly through adaptive reweighting of real and pseudo-label contributions; this is a core feature.

  - **Add Contrastive Loss:** ✓ (L2B-C2D only)
    C2D uses contrastive loss during the pretraining phase to improve feature discrimination and noise separation.

- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ●
    Partially used in both L2B and L2B-C2D. L2B can optionally use a small clean validation set for meta-updates, but also supports online estimation of clean samples via a Gaussian Mixture Model (GMM).
  - **Add Semi-Supervised Loss:** ✓ (L2B-C2D only)
    C2D segments data into clean and noisy subsets, treating them with labeled and unlabeled losses—a semi-supervised training setup.
  - **Meta-Learning Framework:** ✓
    Used in both L2B and L2B-C2D. Meta-learning is central to L2B: it learns sample-specific weights via a bi-level optimization process that improves generalization

Empirical Cues in Noisy-Label Learning for L2B and L2B-C2D

- **Small-Loss Criterion:** ✓
  L2B relies on model prediction quality (pseudo-labels), which are more reliable for low-loss (clean) samples. Also, the GMM used for estimating online meta-sets operates over loss values, exploiting memorization dynamics.
- **Curriculum via Dynamic Thresholding:** ✗
  L2B updates weights on all samples continuously without scheduling or thresholds; there's no progressive curriculum based on difficulty.
- **Temporal Consistency of Predictions:** ✗
  L2B does not explicitly track how predictions evolve over time or enforce consistency across epochs.
- **Noisy Validation Early-Stopping:** ✗
  L2B doesn't use early stopping based on a noisy validation set. It uses a meta-objective to guide training, not to stop it.
- **Normalized Loss Metrics:** ✓
  The online meta-set creation (via GMM) depends on normalized or comparative per-sample losses to distinguish clean vs. noisy samples.
- **Layer-Wise Memorization Rates:** ✗
  L2B does not analyze or leverage layer-specific learning rates or representations.
- **Model Confidence on Labels:** ✓
  The confidence (from model outputs) is implicitly used when generating pseudo-labels and deciding how much to trust them (via higher β values).
- **Soft Label Refinement:** ✓
  L2B replaces noisy labels with weighted combinations of true and pseudo-labels — effectively refining labels into soft targets dynamically.

- **Inter-Network Agreement:** ✗
  Neither L2B nor C2D employs co-training or agreement between multiple models.
- **Frequency-Domain Cues:** ●
  Not explicitly used. Contrastive learning in C2D promotes some similar robustness properties, but no actual frequency-domain transformation is applied.
- **Sparsity of Noise (Label Errors as Outliers):** ✓ (L2B-C2D only)
  C2D assumes that noisy labels can be separated based on feature embedding clustering — treating noise as outliers relative to the clean manifold.
- **Gradient Coherence:** ✗
  L2B does not measure or leverage the coherence of gradients between samples.
- **Augmentation for Robustness:** ✓ (L2B-C2D only)
  C2D uses data augmentations during contrastive learning and training — a core part of contrastive pretraining.
- **Ensembling for Robustness:** ✗
  No explicit ensemble (like SWA or model averaging) is used in either method.
- **Consistency under Augmentation:** ✓ (L2B-C2D only)
  C2D encourages prediction consistency across augmented views during its pretraining and separation steps.

6.2.7 NCOD and NCOD$^+$

Overview of NCOD and NCOD$^+$

NCOD (Noisy Centroid Outlier Discounting) [80] and its enhanced version NCOD$^+$ are robust training methods designed to mitigate the impact of label noise by integrating two key ideas: learnable sample weights and centroid-based feature similarity. The model introduces a dual weighting mechanism to identify and discount noisy samples during training.

Key Components

(a) Learnable Noise Indicator Weight. Each training sample x_i is assigned a learnable weight $\lambda_i \in [0, 1]$ that reflects the likelihood of the sample being clean. These weights are optimized through gradient descent alongside the model parameters.

(b) Centroid Similarity Score. In addition to λ_i, NCOD computes the cosine similarity between the normalized feature representation $f(x_i)$ and the class centroid c_{y_i} to determine whether a sample aligns with its labeled class. The class centroid is computed in the normalized feature space.

The final supervision weight used in loss computation is the product of the learnable indicator and the centroid similarity:

$$w_i = \lambda_i \cdot \cos\left(f(x_i), c_{y_i}\right)$$

This mechanism ensures that samples which are both aligned in feature space and historically consistent receive more trust during training.

As shown in Figure 6.9, samples that are well clustered around their class centroid are more likely to be clean. The figure illustrates separability between clean and noisy instances in the embedding space, justifying the use of centroid-based weighting.

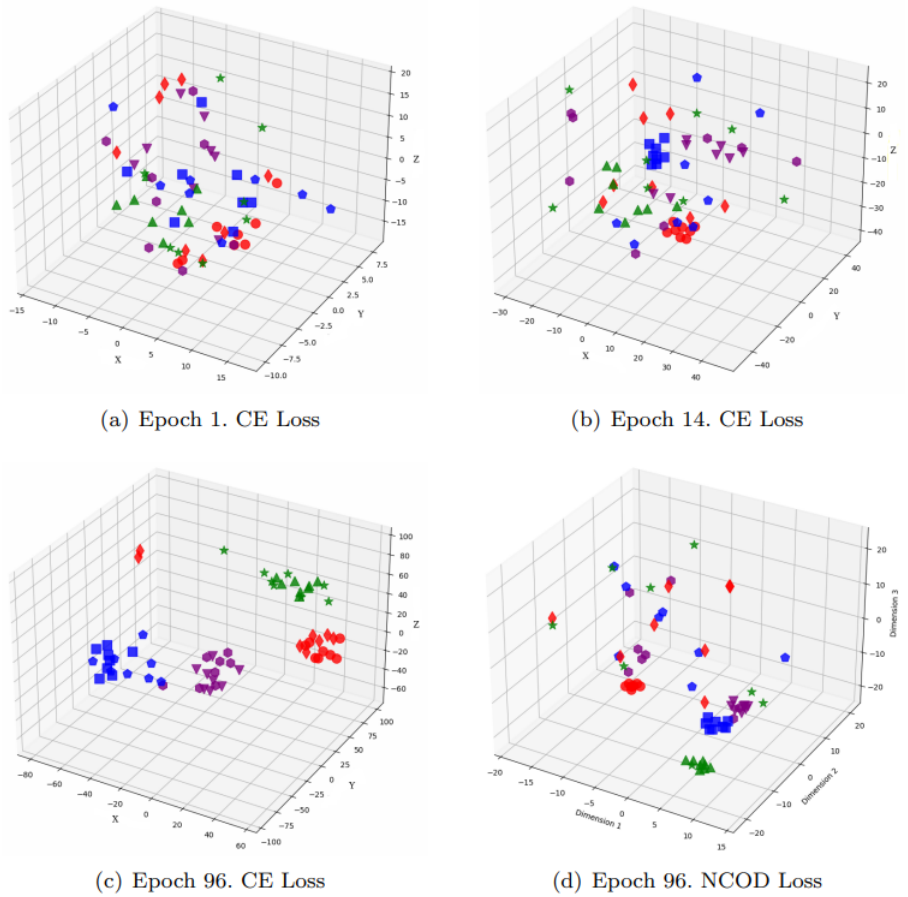
(c) Weighted Cross-Entropy Loss. The primary training loss integrates these sample-specific weights into a reweighted cross-entropy formulation:

$$\mathcal{L}_{\text{NCOD}} = \sum_{i=1}^{n} w_i \cdot \text{CE}(p_i, y_i)$$

where p_i is the predicted softmax probability and y_i is the (possibly noisy) label.

(d) Auxiliary Losses. To improve training stability and prevent degenerate solutions, NCOD introduces three auxiliary loss components:

- **Balance Loss** $\mathcal{L}_{\text{balance}}$: Encourages uniformity in the class prediction distribution by minimizing the KL divergence between the empirical class distribution $\bar{p}$ and the uniform prior U.
- **Prior Loss** $\mathcal{L}_{\text{prior}}$: Regularizes the average of λ_i values across the dataset to approximate the known or estimated clean label ratio.



(a) Epoch 1. CE Loss

(b) Epoch 14. CE Loss

(c) Epoch 96. CE Loss

(d) Epoch 96. NCOD Loss

Figure 6.9. Sample embeddings of four classes from CIFAR-100 with 20% symmetrical noise. Colors represent classes, and shapes distinguish noisy and clean samples: Blue (square: clean, pentagon: noisy), Red (circle: clean, diamond: noisy), Green (triangle-up: clean, star: noisy), and Purple (triangle-down: clean, hexagon: noisy). (Adapted from "Learning with Noisy Labels through Learnable Weighting and Centroid Similarity"[80]

- **Entropy Loss** $\mathcal{L}_{\text{entropy}}$: Penalizes overconfident predictions by maximizing entropy over softmax outputs.

The total training objective is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{NCOD}} + \alpha \cdot \mathcal{L}_{\text{balance}} + \beta \cdot \mathcal{L}_{\text{prior}} + \gamma \cdot \mathcal{L}_{\text{entropy}}$$

NCOD$^+$: Augmentation Consistency

NCOD$^+$ [80] extends NCOD by introducing a pseudo-label filtering strategy based on augmentation consistency. It adds a constraint such that pseudo-labels are only retained if the predicted class remains stable under strong data augmentations (e.g., RandAugment). This stability check ensures that only structurally consistent examples are used during training.

This extension strengthens NCOD's robustness by filtering out examples whose semantic consistency is questionable under transformations—a known indicator of

label noise.

Feature-Based Taxonomy of Robust Learning Methods for NCOD/NCOD$^+$

- **Architectural Structure**
  - **Using One Network:** ✓
    NCOD is a single-model approach—there's only one classifier whose penultimate-layer embeddings are used for centroid computation and whose parameters θ are optimized together with the per-sample weight u.
  - **Using Two or Multiple Networks:** ✗
    No co-teaching or dual-network scheme is used; all robustification comes from the learnable u and soft labels within the same network.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✓ (NCOD+)
    In NCOD+, a class-balance term is added to prevent skewed prediction distributions.
  - **KL Divergence Regularizer:** ✓ (NCOD+)
    NCOD+ uses a Kullback–Leibler consistency loss between predictions on original and augmented inputs.
  - **Embedding Regularization:** ✓
    The method directly regularizes via soft labels computed from distances in the latent space, encouraging samples close to their class centroid to dominate the loss.
  - **Adding Additional Learnable Parameters:** ✓
    Introduces one learnable parameter u_i per sample, which adaptively discounts presumed noisy points during training.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✗
    While NCOD modifies the loss, it also alters training dynamics via the learnable u parameter (not just a fixed robust loss).
  - **Add Contrastive Loss:** ✗
    No contrastive or metric-learning objective is used; robustness comes from centroid similarity and u.
- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    There is no small trusted set; everything is learned on the noisy training data alone.
  - **Add Semi-Supervised Loss:** ✗
    Although NCOD+ uses data augmentation consistency, it does not incorporate unlabeled data or a semi-supervised branch.

  - **Meta-Learning Framework:** ✗
    The per-sample weight u is learned by simple gradient steps on an L_2-style loss, not via any bi-level/meta optimization.

Empirical Cues in Noisy-Label Learning for NCOD/NCOD$^+$

- **Small-Loss Criterion:** ✓
  NCOD/NCOD$^+$ implicitly leverage the small-loss phenomenon by learning per-sample weights u that remain small for clean samples (consistently correct predictions), which are typically low-loss. While there is no hard selection, the model prioritizes these samples in a soft, learnable way.

- **Curriculum via Dynamic Thresholding:** ●
  This cue is *implicitly* used in both NCOD and NCOD+. While there is no explicit curriculum or hard threshold, the continuous update of u serves as a soft curriculum mechanism: clean samples get low u, while noisy ones are downweighted over time. No fixed schedule or difficulty ranking is enforced.

- **Temporal Consistency of Predictions:** ✓
  Used in both NCOD and NCOD+. The update rule for u depends on prediction stability—if a sample continues to be correctly predicted, its u remains low. This directly captures the notion of temporal consistency.

- **Noisy Validation Early-Stopping:** ✗
  Although the paper discusses early learning, they do not use noisy-data peak-accuracy stopping; instead, they counteract overfitting via u.

- **Normalized Loss Metrics:** ✗
  There's no per-sample loss normalization or comparison to class averages beyond the centroid similarity.

- **Layer-Wise Memorization Rates:** ✗
  No layer-specific learning rates or cues are used.

- **Model Confidence on Labels:** ✓
  The soft label $\tilde{y}_i = \max(\text{sim}(h_i, \bar{h}_{c_i}), 0)$ and the cross-entropy shift by u act like a confidence weight—samples far from their centroid (low similarity) or correctly predicted (low loss) get less influence.

- **Soft Label Refinement:** ✓
  Used in both methods. NCOD+ builds soft labels from centroid similarity in feature space, and these are updated continuously over training. This represents a structural and probabilistic refinement of noisy labels.

- **Inter-Network Agreement:** ✗
  Not used. Both NCOD and NCOD+ rely on a single network and do not compare predictions across different models or subnetworks.

- **Frequency-Domain Cues:** ✗
  There is no decomposition in the Fourier domain to separate stable vs. noisy components.
- **Sparsity of Noise (Label Errors as Outliers):** ✓
  The learnable $u_1, \ldots, u_n$ are initialized near zero and grow only for samples the model mispredicts—treating noisy labels as sparse outliers whose loss impact is discounted.
- **Gradient Coherence:** ●
  Partially and implicitly used. While the models do not explicitly compute gradient alignment, the evolution of u indirectly reflects gradient instability—samples that are repeatedly misclassified likely generate inconsistent gradients and are progressively downweighted.
- **Augmentation for Robustness:** ✓ (NCOD+)
  NCOD+ explicitly uses standard data augmentations plus an unsupervised data-augmentation consistency term.
- **Ensembling for Robustness:** ✗
  No model ensembling is employed; all experiments use single models (or internally consistent views in NCOD+).
- **Consistency under Augmentation:** ✓ (NCOD+ only)
  NCOD+ enforces consistency between original and augmented versions of the input using a KL divergence regularizer. This encourages the model to make stable predictions under perturbations and improves robustness to label noise.

Summary: NCOD achieves robustness primarily via embedding-based soft labels and a learnable outlier discounting parameter u, all within a single network and without any clean hold-out. NCOD+ augments this with standard consistency (KL) and class-balance regularization, plus data augmentation, to further stabilize training under noise.

6.2.8 NES

Overview of NES: Noisy Early Stopping

Training deep neural networks with noisy labels poses a significant risk of overfitting, especially in the absence of clean validation data. To address this challenge, Toner and Storkey propose **Noisy Early Stopping (NES)** [76]—a simple and effective method that enables early stopping *without* requiring a clean validation set. Instead, NES monitors the validation accuracy on a held-out subset of the noisy training data and halts training when this noisy accuracy begins to decline.

The key insight is that, under many practical conditions—particularly when label noise is class-preserving—the noisy validation accuracy can act as a reliable proxy for clean generalization performance. This allows NES to function analogously to traditional early stopping, but without the need for costly clean validation labels.

Formally, let f be a model trained using a Fisher-consistent loss function L, such as cross-entropy. Given a noisy dataset $\tilde{D}$ split into a training set D_{train} and a validation set D_{val}, NES operates as follows:

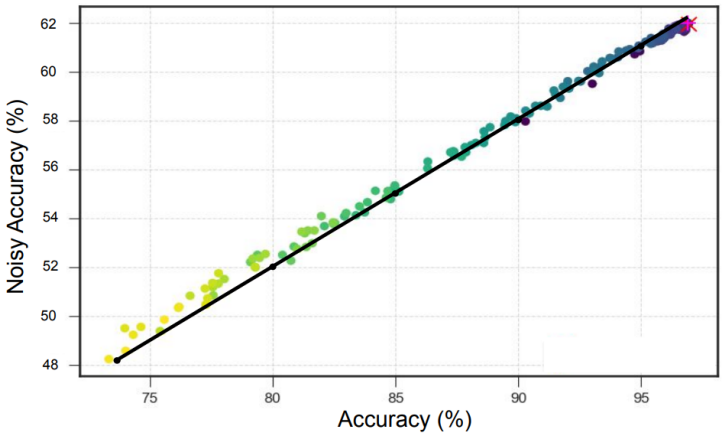


Figure 6.10. Relationship between clean and noisy accuracy under symmetric label noise. A classifier trained on MNIST with 36% symmetric noise shows a nearly linear correlation between noisy and clean accuracy across epochs. Theoretical prediction is illustrated by the black line, closely matching the observed trend. Early and late training epochs are color-coded from blue to yellow. Adapted from Noisy Early Stopping (NES) [76]

1. At each epoch t, compute the noisy validation accuracy $\tilde{\text{acc}}(t)$ on D_{val}.

2. Maintain a patience counter p and track the best observed validation accuracy $\tilde{\text{acc}}^*$.

3. If $\tilde{\text{acc}}(t) > \tilde{\text{acc}}^*$, update the best model and reset $p = 0$.

4. Else, increment p. If $p \geq P$ (patience threshold), stop training and return the best saved model.

Theoretical Insight. Under symmetric label noise, where each label has an equal probability η of being flipped to any other class, the relationship between the noisy risk $R_\eta(q)$ and the clean risk $R(q)$ of a classifier q becomes affine:

$$R_\eta(q) = \left(1 - \frac{c\eta}{c-1}\right) R(q) + \eta,$$

where c is the number of classes. This equation implies that minimizing the noisy risk closely tracks the clean risk when the noise is symmetric and class-preserving. As illustrated in Figure 6.10, experiments on MNIST with 36% symmetric noise confirm this relationship: the noisy and clean accuracies are almost perfectly linearly correlated throughout training.

Empirical Evaluation. NES was tested on MNIST, FashionMNIST, CIFAR-10, and CIFAR-100 under various noise settings (symmetric, asymmetric, class-dependent) and loss functions (CE, MSE, GCE, etc.). Results show that NES performs comparably to clean early stopping in 93% of cases, and consistently outperforms training without early stopping. NES is therefore a highly practical and low-overhead regularization strategy for learning under noisy labels.

Feature-Based Taxonomy of Robust Learning Methods for NES

- **Architectural Structure**
  - **Using One Network:** ✓
    NES uses a single standard neural network without architectural duplication.
  - **Using Two or Multiple Networks:** ✗
    NES does not require co-training, ensemble models, or dual networks.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    NES does not modify the loss function to correct for class imbalance. It is agnostic to label distribution.
  - **KL Divergence Regularizer:** ✗
    NES does not apply any KL divergence constraint to align distributions or enforce prediction consistency.
  - **Embedding Regularization:** ✗
    NES does not directly regularize or constrain the representation space (embeddings).
  - **Adding Additional Learnable Parameters:** ✗
    NES does not introduce any extra modules or trainable components. The architecture is unmodified.
- **Learning Dynamics**

  - **Loss Regularization Only:** ✗
    NES does not modify the loss function (e.g., robust loss, weighted loss). It regularizes training time, not the loss.
  - **Add Contrastive Loss:** ✗
    NES does not use any contrastive learning objective to separate clean and noisy examples.

- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    This is a key advantage of NES: it works without any clean validation data, relying on noisy validation accuracy.
  - **Add Semi-Supervised Loss:** ✗
    NES does not include pseudo-labeling or consistency regularization between labeled and unlabeled data.
  - **Meta-Learning Framework:** ✗
    NES is not meta-learned; it does not dynamically adapt weighting or structure through meta-objectives.

Empirical Cues in Noisy-Label Learning for NES

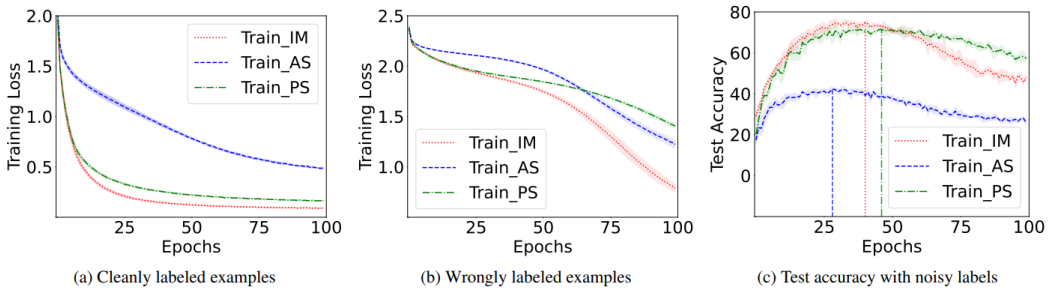
- **Small-Loss Criterion:** ●
  NES does not explicitly prioritize low-loss samples, but implicitly leverages the early-learning phase that arises from the memorization effect — the same phenomenon that underlies the small-loss cue.
- **Curriculum via Dynamic Thresholding:** ✗
  There is no gradual inclusion of samples or curriculum scheduling. NES is epoch-based stopping, not sample selection.
- **Temporal Consistency of Predictions:** ✗
  NES does not track prediction stability over time for individual samples. It focuses only on validation accuracy trends.
- **Noisy Validation Early-Stopping:** ✓
  This is the core principle of NES. It explicitly monitors noisy validation accuracy and stops training when it stops improving.
- **Normalized Loss Metrics:** ✗
  NES does not compute normalized per-sample losses (e.g., loss minus class mean) to detect noise.
- **Layer-Wise Memorization Rates:** ✗
  NES does not analyze or exploit different learning dynamics of network layers (shallow vs. deep).
- **Model Confidence on Labels:** ✗
  NES does not assess prediction confidence as a signal for noise. Only validation accuracy is used.

- **Soft Label Refinement:** ✗
  NES does not replace noisy hard labels with soft targets; it relies on the original noisy labels as-is.
- **Inter-Network Agreement:** ✗
  NES uses only one network, and no agreement or co-training mechanism is involved.
- **Frequency-Domain Cues:** ✗
  NES does not use frequency-based robustness heuristics (e.g., phase/amplitude separation).
- **Sparsity of Noise (Label Errors as Outliers):** ✗
  NES does not assume label noise is sparse or apply sparsity constraints.
- **Gradient Coherence:** ✗
  NES does not analyze gradients to identify noise; it does not distinguish between coherent/incoherent examples.
- **Augmentation for Robustness:** ✗
  NES does not depend on data augmentation like Mixup, Cutout, or RandAugment in its core mechanism.
- **Ensembling for Robustness:** ✗
  NES does not use model averaging. It selects a single best model via early stopping.
- **Consistency under Augmentation:** ✗
  NES does not evaluate how predictions change under input augmentations.

6.2.9 PADDLES

PADDLES: Phase-Amplitude Spectrum Disentangled Early Stopping

PADDLES (Phase-AmplituDe DisentangLed Early Stopping) [**?**] is a robust training strategy designed to mitigate the adverse effects of noisy labels in deep neural networks by operating in the frequency domain. The method is based on the insight that phase spectrum (PS) carries more semantic information, whereas the amplitude spectrum (AS) tends to overfit to label noise. This is motivated by both biological vision systems and empirical studies showing that CNNs overfit more rapidly to AS under noisy supervision, while PS remains more robust (as illustrated in Figure 6.11).



(a) Cleanly labeled examples (b) Wrongly labeled examples (c) Test accuracy with noisy labels

Figure 6.11. Training dynamics of ResNet-18 on CIFAR-10 using raw images, amplitude spectrum (AS), and phase spectrum (PS), under clean and noisy labels. The model overfits more rapidly to AS, especially with noisy labels (Figure 1b), while PS exhibits more robust behavior. Dotted lines indicate peak performance epochs for each input type. Adapted from: Phase-AmplituDe DisentangLed Early Stopping) [25]

The central idea of PADDLES is to disentangle deep features into AS and PS, then stop their training at different points to prevent memorization of noise. This disentanglement is performed via the Discrete Fourier Transform (DFT) applied to the feature maps χ at a selected intermediate layer j:

$$F_\chi(u) = \sum_{p=0}^{M-1} \chi_p \cdot e^{-i \cdot \frac{2\pi}{M} pu}$$

From the complex-valued DFT output $F_\chi(u)$, the amplitude and phase components are computed as:

$$AS_\chi(u) = |F_\chi(u)|, \quad PS_\chi(u) = \arctan\left(\frac{\text{Imag}(F_\chi(u))}{\text{Re}(F_\chi(u))}\right)$$

These components are recombined using the inverse DFT to reconstruct the original feature map χ':

$$\chi'_p = \frac{1}{M} \sum_{u=0}^{M-1} \left(e^{i \cdot PS_\chi(u)} \cdot AS_\chi(u)\right) \cdot e^{i \cdot \frac{2\pi}{M} pu}$$

To control the optimization, PADDLES detaches the gradients of AS and PS components from the computational graph at specific times: AS is stopped first,

followed by PS. This scheduling ensures the network focuses on learning robust semantic structures without overfitting to noisy textures. The overall training flow is depicted in Figure 6.12, which shows forward and backward passes with gradient detachment points.

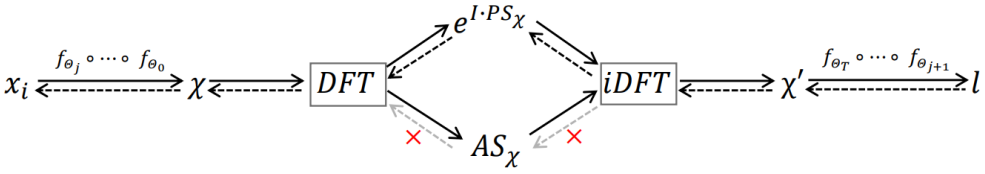


Figure 6.12. Illustration of the PADDLES strategy, where the amplitude spectrum is selectively detached from training. Forward propagation is indicated by arrows, and backward propagation by gradient paths. The model disentangles deep features into amplitude and phase components, allowing independent early stopping for each to control learning dynamics. Adapted from: Phase-AmplituDe DisentangLed Early Stopping) [25]

PADDLES is implemented as a three-phase training strategy:

- Train the network normally (AS + PS) for T_A epochs.
- Detach AS and train with PS only for T_P epochs.
- Detach PS as well and stop frequency-specific learning.

After this, Progressive Early Stopping (PES) is applied: lower layers (up to layer j) are frozen, and upper layers are retrained progressively to refine high-level features without reintroducing noise. The loss function used throughout remains standard cross-entropy, and no architectural changes are required, making the method modular and easily adaptable.

Empirical results on CIFAR-10, CIFAR-100, CIFAR-N, and Clothing-1M datasets demonstrate that PADDLES achieves state-of-the-art robustness, outperforming existing methods like PES, ELR+, and DivideMix across various types and levels of label noise. Particularly strong performance is observed under instance-dependent and real-world noisy conditions, with significant gains attributed to the semantic robustness of PS and the training schedule design.

Feature-Based Taxonomy of Robust Learning Methods for PADDLES

- **Architectural Structure**
  - **Using One Network:** ✓
    PADDLES operates with a single CNN model (e.g., ResNet), without needing dual networks or ensembling.
    All training phases (AS/PS detachment, PES) are applied within one model's architecture.
  - **Using Two or Multiple Networks:** ✗
    PADDLES does not involve any co-training or multiple networks like Co-teaching or DivideMix.

- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    PADDLES uses standard cross-entropy loss. There's no mechanism to reweight loss based on class distribution.
  - **KL Divergence Regularizer:** ✗
    No KL divergence term is introduced between predictions or target distributions.
  - **Embedding Regularization:** ●
    PADDLES operates on intermediate feature maps (deep features) by disentangling their frequency components.
    Why: While not a regularizer in the usual sense, the gradient detachment acts as a form of implicit regularization on embeddings.
  - **Adding Additional Learnable Parameters:** ✗
    PADDLES does not introduce any new layers, parameters, or modules. All operations (DFT, iDFT) are non-parametric.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✗
    The method does not modify the loss function itself. Regularization is applied by controlling training dynamics via frequency filtering and early stopping.
  - **Add Contrastive Loss:** ✗
    No contrastive or self-supervised loss is added to PADDLES.
- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    PADDLES is fully self-supervised with respect to label noise. No clean data or validation labels are required.
  - **Add Semi-Supervised Loss:** ●
    The core PADDLES method is supervised-only. However, it is explicitly combined with MixMatch during semi-supervised evaluation (as described in Section 3.3 and Table 2 of the paper). While the semi-supervised loss is not an internal component of PADDLES, the method is designed to support and benefit from it, which justifies partial use.
  - **Meta-Learning Framework:** ✗
    PADDLES does not employ any meta-learned weights or optimization of loss weights. It's a hand-designed training schedule based on frequency domain behavior.

Empirical Cues in Noisy-Label Learning for PADDLES

- **Small-Loss Criterion:** ●
  PADDLES does not use per-sample small-loss selection, but its training schedule is based on the memorization dynamics of CNNs — specifically, that AS

memorizes noise faster than PS. Early stopping of AS is a form of implicit regularization guided by the memorization effect.

- **Curriculum via Dynamic Thresholding:** ✗
  There is no curriculum or adaptive thresholding in PADDLES. Training does not gradually include harder samples; the transition happens via fixed epoch schedules for AS and PS.

- **Temporal Consistency of Predictions:** ✗
  PADDLES does not monitor prediction consistency over time or penalize changes in class probabilities. Its focus is on the feature space, not prediction dynamics.

- **Noisy Validation Early-Stopping:** ✗
  The model does not use a noisy validation set for stopping. Instead, it performs manual early-stopping per component (AS/PS) without relying on validation accuracy trends.

- **Normalized Loss Metrics:** ✗
  No loss normalization (e.g., subtracting class-average loss) is used. PADDLES operates in the frequency domain, not on loss statistics.

- **Layer-Wise Memorization Rates:** ✓
  PADDLES exploits the fact that deeper layers memorize noise more easily. It applies frequency disentanglement at a specific intermediate layer (j) and then freezes early layers during PES, leveraging their generalization ability.

- **Model Confidence on Labels:** ✗
  The model doesn't use confidence (e.g., softmax scores) to estimate label correctness or filter samples.

- **Soft Label Refinement:** ✗
  No soft labels or label refinement strategies are applied. Labels remain unchanged; robustness comes from training dynamics, not label updates.

- **Inter-Network Agreement:** ✗
  PADDLES trains a single model; there's no cross-network agreement or co-training.

- **Frequency-Domain Cues:** ✓
  This is the core principle of PADDLES. The method explicitly disentangles AS (noise-prone) and PS (semantic) components and controls their optimization to improve noise robustness.

- **Sparsity of Noise (Label Errors as Outliers):** ✗
  PADDLES does not model label noise sparsity or apply sparse regularization. It treats label noise as a general phenomenon affecting the training dynamics.

- **Gradient Coherence:** ✗
  PADDLES does not evaluate gradient directions or coherence to separate clean vs. noisy samples.

- **Augmentation for Robustness:** 🔵
  Partially Used (in evaluation only) — Used in semi-supervised evaluation with MixMatch and Mixup, but not part of the core PADDLES method. In the base supervised version, only simple augmentations like cropping and flipping are applied.
- **Ensembling for Robustness:** ✗
  No ensemble of models or predictions is used. PADDLES uses a single model with early stopping per component.
- **Consistency under Augmentation:** ✗
  The base method does not enforce prediction consistency across augmented inputs.

6.2.10 PES

Overview of PES: Progressive Early Stopping

Progressive Early Stopping (PES) [5] is a robust training method designed to counteract the detrimental effect of noisy labels in deep neural networks (DNNs). PES builds upon the memorization effect observed in DNNs—where networks tend to learn clean patterns in early epochs before overfitting to noisy labels—by introducing a layer-wise early stopping strategy rather than stopping the entire network simultaneously.

Method Overview

Let the training dataset be composed of noisy samples $\{(x_i, \tilde{y}_i)\}_{i=1}^n$, drawn from a corrupted distribution $\tilde{D}$, where $\tilde{y}_i$ is a noisy label. The model $f(x; \Theta)$ is a deep network with parameters Θ, decomposed into L sequential parts:

$$z_1 = f_1(x; \Theta_1), \quad z_l = f_l(z_{l-1}; \Theta_l), \quad l = 2, \ldots, L.$$

The full network is thus represented as $f(x; \Theta) = f(x; \Theta_1, \ldots, \Theta_L)$, and its output is z_L.

PES begins by training the entire network for a small number of epochs T_1, with the goal of fitting general, clean patterns:

$$\min_{\Theta_1, \ldots, \Theta_L} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i; \Theta), \tilde{y}_i),$$

where $\mathcal{L}$ is the cross-entropy loss. After this phase, the first block's parameters Θ_1^* are fixed. Then, for each subsequent part $l = 2, \ldots, L$, PES reinitializes $\Theta_l, \ldots, \Theta_L$ and trains only the remaining parts, progressively reducing the number of training epochs $T_2, \ldots, T_L$:

$$\min_{\Theta_l, \ldots, \Theta_L} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i; \Theta_1^*, \ldots, \Theta_{l-1}^*, \Theta_l, \ldots, \Theta_L), \tilde{y}_i).$$

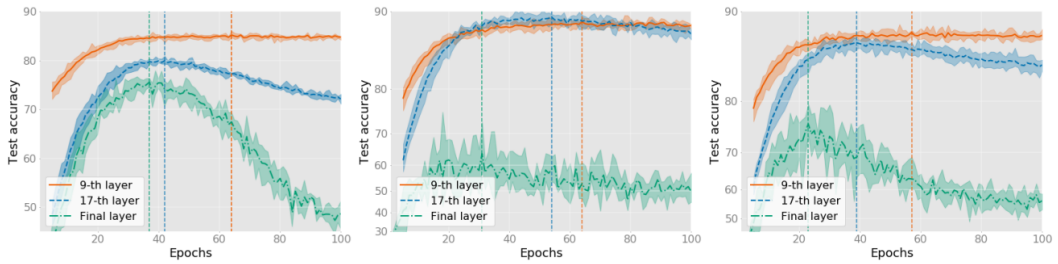
As shown in Figure 6.13, representations extracted from deeper layers degrade more quickly under noisy supervision, supporting the intuition that later layers should be trained for fewer epochs.

Confident Example Selection

After training, PES uses the final model to extract confident examples—data points where the model's prediction matches the noisy label:

$$D_l = \{(x_i, \tilde{y}_i) \mid \tilde{y}_i = \hat{y}_i\}, \quad \hat{y}_i = \arg\max_k \frac{1}{2} \left[f_k(\mathrm{Aug}(x_i)) + f_k(\mathrm{Aug}(x_i)) \right],$$

where $\mathrm{Aug}(\cdot)$ denotes stochastic data augmentation (e.g., random flip and crop). The average prediction over two augmentations provides a more stable pseudo-label.



(a) Symmetric 50% (b) Pairflip 45% (c) Instance 40%

Figure 6.13. We train a ResNet-18 model on CIFAR-10 with three types of noisy labels and evaluate the impact of noisy labels on the representations from the 9-th layer, the 17-th layer, and the final layer. The X-axis is the number of epochs for the first block of the network. The curves present the mean of five runs and the best performances are indicated with dotted vertical lines. Adapted from Progressive Early Stopping (PES) [5]

To mitigate class imbalance in the confident set, a weighted classification loss is applied:

$$\mathcal{L}_c = \sum_{i=1}^{N} w_{\tilde{y}_i} \cdot \mathcal{L}_p(\tilde{y}_i, f(x_i; \Theta)), \quad w_k = \frac{\sigma_k}{\sum_{j=1}^{K} \sigma_j},$$

where σ_k is the count of confident examples with label k.

Semi-Supervised Refinement

To further leverage noisy data, PES combines confident examples (as labeled) with the remaining data (as unlabeled) and applies a semi-supervised learning framework, such as MixMatch. This improves generalization by using all available data without requiring clean supervision.

Summary

PES provides a simple yet powerful extension to early stopping, improving robustness to label noise by controlling overfitting at the layer level. It requires no clean labels, adapts to multiple architectures, and performs well under heavy and complex noise settings. Its benefits are clearly reflected in test accuracy, label precision, and recall, as reported in Table 1, and its performance across varying noise levels demonstrates strong robustness with minimal computational overhead.

Feature-Based Taxonomy of Robust Learning Methods for PES

- **Architectural Structure**
  - **Using One Network:** ✓
    PES operates on a single deep neural network, which is progressively trained by splitting it into parts. There is no duplication or ensemble of networks during training (though ensemble results are reported separately).

  - **Using Two or Multiple Networks:** ✗
    PES does not involve multiple models (unlike Co-teaching or DivideMix). It works with just one network trained in a staged manner.

- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✓
    During confident example learning, PES reweights the loss based on class frequencies to address imbalance in confident samples:

$$w_k = \frac{\sigma_k}{\sum_j \sigma_j}$$

    where σ_k is the number of confident examples in class k.
  - **KL Divergence Regularizer:** ✗
    PES does not include any KL-divergence-based penalties to align prediction distributions across time or networks.
  - **Embedding Regularization:** ✗
    There is no direct regularization applied to feature embeddings or intermediate representations.
  - **Adding Additional Learnable Parameters:** ✗
    PES does not introduce any extra modules, such as noise layers, adapters, or new trainable structures. It reuses the base architecture fully.

- **Learning Dynamics**
  - **Loss Regularization Only:** ✗
    While PES does regularize learning, it does not achieve this by modifying the loss function itself. The regularization effect arises from training dynamics (layer-wise early stopping), not from altering loss terms.
  - **Add Contrastive Loss:** ✗
    PES does not use contrastive learning or objectives that compare sample pairs or clusters.

- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    PES does not require any clean validation data or access to ground-truth labels for early stopping. All selection is based on model behavior.
  - **Add Semi-Supervised Loss:** ✓ (PES + Semi-supervised (extended version))
    PES integrates semi-supervised learning (MixMatch) after the base model training. It treats confident examples as labeled and the remaining as unlabeled, allowing the use of all data in a semi-supervised framework for further refinement.
  - **Meta-Learning Framework:** ✗
    PES does not perform meta-learning or involve learned sample weighting strategies.

Empirical Cues in Noisy-Label Learning for PES

- **Small-Loss Criterion:** ●
  PES does not select samples based on low loss, but its core design exploits the memorization effect, which underlies the small-loss criterion. By stopping deeper layers early, PES avoids memorizing noisy labels and retains generalization learned from clean patterns.

- **Curriculum via Dynamic Thresholding:** ✗
  PES does not use dynamic thresholds to gradually include harder samples. All data is used from the beginning; the curriculum is implemented through layer-wise training, not sample selection.

- **Temporal Consistency of Predictions:** ✗
  PES does not monitor prediction consistency over time or penalize temporal prediction shifts.

- **Noisy Validation Early-Stopping:** ✗
  PES does not rely on a validation set (clean or noisy) to determine when to stop training. Instead, it uses fixed epoch schedules for each part of the network.

- **Normalized Loss Metrics:** ✗
  No form of loss normalization (e.g., subtracting class averages) is employed to detect noisy samples.

- **Layer-Wise Memorization Rates:** ✓
  PES is explicitly built on this cue. It observes that deeper layers overfit to noise faster, so it stops training later layers earlier to reduce noise impact.

- **Model Confidence on Labels:** ✓
  After initial training, PES identifies confident examples where predictions match the noisy label. This model confidence is used for semi-supervised learning.

- **Soft Label Refinement:** ✗
  PES does not generate soft labels or distributions to replace noisy hard labels. Only confident hard labels are used.

- **Inter-Network Agreement:** ✗
  PES uses a single network and does not rely on agreement between multiple models or subnetworks.

- **Frequency-Domain Cues:** ✗
  PES operates in the time/feature domain, with no explicit decomposition into frequency components.

- **Sparsity of Noise (Label Errors as Outliers):** ✗
  PES does not assume label noise is sparse or apply any sparse regularization techniques.

- **Gradient Coherence:** ✗
  PES does not analyze or exploit the coherence or divergence of gradients to distinguish clean from noisy samples.
- **Augmentation for Robustness:** ✓
  PES applies standard data augmentations (e.g., crop, flip) during confident example selection and semi-supervised training to improve robustness.
- **Ensembling for Robustness:** ✓ (optional)
  The paper also reports results using PES + model ensembling, where predictions from multiple PES-trained networks are averaged. However, this is an optional enhancement.
- **Consistency under Augmentation:** ✓
  To extract confident examples, PES checks whether predictions remain consistent across two augmented views of the same input. This is central to the confidence estimation step.

6.2.11 PGDF

Overview of PGDF: Prior Guided Denoising Framework

The Prior Guided Denoising Framework (PGDF) [10] addresses the challenge of learning with noisy labels by introducing a method that preserves informative hard samples while suppressing noisy ones. Unlike conventional approaches that rely solely on the small-loss criterion to filter clean samples, PGDF leverages training history to better distinguish between clean-hard and truly noisy data.

Method Overview

PGDF is a two-stage framework designed to enhance robustness against label noise:

- Prior Guided Sample Dividing
- Denoising with the Divided Sets

Stage 1: Prior Guided Sample Dividing

In the first stage, PGDF classifies training samples into three sets:

- **Easy samples** – consistently high model confidence across training epochs.
- **Hard samples** – ambiguous, but potentially informative.
- **Noisy samples** – consistently low confidence and unstable predictions.

This is achieved through the Prior Generation Module, which estimates the clean probability of each sample based on its training history. The module first trains a classifier and records the confidence vector h_i for each sample d_i. By calculating the mean prediction confidence over epochs, samples are initially split into an easy set D_e, and a subset of low-confidence samples D_{n1} (assumed noisy).

To differentiate hard samples from remaining noisy data, artificial noise is injected into D_e to create a noisy version D_a. A second classifier M_m is trained on this data to distinguish between hard and noisy patterns. Finally, the ambiguous samples $D \setminus (D_e \cup D_{n1})$ are passed through M_m to determine the hard set D_h and an additional noisy set D_{n2}. The complete noisy set is then $D_n = D_{n1} \cup D_{n2}$.

As shown in Figure 6.14, this module integrates artificial noise to simulate realistic decision boundaries between clean-hard and noisy examples, enhancing the discriminative capability of the model.

Sample Dividing Optimization

To further refine the sample division, PGDF combines prior knowledge w_i^p with online training loss using a Gaussian Mixture Model (GMM) to compute a clean probability w_i. Specifically:

$$w_i = \begin{cases} 1 & \text{if } d_i \in D_e \\ m \cdot w_i^t + (1-m) \cdot w_i^p & \text{if } d_i \in D_h \cup D_n \end{cases}$$

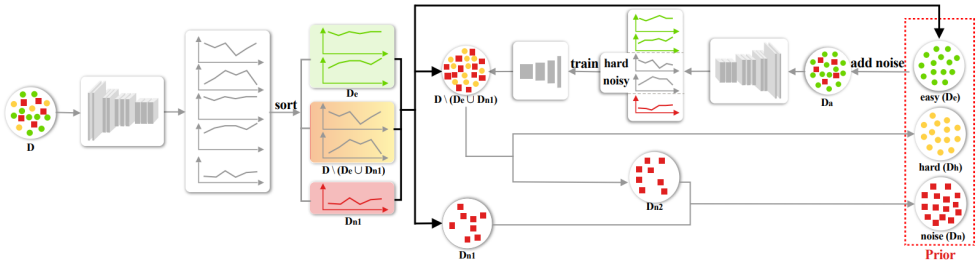


Figure 6.14. The overview of the Prior Generation Module. It pre-classifies the samples into an easy set, a hard set, and a noisy set. Adapted from: The Prior Guided Denoising Framework (PGDF) [10]

where w_i^t is the GMM-based estimate from current loss, and m is a mixing parameter. This results in a refined division into:

- $\tilde{D}_e$: Easy set (high clean probability)
- $\tilde{D}_h$: Hard set (moderate probability)
- $\tilde{D}_n$: Noisy set (low probability)

Each network in PGDF's dual-model setup performs this sample division for the other network, preventing self-confirmation bias—a strategy illustrated in Figure 6.15.

Stage 2: Denoising with the Divided Sets

After the sample sets are finalized, PGDF applies pseudo-label correction and reweighting:

Pseudo-label Refining: The model computes a transition matrix T using the reliable easy set $\tilde{D}_e$, and corrects pseudo-labels P via:

$$\tilde{P} = PT^{-1}$$

Negative values in $\tilde{P}$ are clamped to zero and normalized to form valid probability distributions.

Label Assignment: Noisy samples receive the corrected pseudo-labels $\tilde{P}$, while hard samples receive a blended label:

$$y_i = w_i y_i + (1 - w_i)p_i$$

where y_i is the original label and p_i the pseudo-label.

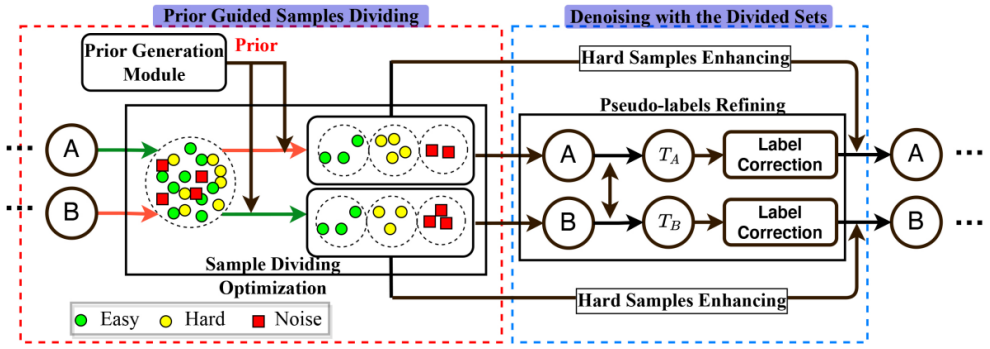


Figure 6.15. PGDF first generates the prior knowledge by the Prior Generation Module. Then, it trains two models (A and B) simultaneously. At each epoch, a model divides the original dataset into an easy set, a hard set, and a noisy set by combining the prior knowledge and the loss value of each sample. The divided dataset is used by the other network. After the first stage, the models conduct label correction for samples with the help of the estimated distribution transition matrix. Finally, the training loss is reweighted by the dividing result to further enhance the hard samples. Adapted from: The Prior Guided Denoising Framework (PGDF) [10]

Loss Reweighting and Training: Using MixMatch, PGDF creates mixed labeled and unlabeled batches. The training loss for hard and easy samples is reweighted by clean probability w_i:

$$L_X = -\frac{1}{|X'|} \sum_{x_i \in X'} \frac{1}{w_i^r} \sum_c p_c \log p_{c,\text{model}}(x_i)$$

The overall objective includes this supervised loss, an unsupervised mean squared error loss for unlabeled data, and a regularization term.

Effectiveness and Contribution

PGDF achieves state-of-the-art performance across various datasets (CIFAR-10/100, WebVision, Clothing1M) and noise conditions—including heavy (90%) and instance-dependent noise. Its key contribution lies in using training history to preserve hard examples and a carefully designed dual-model, multi-phase sample selection strategy.

By integrating sample prior knowledge, GMM-based online loss modeling, pseudo-label refinement, and loss reweighting, PGDF sets a strong benchmark in robust learning from noisy labels.

Feature-Based Taxonomy of Robust Learning Methods for PGDF

- **Architectural Structure**
  - **Using One Network: ✗**
    PGDF uses two networks (A and B) that divide the dataset for each other to avoid confirmation bias. This is explicitly part of its co-training design (see Figure 6.15).

  - **Using Two or Multiple Networks:** ✓
    The framework employs a dual-network structure where each model performs sample division for the other. This technique is inspired by DivideMix and Co-teaching.

- **Regularization Mechanisms**

  - **Class-Balanced Regularizer:** ✗
    PGDF does not use class rebalancing or reweighting based on class frequency. All sample weighting is based on clean probability, not class distribution.
  - **KL Divergence Regularizer:** ✗
    PGDF does not enforce distribution consistency via KL divergence between model predictions or between model pairs. No such loss appears in the formulation.
  - **Embedding Regularization:** ✗
    PGDF does not apply regularization directly to feature representations (e.g., via cosine similarity or norm penalties). Its focus is on output-based confidence and label corrections.
  - **Adding Additional Learnable Parameters:** ✗
    No additional trainable modules (e.g., noise adaptation layers) are introduced. The method only uses standard classifiers and a small CNN for prior estimation, which is not optimized jointly with the main model.

- **Learning Dynamics**

  - **Loss Regularization Only:** ✗
    PGDF is not a pure loss regularization method. Instead, it modifies the training process with sample selection, pseudo-labeling, and reweighting based on clean probabilities.
  - **Add Contrastive Loss:** ✗
    There is no contrastive learning component. PGDF does not use feature similarity or contrastive objectives.

- **Dependence on Auxiliary Information**

  - **Needs Clean Validation Dataset:** ✗
    PGDF is designed to work without any clean validation set or trusted labels. It operates in a fully noisy-label setting, using only training data.
  - **Add Semi-Supervised Loss:** ✓
    PGDF incorporates semi-supervised learning via MixMatch. Noisy samples are treated as unlabeled, and pseudo-labels are used in the unsupervised loss component (L_U), combined with labeled loss.
  - **Meta-Learning Framework:** ✗
    PGDF does not involve any meta-learning strategy (e.g., outer-loop optimization, adaptive loss learning, or learned sample weighting). All decisions are rule-based or based on training dynamics.

Empirical Cues in Noisy-Label Learning for PGDF

- **Small-Loss Criterion:** ✓
  PGDF uses small-loss indirectly in its GMM-based clean probability estimation (w_{it}), following the same intuition as DivideMix: samples with small loss are more likely clean.
- **Curriculum via Dynamic Thresholding:** ✗
  PGDF does not gradually include harder samples via evolving thresholds. It divides the dataset in a single-stage strategy based on fixed prior generation and clean probability blending.
- **Temporal Consistency of Predictions:** ✓
  PGDF relies heavily on training history, tracking model confidence over multiple epochs to identify easy, hard, and noisy samples. This is the foundation of its Prior Generation Module.
- **Noisy Validation Early-Stopping:** ✗
  PGDF does not rely on validation sets (clean or noisy) for early stopping. Training proceeds for a fixed number of epochs.
- **Normalized Loss Metrics:** ✗
  PGDF does not compare sample loss to class-averaged or global-normalized metrics to identify outliers. It only uses absolute loss in GMM.
- **Layer-Wise Memorization Rates:** ✗
  PGDF does not analyze differences in overfitting behavior across network layers. It treats the model as a black box in terms of memorization depth.
- **Model Confidence on Labels:** ✓
  PGDF's prior knowledge is based on model confidence (prediction probability) across epochs. This directly drives the sample classification into easy, hard, and noisy.
- **Soft Label Refinement:** ✓
  For noisy samples, PGDF replaces the original label with a refined pseudo-label computed from a corrected transition matrix. Hard samples also receive blended soft labels.
- **Inter-Network Agreement:** ✓
  PGDF trains two networks and uses each model's sample split to train the other, promoting consistency and reducing confirmation bias. This is an indirect form of agreement like in DivideMix.
- **Frequency-Domain Cues:** ✗
  PGDF does not operate in the frequency domain (e.g., using amplitude/phase components or Fourier transforms).
- **Sparsity of Noise (Label Errors as Outliers):** ✗
  PGDF does not use sparse modeling or treat label noise as outliers in an optimization objective.

- **Gradient Coherence:** ✗
  PGDF does not analyze or leverage gradient directions to distinguish clean vs. noisy samples.
- **Augmentation for Robustness:** ✓
  PGDF integrates MixMatch, which includes data augmentation such as random transformations and mixup. This is central to its supervised and unsupervised loss design.
- **Ensembling for Robustness:** ✗
  PGDF does not average predictions over epochs or use techniques like SWA or mean teacher. While it uses two models, there is no ensemble of predictions.
- **Consistency under Augmentation:** ✓
  PGDF inherits MixMatch's consistency loss, which encourages the model to produce similar predictions for different augmented views of the same sample.

6.2.12 ProMix

Overview of ProMix: Progressive Clean Sample Utilization for Noisy Label Learning

ProMix [86] is a robust deep learning framework designed to address the challenges of training with noisy labels. It belongs to the Sample Selection – Hybrid Approach category, as it combines clean sample identification with semi-supervised learning and bias mitigation techniques. The central motivation of ProMix is to maximize the utility of clean samples while mitigating the adverse effects of label noise.

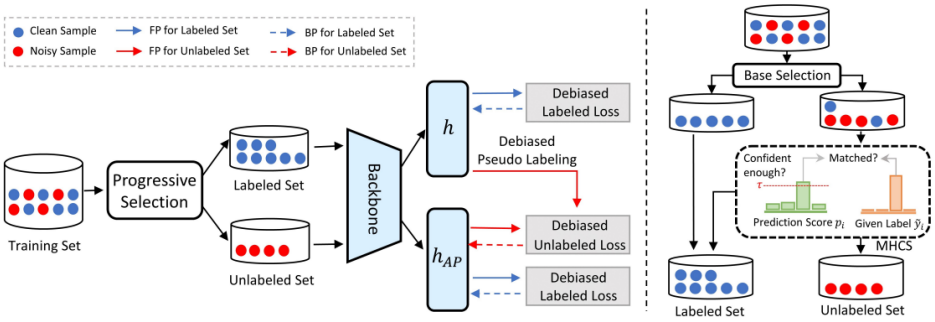
Motivation

Conventional methods like DivideMix focus on selecting a small, safe subset of clean examples to avoid label noise. However, this strategy often underutilizes many valuable clean samples. ProMix addresses this limitation by introducing a progressive sample selection mechanism that expands the clean set over time with high-confidence, label-aligned samples.

Method Overview

As illustrated in Figure 6.16, ProMix follows a three-stage pipeline:

- Progressive Clean Sample Selection
- Debiased Semi-Supervised Learning
- Bias-Calibrated Loss and Pseudo-Labeling



(a) Overall Framework of ProMix. (b) Progressive Selection.

Figure 6.16. The left: Overall framework of ProMix. After progressive selection, the source dataset is divided into the labeled set and unlabeled set for debiased SSL. The labeled data are fed forward to both primary head h and auxiliary pseudo h_{AP}, while the unlabeled data are provided with debiased pseudo-labels by h and then propagated forward through h_{AP}. Finally, the standard cross entropy in classification loss is replaced by debiased margin-based loss. FP/BP indicates forward/backward propagation. The right: Details of progressive selection. The MHCS selects those examples with high confidence and matched predictions with given labels to dynamically expand a base selection set. Adapted from: ProMix [86]

Progressive Sample Selection ProMix first divides the training data into:

A clean labeled set D_l

An unlabeled set D_u

This is achieved using two complementary mechanisms:

Class-wise Small-Loss Selection (CSS): Samples are selected based on their individual cross-entropy losses, within each class, to ensure balanced class coverage. Only the lowest-loss samples are chosen to form the initial clean set.

Matched High-Confidence Selection (MHCS): From the remaining data, samples are added to the clean set if their predicted label matches their noisy label and the prediction confidence exceeds a threshold τ. This ensures that high-confidence, correctly labeled examples are not overlooked.

As shown in Figure 6.16 (right), MHCS dynamically expands the clean set, maintaining high precision while improving recall.

Debiased Semi-Supervised Learning Once the data is split, ProMix performs semi-supervised learning using:

- The clean set D_l for supervised loss
- The unlabeled set D_u with pseudo-labels for unsupervised loss

To mitigate confirmation bias, ProMix uses two classification heads:

- **Primary head h:** Generates pseudo-labels from clean data only
- **Auxiliary pseudo head h_{AP}:** Learns from both clean and pseudo-labeled samples

The overall classification loss becomes:

$$L_{cls} = L_x(\{y_i, p_i\}) + L_x(\{y_i, p'_i\}) + \lambda_u L_u(\{\text{Sharpen}(p_i, T), p'_i\})$$

Where:

p_i: softmax predictions from h

p'_i: predictions from h_{AP}

λ_u: controls the strength of the unsupervised loss

The auxiliary head benefits from broader training exposure, while the primary head remains robust by learning only from the reliable clean set.

Bias Calibration To further enhance robustness, ProMix introduces two debiasing mechanisms:

Debiased Margin-Based Loss (DML): Adjusts the logits using estimated class distributions π, encouraging fairness between dominant and minority classes. The adjusted loss encourages the model to maintain a larger margin for overrepresented classes.

Debiased Pseudo-Labeling (DPL): Refines pseudo-labels by down-weighting overconfident classes, ensuring that predictions do not overly favor frequent classes.

The class distribution π is estimated using a moving average over training batches, and is tracked separately for D_l and D_u.

Results and Effectiveness

ProMix demonstrates state-of-the-art results across synthetic and real-world noisy datasets (CIFAR-10/100, CIFAR-N, Clothing1M), particularly under heavy and instance-dependent noise. Its clean sample selection strategy outperforms baselines like DivideMix in both precision and recall, effectively maximizing the clean sample utility throughout training.

Feature-Based Taxonomy of Robust Learning Methods for ProMix

- **Architectural Structure**
  - **Using One Network:** ✓
    ProMix uses a single neural network with two heads (h and h_{AP}), but not two separate networks. These heads share the same backbone and are part of the same architecture.
  - **Using Two or Multiple Networks:** ✗
    Unlike Co-teaching or DivideMix, ProMix does not use separate networks trained in parallel. It relies on a single shared model with distinct heads internally.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✓
    ProMix uses Debiased Margin-Based Loss, which adjusts the logits based on estimated class distributions, making it a class-aware regularizer. This explicitly addresses class imbalance in training.
  - **KL Divergence Regularizer:** ✗
    ProMix does not use KL divergence explicitly for enforcing prediction consistency (unlike JoCoR or ELR). Instead, it uses cross-entropy with debiased logits.
  - **Embedding Regularization:** ✗
    ProMix does not apply regularization directly to feature embeddings. It focuses on sample selection, pseudo-labels, and output-space regularization.
  - **Adding Additional Learnable Parameters:** ✗
    There are no new trainable modules (e.g., noise adaptation layers). The additional head (h_{AP}) shares the same backbone, and adds no independent learnable components beyond standard classifier parameters.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✓
    ProMix modifies the loss function (via DML and pseudo-label sharpening) without changing architecture. So, it qualifies as using loss regularization techniques.
  - **Add Contrastive Loss:** ✗
    ProMix does not use contrastive learning objectives. It is purely classification-driven with supervised and semi-supervised losses.

- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    ProMix is trained without any clean validation set. All clean samples are selected from the noisy data itself via CSS and MHCS.
  - **Add Semi-Supervised Loss:** ✓
    ProMix integrates semi-supervised learning by treating the noisy/unlabeled samples with pseudo-labels and applying unsupervised consistency loss.
  - **Meta-Learning Framework:** ✗
    ProMix does not use meta-learning techniques for weighting samples or learning loss parameters dynamically. Everything is hand-designed and heuristically controlled.

Empirical Cues in Noisy-Label Learning for ProMix

- **Small-Loss Criterion:** ✓
  ProMix uses Class-wise Small-Loss Selection (CSS) to build its initial clean sample set, based on the idea that DNNs memorize clean samples earlier and assign them lower losses.
- **Curriculum via Dynamic Thresholding:** ✗
  ProMix does not gradually increase the threshold to include harder samples over time. The MHCS threshold is fixed, and selection is binary: either a sample meets the confidence + agreement criterion or not.
- **Temporal Consistency of Predictions:** ✗
  ProMix does not monitor or enforce temporal prediction stability (e.g., consistency over multiple epochs or EMA of predictions). It uses current predictions only.
- **Noisy Validation Early-Stopping:** ✗
  ProMix does not rely on validation accuracy for early stopping. Training proceeds for a fixed number of epochs.
- **Normalized Loss Metrics:** ✗
  Sample losses are not normalized across the dataset or classes. ProMix selects clean samples per class, but based on absolute loss, not normalized loss.
- **Layer-Wise Memorization Rates:** ✗
  ProMix does not analyze or exploit different learning behaviors across layers (e.g., how shallow layers generalize vs. deep layers overfit).
- **Model Confidence on Labels:** ✓
  Matched High-Confidence Selection (MHCS) is based on prediction confidence. ProMix selects samples where:
  - Model is confident (above a threshold), and
  - Prediction matches the noisy label

- **Soft Label Refinement:** ✓
  In Debiased Pseudo-Labeling (DPL), ProMix modifies logits based on class frequency and applies softmax – effectively creates soft pseudo-labels instead of relying on hard labels.
- **Inter-Network Agreement:** ✗
  Unlike Co-teaching or DivideMix, ProMix does not use two separate networks for agreement-based selection. All decisions come from one network with two heads, but not cross-model agreement.
- **Frequency-Domain Cues:** ✗
  ProMix does not perform or use frequency-based analysis (e.g., amplitude-phase separation) in its selection or training processes.
- **Sparsity of Noise (Label Errors as Outliers):** ✗
  ProMix does not apply sparsity constraints or explicitly model label noise as rare outliers in a structured loss function or representation space.
- **Gradient Coherence:** ✗
  ProMix does not evaluate or exploit coherence in gradient directions as a cue for identifying clean vs. noisy samples.
- **Augmentation for Robustness:** ✓
  ProMix applies strong data augmentation techniques (e.g., RandAugment, Mixup) during SSL and consistency training to improve robustness to noise.
- **Ensembling for Robustness:** ✓
  During inference, ProMix ensembles outputs from two peer networks to improve prediction stability and reduce overfitting.
- **Consistency under Augmentation:** ✓
  ProMix encourages consistency between weak and strong augmentations during SSL. Predictions on augmented views are matched to sharpened pseudo-labels, promoting stable learning.

6.2.13 SOP and SOP+

Overview of SOP: Sparse Over-Parameterization and SOP+

The SOP [44] method addresses learning from noisy labels by explicitly modeling label corruption as a sparse correction to the network output. Given a sample (x_i, y_i), where y_i may be noisy, SOP augments the model prediction with a learnable correction term s_i, leading to the modified loss:

$$\min_{\theta,\{u_i,v_i\}} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i;\theta) + s_i,\, y_i), \quad \text{with} \quad s_i = u_i \odot u_i \odot y_i - v_i \odot v_i \odot (1 - y_i),$$

where $u_i, v_i \in \mathbb{R}^K$ are sample-specific parameters initialized near zero and optimized via gradient descent. This formulation ensures that only the necessary corrections emerge over training, effectively achieving robustness through implicit regularization. No explicit sparsity penalty is required — the optimization path naturally leads to sparse s_i.

The SOP method trains the main network parameters θ along with u_i and v_i, using standard cross-entropy (with normalization) or mean squared error losses. Importantly, it uses no clean labels, no pretraining, and retains all training samples.

SOP+ extends SOP by adding:

- Consistency regularization across augmentations
- Class-balance regularization to stabilize training under severe imbalance

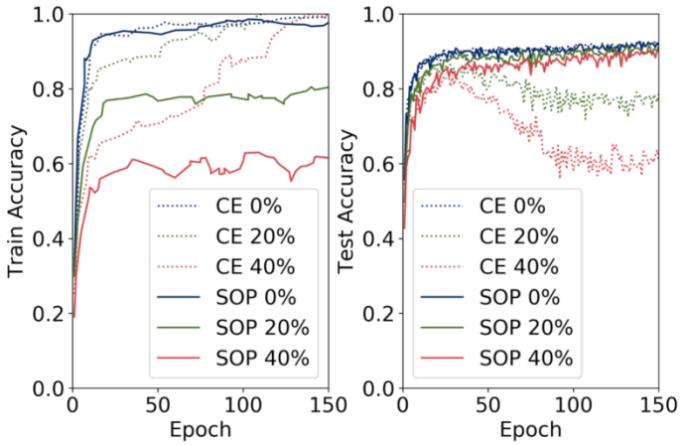


Figure 6.17. SOP prevents overfitting to noisy labels. Compared to standard cross-entropy, SOP maintains higher test accuracy on CIFAR-10 under 0%, 20%, and 40% label noise by avoiding memorization of corrupted labels. Adapted from: SOP [44]

As illustrated in Figure 6.17, SOP avoids memorizing label noise. While the training loss continues to decrease, the corrected prediction $f(x) + s$ maintains high test accuracy by not overfitting to corrupted labels.

This simple and model-agnostic approach demonstrates competitive performance across synthetic and real-world noisy datasets (e.g., CIFAR-10N, Clothing1M), and SOP+ outperforms state-of-the-art methods under heavy noise conditions.

Feature-Based Taxonomy of Robust Learning Methods for SOP and SOP+

- **Architectural Structure**
  - **Using One Network:** ✓
    Both methods rely on a single neural network (e.g., ResNet) and do not require co-training or network ensembles.
  - **Using Two or Multiple Networks:** ✗
    SOP/SOP+ does not involve any collaborative training or ensemble methods.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✓(SOP+ only)
    SOP+ includes a class-balance regularization term to mitigate the impact of class imbalance under heavy label noise. SOP does not include this component.
  - **KL Divergence Regularizer:** ✗
    SOP and SOP+ do not include KL divergence terms in their loss formulation.
  - **Embedding Regularization:** ✗
    No regularization is applied directly to feature embeddings (i.e., internal network representations).
  - **Adding Additional Learnable Parameters:** ✓
    SOP introduces per-sample learnable parameters u_i and v_i, which are updated during training to model label noise.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✗
    SOP does not regularize the loss itself; rather, it adjusts model outputs before computing the loss. Its robustness comes from optimization structure, not modified loss terms.
  - **Add Contrastive Loss:** ✗
    SOP/SOP+ does not use contrastive or instance-level discrimination objectives.
- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    Neither SOP nor SOP+ requires a clean validation set. Training is fully unsupervised with respect to noise detection.
  - **Add Semi-Supervised Loss:** ✗
    SOP/SOP+ does not incorporate unlabeled data or semi-supervised components such as pseudo-labeling or consistency with unlabeled examples.
  - **Meta-Learning Framework:** ✗
    No meta-learning strategy is involved (e.g., no dynamic reweighting based on learned meta-parameters).

Empirical Cues in Noisy-Label Learning for SOP and SOP+

- **Small-Loss Criterion:** ✗
  SOP does not select or prioritize low-loss samples. All data points are used equally; no curriculum or early-clean sample selection is applied.
- **Curriculum via Dynamic Thresholding:** ✗
  SOP does not use an evolving threshold to include harder examples. There's no progressive sample selection.
- **Temporal Consistency of Predictions:** ✓(SOP+ only)
  SOP+ adds a consistency regularization term encouraging stability of predictions under augmentations, which aligns with this cue. SOP (base) does not include this.
- **Noisy Validation Early-Stopping:** ✗
  SOP/SOP+ does not use a clean or noisy validation set to determine early stopping. Training runs for a fixed number of epochs.
- **Normalized Loss Metrics:** ✗
  SOP does not normalize losses across samples or classes to detect noise. The correction is learned directly from optimization without loss scaling.
- **Layer-Wise Memorization Rates:** ✗
  SOP does not analyze or exploit differences in memorization behavior across layers.
- **Model Confidence on Labels:** ✗
  SOP does not use prediction confidence as a noise indicator. It models the noise via learnable corrections instead.
- **Soft Label Refinement:** ✗
  SOP keeps the original (hard) labels and does not refine or smooth them based on predictions or distributions.
- **Inter-Network Agreement:** ✗
  SOP/SOP+ uses only a single network and does not employ agreement strategies like Co-teaching or JoCoR.
- **Frequency-Domain Cues:** ✗
  SOP does not analyze or manipulate frequency components (e.g., phase/amplitude) in feature representations.
- **Sparsity of Noise (Label Errors as Outliers):** ✓
  This is the core mechanism of SOP: label noise is treated as sparse error s_i, isolated via optimization dynamics and over-parameterization. This aligns directly with this empirical cue.
- **Gradient Coherence:** ✗
  SOP does not measure or enforce gradient similarity across samples to detect noise.

- **Augmentation for Robustness:** ✓(SOP+ only)
  SOP+ uses data augmentation (e.g., random crops, flips) combined with consistency regularization. SOP (base) does not rely on augmentation in its core design.
- **Ensembling for Robustness:** ✗
  SOP/SOP+ does not perform ensembling or use temporal averaging across epochs.
- **Consistency under Augmentation:** ✓(SOP+ only)
  SOP+ explicitly includes a term that encourages prediction consistency across different augmentations of the same image.

6.2.14 SURE

Overview of SURE: A Unified Framework for Reliable and Robust Deep Networks

SURE (SUrvey REcipes) [4] is a training framework that integrates complementary strategies for uncertainty estimation and robustness in deep neural networks. It combines techniques from regularization, classifier design, and optimization to produce models that are both accurate and uncertainty-aware—without requiring task-specific adjustments.

The method is centered around two key principles:

- Increasing entropy for hard samples, to reduce overconfidence on uncertain inputs.
- Enforcing flat minima, to improve generalization and stability under noise or distribution shifts.

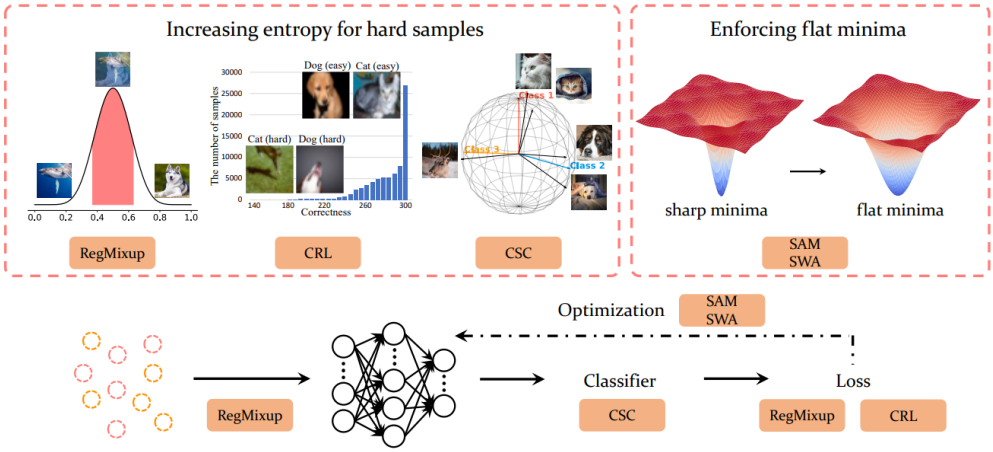


Figure 6.18. Overview of the SURE framework, which combines two key components—increasing entropy for hard samples (via RegMixup, CRL, and CSC) and enforcing flat minima (via SAM and SWA)—to improve robustness and uncertainty estimation. Adapted from: SURE (SUrvey REcipes) [4]

These are implemented via:

- **RegMixup:** a strong data augmentation method that interpolates between samples to regularize the network,
- **Correctness Ranking Loss (CRL):** which aligns model confidence with historical correctness,
- **Cosine Similarity Classifier (CSC):** improving robustness by angular decision boundaries,
- **Sharpness-Aware Minimization (SAM)** and **Stochastic Weight Averaging (SWA):** for finding flat, generalizable minima.

The total training objective is:

$$L_{\text{total}} = L_{\text{ce}} + \lambda_{\text{mix}} L_{\text{mix}} + \lambda_{\text{crl}} L_{\text{crl}},$$

where L_{ce} is the standard cross-entropy loss, L_{mix} is the RegMixup loss applied on interpolated samples, and L_{crl} penalizes mismatch between model confidence and sample correctness rankings.

As we can see in Figure 6.18, the architecture of SURE integrates these components in a unified pipeline, distinguishing between classifier-level and optimization-level contributions to robustness.

SURE achieves strong empirical performance on failure prediction, noisy label learning (e.g., Animal-10N, Food-101N), long-tailed classification (CIFAR-LT), and corruption benchmarks (CIFAR10-C), outperforming or matching specialized methods—all without requiring clean validation sets or pretraining.

Feature-Based Taxonomy of Robust Learning Methods for SURE

- **Architectural Structure**
  - **Using One Network:** ✓
    Used. SURE trains a single neural network without using auxiliary or co-teaching models. All components (e.g., RegMixup, CRL, CSC) are applied to a single model architecture.
  - **Using Two or Multiple Networks:** ✗
    Not Used. There is no mention of multiple networks, cross-teaching, or ensemble-based approaches.
- **Regularization Mechanisms**
  - **Class-Balanced Regularizer:** ✗
    Not Used. SURE does not apply any class reweighting or loss adjustment specifically to address class imbalance (although it evaluates on long-tailed data, the method itself is not class-balanced).
  - **KL Divergence Regularizer:** ✗
    Not Used. No KL divergence penalty is used to enforce prediction consistency.
  - **Embedding Regularization:** ✗
    Not Used. SURE does not apply regularization directly on intermediate feature representations (e.g., no L2 norm or manifold smoothing in embedding space).
  - **Adding Additional Learnable Parameters:** ✗
    Not Used. SURE adds no extra modules or trainable parameters like noise adaptation layers or confidence estimators.
- **Learning Dynamics**
  - **Loss Regularization Only:** ✓
    Used. SURE introduces explicit loss regularization (CRL and RegMixup) without modifying the network architecture or using external datasets—this aligns with this feature.

  - **Add Contrastive Loss:** ✗
    Not Used. There is no contrastive learning objective (e.g., SimCLR, SupCon) in the method.

- **Dependence on Auxiliary Information**
  - **Needs Clean Validation Dataset:** ✗
    Not Used. SURE is trained without clean labels or supervision. Hyperparameters are tuned using a small held-out validation set from the same noisy dataset (not clean).
  - **Add Semi-Supervised Loss:** ✗
    Not Used. SURE is a fully supervised approach. While its robustness allows it to be evaluated on semi-supervised benchmarks, no semi-supervised loss is integrated.
  - **Meta-Learning Framework:** ✗
    Not Used. SURE does not use meta-learning techniques like bi-level optimization or learned sample weighting.

Empirical Cues in Noisy-Label Learning for SURE

- **Small-Loss Criterion:** ●
  SURE does not explicitly use the small-loss criterion (e.g., it doesn't filter or reweight based on current loss). But CRL does take advantage of the memorization effect indirectly, by learning from the discrepancy between confidence and correctness history.
- **Curriculum via Dynamic Thresholding:** ✗
  There is no progressive thresholding or dynamic inclusion of harder samples based on model confidence or loss. All examples are used from the start.
- **Temporal Consistency of Predictions:** ✗
  SURE does not enforce temporal consistency (e.g., across epochs or augmentations). There is no mechanism penalizing unstable predictions over time.
- **Noisy Validation Early-Stopping:** ✗
  SURE runs for a fixed number of epochs. There is no early stopping based on noisy or clean validation accuracy.
- **Normalized Loss Metrics:** ✗
  SURE does not compute normalized loss across classes or global averages to detect noisy samples. All regularization is loss-agnostic in this sense.
- **Layer-Wise Memorization Rates:** ✗
  The method does not exploit different memorization behavior across layers or use layer-wise cues to detect noise.
- **Model Confidence on Labels:** ✓
  The Correctness Ranking Loss (CRL) uses model confidence scores (softmax values) to regularize predictions in relation to historical correctness. This aligns directly with this cue.

- **Soft Label Refinement:** ✗
  SURE does not refine hard labels into soft targets. It operates with original labels and does not generate pseudo-labels or probability distributions as targets.
- **Inter-Network Agreement:** ✗
  SURE is a single-network method. There is no use of agreement between two or more networks.
- **Frequency-Domain Cues:** ✗
  SURE does not analyze features in the frequency domain or manipulate phase/amplitude components.
- **Sparsity of Noise (Label Errors as Outliers):** ✗
  SURE does not use any sparsity-inducing regularization or treat label noise as sparse outliers.
- **Gradient Coherence:** ✗
  The method does not explicitly analyze gradient directions to distinguish noisy vs. clean samples.
- **Augmentation for Robustness:** ✓
  RegMixup is a strong data augmentation technique used explicitly in SURE to increase entropy and improve robustness.
- **Ensembling for Robustness:** ✓
  SWA (Stochastic Weight Averaging) is an ensemble-like technique used in SURE to stabilize training and reduce variance in predictions.
- **Consistency under Augmentation:** ✗
  SURE does not measure or enforce consistency across augmented views (e.g., as done in contrastive learning or semi-supervised consistency methods).

6.3 Concluding Remarks and Impact of the Comparative Analysis

To build a principled and comprehensive understanding of the most effective strategies for learning with noisy labels, this chapter has presented a detailed, method-by-method analysis of 14 representative techniques. Each method was examined through two complementary lenses:

Feature-Based Taxonomy (Table 6.1): This structural perspective categorizes methods according to eleven design features, including architectural setup, regularization mechanisms, learning dynamics, and dependence on auxiliary information. It allows us to understand what each method is composed of and how it is constructed.

Empirical Cue Framework (Table 6.2): This behavioral axis captures the operational heuristics that methods rely on—whether explicitly or implicitly—during training. These cues reflect how methods behave, particularly in how they identify, trust, or suppress noisy samples. Examples include the small-loss criterion, temporal prediction consistency, model confidence, augmentation consistency, and feature-space stability.

The motivation behind combining these two frameworks is rooted in the insight that robustness to label noise cannot be fully explained by architecture alone. Many modern methods share similar components (e.g., early stopping, soft labels, dual networks) but differ in how they exploit empirical regularities during training. By bringing both the structural and behavioral layers into the analysis, we can answer not just *"what a method does"*, but also *"why it works"*.

This dual-perspective analysis provides several valuable outcomes:

- **Clarification of Design Principles:** It helps disentangle whether performance gains stem from architectural innovations, empirical heuristics, or their interaction.
- **Identification of Common Patterns:** It reveals which structural components and empirical cues recur across high-performing methods—offering evidence of best practices in the field.
- **Support for Rigorous Comparison and Benchmarking:** The results of the analysis are summarized in two comprehensive tables—one for the feature taxonomy and one for empirical cues—offering a compact but informative overview of all analyzed methods.
- **Groundwork for Future Research:** By distinguishing redundant mechanisms from novel contributions, and by exposing underexplored empirical signals, this analysis supports more targeted and transparent development of future approaches.

Ultimately, this two-axis analysis—what methods are built from and how they behave in training—offers a grounded and multi-dimensional understanding of state-of-the-art solutions to noisy-label learning. It sets the stage for the following comparative discussion, where these insights will be synthesized to highlight trends, trade-offs, and future directions.

Table 6.1. Feature-Based Taxonomy of Robust Learning Methods

Feature	ADELE	CORES^2	ELR/ELR$^+$	GNL	ILL	L2B/L2B-C2D	NCOD/NCOD$^+$	NES	PADDLES	PES	PGDF	ProMix	SOP/SOP$^+$	SURE
D1														
f1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
f2	✗	✗	✓(ELR$^+$)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
D2														
f3	✗	✗	✗	✗	✗	✗	✓(NCOD$^+$)	✗	✗	✓	✗	✓	✓(SOP$^+$)	✗
f4	✓	✓	✗	✓	✗	✗	✓(NCOD$^+$)	✗	✗	✗	✗	✗	✗	✗
f5	✗	✗	✗	✗	✗	✓(L2B-C2D)	✓	✗	●	✗	✗	✗	✗	✗
f6	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗
D3														
f7	●	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
f8	✗	✗	✗	✗	✗	✓(L2B-C2D)	✗	✗	✗	✗	✗	✗	✗	✗
D4														
f9	✗	✗	✗	✗	✗	●	✗	✗	✗	✗	✗	✗	✗	✗
f10	✗	✓	✓(ELR$^+$)	✗	✓	✓(L2B-C2D)	✗	✗	●	✓	✓	✓	✗	✗
f11	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗

Legend:

- ✓ = Fully supported
- ✗ = Not supported
- ● = Partially supported

Feature Dimension (Dx) and Feature Labels (fx):

- **D1: Architectural Structure**
  - f1: Using One Network
  - f2: Using Two or Multiple Networks
- **D2: Regularization Mechanisms**
  - f3: Class-Balanced Regularizer
  - f4: KL Divergence Regularizer
  - f5: Embedding Regularization
  - f6: Adding Additional Learnable Parameters
- **D3: Learning Dynamics**
  - f7: Loss Regularization Only
  - f8: Add Contrastive Loss
- **D4: Dependence on Auxiliary Information**
  - f9: Needs Clean Validation Dataset
  - f10: Add Semi-Supervised Loss
  - f11: Meta-Learning Framework

Table 6.2. Empirical Cues in Noisy-Label Learning Methods

Empirical Cue	ADELE	CORES^2	ELR/ELR$^+$	GNL	ILL	L2B/C2D	NCOD/NCOD$^+$	NES	PADDLES	PES	PGDF	ProMix	SOP/SOP$^+$	SURE
e1	✗	✓	✗	✗	●	✓	✓	●	●	●	✓	✓	✗	●
e2	●	✓	✗	●	✗	✗	●	✗	✗	✗	✗	✗	✗	✗
e3	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✓(SOP$^+$)	✗
e4	●	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
e5	✗	●	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
e6	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
e7	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✗	✓
e8	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗
e9	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
e10	✗	✗	✗	✗	✗	●	✗	✗	✓	✗	✗	✗	✗	✗
e11	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗
e12	✗	●	✓	●	✗	✗	●	✗	✗	✗	✗	✗	✗	✗
e13	✓	✓	✓(ELR$^+$)	●	✗	✓(C2D only)	✓(NCOD$^+$	✗	●	✓	✓	✓(SOP$^+$)	✓	✓
e14	●	✗	✓(ELR$^+$)	●	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓
e15	✓	✓	✓(ELR$^+$)	✗	✗	✓(C2D only)	✓(NCOD$^+$	✗	✗	✓	✓	✓(SOP$^+$)	✗	✗

Legend for Empirical Cue Codes:

- **e1** Small-Loss Criterion.
- **e2** Curriculum via Dynamic Thresholding.
- **e3** Temporal Consistency of Predictions.
- **e4** Noisy Validation Early-Stopping.
- **e5** Normalized Loss Metrics.
- **e6** Layer-Wise Memorization Rates.
- **e7** Model Confidence on Labels.
- **e8** Soft Label Refinement.
- **e9** Inter-Network Agreement.
- **e10** Frequency-Domain Cues.
- **e11** Sparsity of Noise.
- **e12** Gradient Coherence.
- **e13** Augmentation for Robustness.
- **e14** Ensembling for Robustness.
- **e15** Consistency under Augmentation.

Chapter 7

Experiments

7.1 Design and Implementation of the Experimental Framework

Motivation

Evaluating and comparing methods for learning with noisy labels remains a challenge in the literature. Published results are scattered across diverse datasets (e.g., CIFAR-10, Clothing1M, WebVision, and various synthetic noise benchmarks), often with differing noise rates, network architectures, training schedules, and evaluation protocols. As a result, reproducing and fairly comparing competing approaches can consume significant time and effort—and may still yield inconclusive conclusions due to hidden implementation details.

Our experimental framework addresses these issues by providing:

- A unified codebase for multiple state-of-the-art noisy-label methods (CORES, ELR, ELR$^+$, SOP, etc.),
- Centralized configuration files for dataset selection, noise specification, model architectures, and hyperparameters,
- Standardized logging of training / validation / test metrics and wall-clock timings (training vs. inference), and
- Reproducibility controls, including fixed random seeds and consistent data-loading pipelines.

By abstracting away method-specific boilerplate and enforcing consistent evaluation protocols, this framework empowers practitioners to benchmark existing and new algorithms under identical conditions—dramatically lowering the barrier to reproducible, fair comparisons.

Framework Architecture

At a high level, the framework consists of three layers:

Core Utilities

Reproducibility Module

At program start, we seed Python's `random`, `NumPy`, and `PyTorch` (CPU & GPU) RNGs to a user-specified value (e.g., `seed = 123`), ensuring identical shuffles, augmentations, and initial weights across methods.

Logger A unified `Logger` class captures per-epoch metrics (loss, accuracy, learning rate, timing) and writes them both to console and to structured log files under:

```
results/<method>/<dataset>/<model_type>/run_<timestamp>/
```

Additionally, the framework supports automatic generation of training plots based on the logged data. For example, users can visualize accuracy and loss curves by running:

```
python plot.py -history models/SOP/history.npz
```

This produces line plots that track Top-1 and Top-5 accuracy (or other metrics) over epochs, helping users diagnose learning dynamics and performance trends at a glance.

Model Wrappers

For each algorithm—CORES, ELR, ELR$^+$, SOP—the framework provides a minimal wrapper (`wrappers/<method>_wrapper.py`) that:

- Parses a high-level JSON config.
- Invokes the original training script (via `subprocess`) in its own repo.
- Streams live progress back to your console.
- Extracts just the key final lines (e.g., "Epoch X completed. Train Acc: ..."), measures wall-clock training vs. inference time, and appends them to our standardized log files.

Because each method retains its own codebase untouched, integrating new algorithms is as simple as writing a small wrapper that points at its entry script, matches its printed output format, and providing a JSON config file that specifies its hyperparameters (dataset, seed, noise settings, optimizer, etc.). The framework's top-level driver then reads that config and invokes the wrapper automatically—no core code changes required.

Top-Level Driver

The central driver `scripts/train.py` does nothing more than:

- Read your JSON config (example below),
- Seed all RNGs,

- Dispatch to the appropriate wrapper (or fall back to the built-in "simple baseline" trainer),
- Exit once the wrapper finishes.

Example Configuration for SOP: Sparse Over-Parameterization Method

```
{
  "model":       "sop",
  "architecture":"PreActResNet18",
  "dataset":     "CIFAR-10",
  "noise": {
    "type":      "symmetric",
    "rate":      0.6
  },
  "augment": {
    "autoaugment": true,
    "cutout":      16
  },
  "optimizer": {
    "type":      "SGD",
    "lr":        0.02,
    "momentum":  0.9,
    "wd":        1e-3
  },
  "loss":        "overparametrization (consistency=0.9, balance=0.1)",
  "metrics":    ["Top-1 Accuracy", "Top-5 Accuracy"],
  "training": {
    "epochs":    10,
    "batch_size":128,
    "seed":      123
  },
  "results":     "results/sop/cifar10_sop_run"
}
```

A single command:

```
python -m scripts.train -c configs/sop_cifar10.json
```

runs any method under the same conventions and produces comparable logs.

Training Log Analysis

To evaluate the effectiveness of each method, we extract key performance metrics from the training logs. For every epoch, we record the training loss, test loss, and accuracy values. This provides a detailed, time-resolved view of the model's learning progression and generalization behavior under label noise.

Example (SOP on CIFAR-10, Symmetric Noise 60%): Table 7.1 summarizes the epoch-wise training and test performance of SOP. it reports the train and test loss, as well as top-1 and top-5 accuracy. The model demonstrates steady improvement throughout training.

This log-based reporting enables both quantitative benchmarking and qualitative insights into memorization dynamics.

Table 7.1. Training and Test Metrics (Loss, Top-1 Accuracy, Top-5 Accuracy) for SOP on CIFAR-10 (Symmetric Noise 60%)

Epoch	Train Loss	Test Loss	Train Top-1	Train Top-5	Test Top-1	Test Top-5
1	4.1276	1.7714	0.1888	0.6288	0.4315	0.8919
2	3.9760	1.6851	0.2399	0.6641	0.4779	0.9225
3	3.8896	1.6391	0.2698	0.6764	0.4818	0.9118
4	3.8230	1.5487	0.2916	0.6803	0.5784	0.9424
5	3.7705	1.4705	0.3081	0.6853	0.6178	0.9487
6	3.7225	1.4083	0.3234	0.6899	0.6617	0.9625
7	3.6800	1.3767	0.3371	0.6939	0.6716	0.9653
8	3.6438	1.3734	0.3489	0.6910	0.7049	0.9606
9	3.6197	1.3618	0.3546	0.6994	0.6775	0.9523
10	3.5961	1.3874	0.3620	0.7020	0.7065	0.9588

Timing Measurement

To assess the computational cost of each method, we record both training and inference times:

- **Training Time:** Measured by timestamping just before the first training epoch begins and immediately after the final parameter update. This includes the total time required to optimize the model over the specified number of epochs.
- **Inference Time:** Measured separately during evaluation. It includes the time spent on validation and final testing, averaged across epochs where applicable.

Example (SOP on CIFAR-10, Symmetric Noise 60%):

- Training Time (Epoch 1): 302.86 seconds
- Training Time (Epoch 10): 298.04 seconds
- Inference Time (Epoch 1): 42.59 seconds
- Inference Time (Epoch 10): 42.94 seconds

These measurements were automatically logged per epoch during training, allowing us to track computational efficiency and detect any anomalies during evaluation.
.

Training Dynamics Visualization

To better understand the learning behavior of robust learning methods under noisy supervision, we can visualize key metrics over the course of training. As an example, Figures 7.1, 7.2, and 7.3 illustrate the training dynamics of the SOP model on CIFAR-10 with 60% symmetric label noise. Specifically, they show:

- The trend of training and test loss over epochs,
- The evolution of top-1 accuracy for both training and test sets,
- The evolution of top-5 accuracy for both training and test sets.

These visualizations reveal a consistent reduction in loss and improvement in accuracy, confirming the numerical results reported in Table 7.1. They also offer a clearer view of the learning progression and generalization performance throughout training.

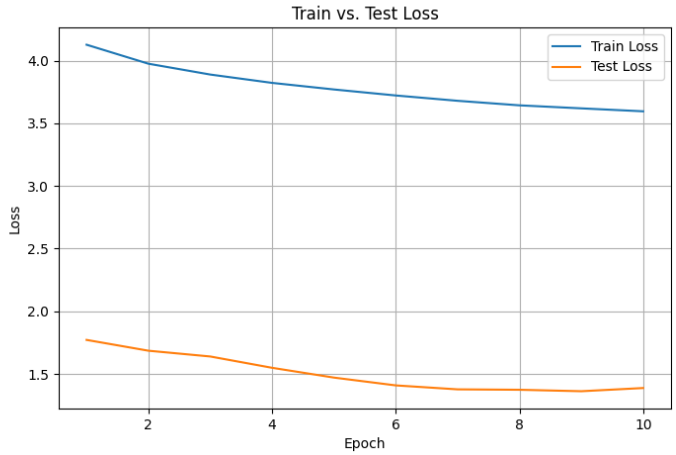


Figure 7.1. Training and Test Loss over Epochs for SOP on CIFAR-10 (60% Symmetric Noise)

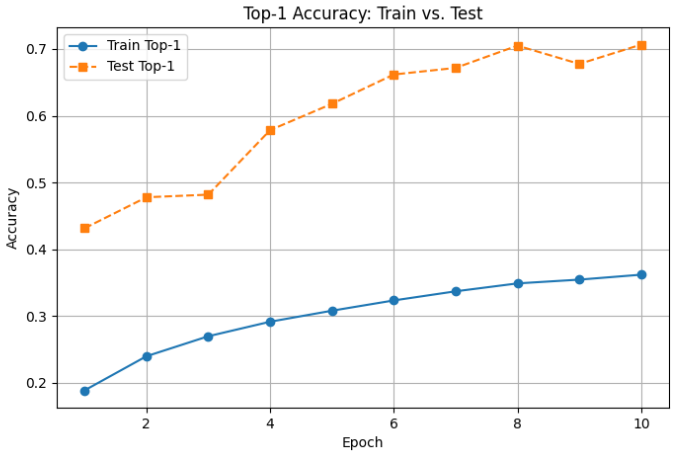


Figure 7.2. Top-1 Accuracy (Train and Test) for SOP across 10 Training Epochs

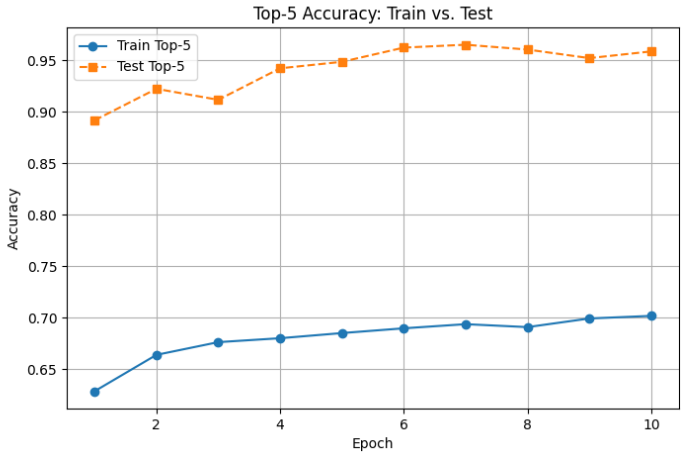


Figure 7.3. Top-5 Accuracy (Train and Test) for SOP across 10 Training Epochs

Summary

This experimental framework enforces a unified seeding strategy, shared logging conventions, and lightweight "wrapper" adapters around each official codebase—whether it's CORES, ELR, SOP, or any other future method. To add a new algorithm, you simply write a small wrapper pointing at its entry script, define a corresponding JSON config file, and map its key outputs into our logs. As a result, comparing methods side-by-side—on the same hardware, with identical randomness, and across identical datasets—becomes trivial. Moving forward, researchers and practitioners can extend the framework by adding new wrappers, integrating additional datasets, or experimenting with novel evaluation protocols, all without altering the core pipeline. The full implementation is available on GitHub:

`https://github.com/Davood-sh/Noisy-Label-Learning-Experimental-Framework.git`

7.2 Noise Rate Estimation

The estimation of the noise rate τ is an imperative part of utilizing robust methods for better practical performance—particularly for loss-adjustment and sample-selection approaches. The estimated $\hat{\tau}$ is widely used to reweight examples for a robust classifier or to determine how many examples should be selected as clean ones. However, despite its centrality, detailed analysis of estimator accuracy remains limited. Below we detail the three most common estimation strategies:

A. Noise Transition Matrix

The noise transition matrix

$$T_{ij} = \Pr(\tilde{y} = j \mid y = i)$$

encodes class-conditional flip probabilities and underpins statistically consistent robust classifiers. To estimate τ, one first infers $\hat{T}$ via *anchor points*—examples x for which $p(y = i \mid x) = 1$. Let

$$A_i = \{x : p(y = i \mid x) = 1\},$$

then for each i, j:

$$\hat{T}_{ij} = \frac{1}{|A_i|} \sum_{x \in A_i} p(\tilde{y} = j \mid x; \Theta) \tag{7.1}$$

where $p(\tilde{y} = j \mid x; \Theta)$ is the noisy-label posterior of a network trained on the corrupted dataset.

The overall noise rate on balanced data follows:

$$\hat{\tau}_{\mathrm{TM}} = \frac{1}{c} \sum_{i=1}^{c} \sum_{j \neq i} \hat{T}_{ij} = 1 - \frac{1}{c} \sum_{i=1}^{c} \hat{T}_{ii} \tag{7.2}$$

Since true anchor points are not known a priori, they are identified via theoretical criteria, heuristic selection, or by end-to-end schemes that relax anchor assumptions.

B. Gaussian Mixture Model (GMM)

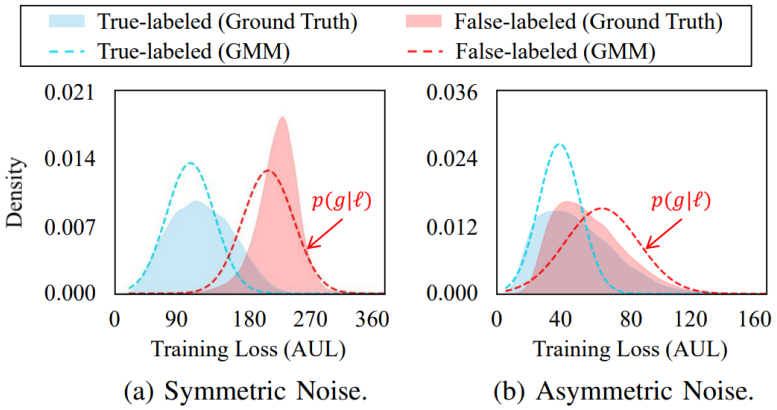
True-labeled and false-labeled examples typically exhibit a bimodal distribution of per-example losses $\{\ell_i\}$ after a warm-up phase. We fit a two-component GMM:

$$p(\ell) = \pi_{\mathrm{clean}} \cdot \mathcal{N}(\ell; \mu_1, \sigma_1^2) + \pi_{\mathrm{noisy}} \cdot \mathcal{N}(\ell; \mu_2, \sigma_2^2),$$

with $\pi_{\mathrm{clean}} + \pi_{\mathrm{noisy}} = 1$. The noise-rate estimate is simply:

$$\hat{\tau}_{\mathrm{GMM}} = \pi_{\mathrm{noisy}}.$$

As we can see in Figure 7.4, the lower-loss component (blue) captures clean examples, while the higher-loss component (orange) captures noisy ones. further improve separability by using the area under the loss (AUL) curve, summing losses across epochs; this makes the two modes more distinct even late in training.



(a) Symmetric Noise. (b) Asymmetric Noise.

Figure 7.4. Training loss distributions of true-labeled and false-labeled examples using the ground-truth label and the GMM on CIFAR-100 data with two synthetic noises of 40%. Adapted from Song et al. [70]

C. Cross-Validation Search

A model-agnostic but computationally heavier approach treats τ as a hyperparameter over a grid $\{\tau_k\}$. For each τ_k:

1. Train the chosen robust method assuming noise rate τ_k.
2. Evaluate validation accuracy $Acc_V(\tau_k)$ on a small clean set.
3. Select
$$\hat{\tau}_{\mathrm{CV}} = \arg\max_{\tau_k} Acc_V(\tau_k).$$

This yields an estimate that directly optimizes downstream performance, at the cost of retraining for each candidate. By combining multiple estimators or selecting one that best fits the data regime at hand, practitioners can obtain reliable $\hat{\tau}$ values to drive robust training under label noise.

7.3 Experimental Design

This section outlines the standard experimental setup used to compare robust training methods under label noise. We first introduce the publicly available image datasets employed, then describe the evaluation metrics applied in our studies.

7.3.1 Publicly Available Datasets

To validate algorithmic robustness, image classification tasks are run on a diverse collection of benchmark datasets. These are grouped into two categories:

Clean Datasets These datasets consist of nearly all correctly labeled examples, annotated by experts. For synthetic-noise experiments, labels are artificially corrupted to simulate controlled noise levels (symmetric or asymmetric). The seven most common clean datasets are:

- **MNIST:** 60,000 training and 10,000 test gray-scale images of handwritten digits (10 classes).
- **Fashion-MNIST:** 60,000 training and 10,000 test gray-scale images of clothing items (10 classes).
- **CIFAR-10:** 50,000 training and 10,000 test RGB images across 10 object categories.
- **CIFAR-100:** 50,000 training and 10,000 test RGB images across 100 fine-grained classes.
- **SVHN:** 73,257 training and 26,032 test RGB images of street-view house numbers (10 classes).
- **Tiny-ImageNet:** 100,000 training, 10,000 validation, and 10,000 test RGB images over 200 classes (a subset of ImageNet).
- **ImageNet:** 1.3 million training, 50,000 validation, and 50,000 test RGB images spanning 1,000 classes.

Real-World Noisy Datasets These datasets contain naturally occurring label errors, often sourced from web queries or crowd-sourced annotations. Each provides its own clean validation set and an estimated overall noise rate. The six widely used real-world noisy datasets are:

- **Animal-10N:** 50,000 training and 5,000 test images of 10 animal classes with ∼8% label noise. (10 classes)
- **CIFAR-10N:** Variants of CIFAR-10 with crowd-sourced noise at approximately 9%, 18%, or 40% levels. (10 classes)
- **CIFAR-100N:** Variants of CIFAR-100 with crowd-sourced noise at approximately 25.6% or 40.2%. (100 classes)
- **Food-101N:** 310,000 training, 5,000 validation, and 25,000 test food images with ∼18.4% noise. (101 classes)
- **Clothing1M:** 1 million training, 14,000 validation, and 10,000 test clothing images with ∼38.5% noise. (14 classes)
- **WebVision:** 2.4 million training, 50,000 validation, and 50,000 test web-sourced images over 1,000 classes with ∼20% noise. (100 classes)

7.3.2 Evaluation Metrics

A typical metric to assess the robustness of a particular method is the prediction accuracy for unbiased and clean examples that are not used in training. The prediction accuracy degrades significantly if the DNN overfits to false-labeled examples. Hence, test accuracy has generally been adopted for evaluation. For a

test set $T = \{(x_i, y_i)\}_{i=1}^{|T|}$, let $\hat{y}_i$ be the predicted label of the i-th example in T. Subsequently, the test accuracy is formalized by:

$$\text{Test Accuracy} = \frac{|\{(x_i, y_i) \in T : \hat{y}_i = y_i\}|}{|T|}. \tag{7.3}$$

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

$$\text{Validation Accuracy} = \frac{|\{(x_i, y_i) \in V : \hat{y}_i = y_i\}|}{|V|}. \tag{7.4}$$

Furthermore, if the specified method belongs to the *sample selection* category, label precision and label recall can be used as metrics:

$$\text{Label Precision} = \frac{|\{(x_i, \tilde{y}_i) \in S_t : \tilde{y}_i = y_i\}|}{|S_t|}, \quad \text{Label Recall} = \frac{|\{(x_i, \tilde{y}_i) \in S_t : \tilde{y}_i = y_i\}|}{|\{(x_i, \tilde{y}_i) \in B_t : \tilde{y}_i = y_i\}|}, \tag{7.5}$$

where S_t is the set of selected clean examples in a mini-batch B_t. These two metrics indicate how accurately true-labeled examples are identified from the mini-batch.

Meanwhile, if the method belongs to the *label refurbishment* category, *correction error* can be used to evaluate how many examples are incorrectly refurbished:

$$\text{Correction Error} = \frac{|\{x_i \in R : \arg\max(y_i^{\text{refurb}}) \neq y_i\}|}{|R|}, \tag{7.6}$$

where R is the set of examples whose labels are refurbished, and y_i^{refurb} is the refurbished label of the i-th example in R.

Chapter 8

Conclusions and Future Works

8.1 Conclusions

This thesis explored the challenge of training deep neural networks on datasets with noisy labels, with a particular focus on early stopping as a mechanism to mitigate memorization. Through a comprehensive literature review, feature and empirical evaluation of 14 state-of-the-art methods, several key conclusions were drawn.

Robust Training Improves Generalization:
Modern robust training methods significantly improve generalization under label noise. All evaluated methods outperformed the baseline when noise was non-trivial, with some recovering accuracy close to clean training even under 40% corruption. This highlights the effectiveness of techniques such as loss correction, sample selection, and semi-supervised learning. Moreover, combining complementary strategies—like label cleaning with regularization—often leads to further gains. A consistent finding is that standard regularization alone (e.g., data augmentation or weight decay) is insufficient; specialized noise-aware mechanisms are essential.

Early Learning Dynamics Are Crucial:
A recurring theme across theory and practice is the importance of early training dynamics. Networks tend to learn clean patterns early on, before gradually memorizing noise. This two-phase behavior was clearly visible in our experiments: test accuracy often peaked early, then declined. Methods that harness this pattern—either through early stopping or filtering—achieved better performance. Even a simple early stopping rule based on clean validation loss was competitive with more complex approaches. Many robust methods implicitly implement a form of early stopping, treating noisy samples as if they were stopped on early. Thus, early stopping—explicit or implicit—is central to preventing memorization and preserving generalizable knowledge.

General Principles, Context-Specific Tools:
While robust learning methods are diverse, common principles like simplicity and exploiting learning dynamics often underlie their success. Lightweight techniques like ELR, which use prior predictions, performed on par with more complex models. However, effectiveness can depend on the noise regime: semi-supervised learning becomes vital under high noise levels, while instance-dependent noise remains especially challenging and may require new approaches. Therefore, method selection

should be guided by noise characteristics, although early stopping and the use of unlabeled data appear broadly useful across settings.

Early Stopping as a Core Mechanism:
This thesis demonstrated that early stopping is more than a training heuristic—it is a principled and practical strategy for robustness. It aligns with the natural dynamics of neural networks, is easy to integrate, and can enhance even advanced pipelines. Its simplicity and general applicability make it a strong baseline and a valuable addition to any noise-robust learning system. As shown throughout the thesis, sometimes the most effective defense against label noise is simply knowing when to stop.

8.2 Future Work

While this thesis provides an extensive analysis of early stopping techniques and robust training methods under label noise, numerous challenges remain unaddressed. The development of reliable deep learning systems in noisy environments calls for future research in several emerging directions. The following areas highlight critical frontiers that merit further investigation:

A. Advancing Learning under Instance-Dependent Label Noise

Most robust learning techniques and theoretical models are built on the assumption of instance-independent label noise, where the probability of label corruption is independent of input features. However, real-world datasets, such as Clothing1M, exhibit instance-dependent noise patterns, where certain input characteristics increase the likelihood of mislabeling. Under such conditions, traditional sample selection or loss correction methods may fail due to heavy overlap between clean and noisy loss distributions. Future research should focus on designing methods that explicitly account for input-dependent noise—especially for multi-class and high-dimensional settings—by leveraging instance-level features, dynamic uncertainty modeling, or local neighborhood statistics.

B. Robust Multi-Label Learning with Noisy Annotations

Most robust training methods assume single-label classification, where each instance corresponds to one true class. However, in multi-label learning scenarios—common in music tagging, scene recognition, and medical diagnostics—each instance may be associated with multiple true labels. These datasets frequently suffer from missing or incorrect labels and suffer from label co-occurrence bias. Addressing noise in this context requires novel strategies that can model inter-label dependencies while still handling label corruption. Future directions include the development of robust multi-label loss functions, noise-aware label correlation modeling, and scalable architectures for partial supervision.

C. Handling Label Noise in the Presence of Class Imbalance

Many real-world datasets suffer from long-tailed distributions where some classes are underrepresented. When combined with label noise, class imbalance exacerbates the problem—small-loss sample selection strategies may inadvertently discard rare but important classes, leading to biased or incomplete models. Moreover, robust training often favors small-loss instances (assuming cleanliness), while imbalance-aware training typically focuses on high-loss or rare examples—creating a conceptual conflict. Future research must develop integrated solutions that jointly address label noise and class imbalance, potentially via dual-objective optimization, class-conditional noise modeling, or adaptive loss weighting mechanisms.

D. Toward Fair and Robust Deep Learning

Fairness in machine learning aims to ensure equitable performance across demographic or protected groups, while robustness focuses on generalization under noisy supervision. These two objectives are often studied in isolation. However, noisy labels and biased data distributions often co-occur in real-world applications, leading to models that are neither fair nor robust. Future research must explore unified frameworks that ensure both robustness to label noise and fairness across groups. This requires new fairness metrics that remain valid under noisy conditions, as well as optimization strategies that enforce group parity without relying on clean or accurately labeled subgroup annotations.

E. Bridging Robust Learning and Adversarial Training

Traditionally, robustness to input perturbations (e.g., adversarial attacks) and robustness to label noise have been treated as separate concerns. However, recent work suggests that adversarial training—originally designed to defend against input corruption—can also improve robustness to noisy labels. This emerging connection invites a deeper investigation into how adversarial objectives, regularization techniques, and perturbation-based sample selection can be used to filter or correct noisy labels. Future research may explore joint frameworks that learn noise-tolerant representations via both adversarial and label-cleaning mechanisms.

F. Designing Efficient and Scalable Robust Learning Pipelines

Most robust training techniques—especially those involving multiple networks, meta-learning modules, or iterative filtering—impose significant computational overhead. As datasets scale to millions of samples, there is an urgent need for efficient training pipelines that balance robustness and speed. Methods that reduce training time, memory usage, or model complexity—without compromising performance—will be essential for practical deployment. Future work should explore lightweight single-network solutions, early-stopping variants optimized for runtime, or approximation-based filtering strategies that require fewer computations per epoch.

Bibliography

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

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning (ICML)*, 2019.

[3] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

[4] Anshuman Awasthi, Akilesh Ramaswamy, and Gaurav Manek. Sure: Sample uncertainty ranking for estimation under noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[5] Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. Understanding and improving early stopping for learning with noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[7] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[8] Hao Chen, Ankit Shah, Jindong Wang, Ran Tao, Yidong Wang, Xiang Li, Xing Xie, Masashi Sugiyama, Rita Singh, and Bhiksha Raj. Imprecise label learning: A unified framework for learning with various imprecise label configurations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020.

[10] Wenkai Chen, Chuang Zhu, Yi Chen, Mengting Li, and Tiejun Huang. Sample prior guided robust model learning to suppress noisy labels. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[11] De Cheng, Tongliang Liu, Yixiong Ning, Bo Chen, and Dacheng Tao. Instance-dependent label-noise learning with manifold-regularized transition matrix estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16631–16640, 2022.

[12] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. Learning with instance-dependent label noise: A sample sieve approach. In *International Conference on Learning Representations (ICLR)*, 2023. `https://openreview.net/forum?id=RydTeV5twX`.

[13] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML)*, 2008.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

[15] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[16] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2013.

[17] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[18] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations (ICLR)*, 2017. Poster.

[19] Bo Han et al. A survey of label-noise representation learning: Past, present and future. *arXiv preprint arXiv:2011.04406*, 2020.

[20] Bo Han, Jiangchao Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8527–8537, 2018.

[21] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2712–2721, 2019.

[22] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[23] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[24] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations (ICLR)*, 2017.

[25] Huaxi Huang, Hui Kang, Sheng Liu, Olivier Salvado, Thierry Rakotoarivelo, Dadong Wang, and Tongliang Liu. Paddles: Phase-amplitude spectrum disentangled early stopping for learning with noisy labels. *arXiv preprint arXiv:2207.11353*, 2022.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[27] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2018.

[28] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning (ICML)*, 2018.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

[30] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1992.

[31] Jikai Li and Tengyu Ma. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[32] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[33] Junnan Li, Hengshu Zhao, Zhen Liu, Yong Zhang, Wayne Zhang, Xiangyang Xue, and Weiyao Zhang. Learning from noisy labels with sparse overparameterization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. *arXiv preprint arXiv:1903.11680*, 2019.

[35] Wen Li, Limin Wang, Wei Li, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[36] Yifan Li, Yifan Zhou, Xiaolin Wu, and Shanghang Wang. Prediction-guided divide-and-filter for learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[37] Yuting Li, Yingyi Chen, Xuanlong Yu, Dexiong Chen, and Xi Shen. Sure: Survey recipes for building reliable and robust deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[38] Yunchao Lin, Xiaohui Shen, Zhe Liu, Wangmeng Zuo, Rainer Mech, and Alan L Yuille. Learning from weakly labeled web data with category-level supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] Fengbei Liu, Meng Li, Lei Yu, Hao Xu, and Dacheng Tao. Imprecise label learning: A unified framework for learning with various imprecise label configurations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[40] Fengbei Liu, Chong Wang, Yuanhong Chen, Yuyuan Liu, and Gustavo Carneiro. Partial label supervision for agnostic generative noisy label learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024.

[41] Huafeng Liu, Mengmeng Sheng, Zeren Sun, et al. Learning with imbalanced noisy data by preventing bias in sample selection. *arXiv preprint arXiv:2402.11242*, 2024.

[42] Sheng Liu, Kangning Liu, Weicheng Zhu, Yiqiu Shen, and Carlos Fernandez-Granda. Adaptive early-learning correction for segmentation from noisy annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.

[43] Sheng Liu, N Natarajan, and Inderjit S Dhillon. Early-learning regularization prevents memorization of noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 4381–4392, 2020.

[44] Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

[45] Shikun Liu, Chen Xie, Mingsheng Long, and Jianmin Wang. Pes: Progressive early stopping for learning with noisy labels. *arXiv preprint arXiv:2106.08177*, 2021.

[46] Ricard V Lloyd, Lori A Erickson, Michael B Casey, Kwan K Lam, Christine M Lohse, et al. Observer variation in the diagnosis of follicular variant of papillary

thyroid carcinoma. *American Journal of Surgical Pathology*, 28(10):1336–1340, 2004.

[47] Jiale Ma, Yisen Guo, Yuhang Wang, Lei Zhang, and Jiwen Lu. Paddles: Phase-amplitude spectrum disentangled early stopping for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[48] Long Ma, Ziyu Song, Yisen Huang, and Yi Yang. Normalized loss functions for deep learning with noisy labels. In *International Conference on Machine Learning (ICML)*, 2020.

[49] Eran Malach and Shai Shalev-Shwartz. Decoupling "when to update" from "how to update". In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 960–970, 2017.

[50] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. The stanford typed dependencies representation. In *Proceedings of the COLING 2008 Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 2008.

[51] Winter Mason and Siddharth Suri. Conducting behavioral research on amazon's mechanical turk. *Behavior Research Methods*, 44(1):1–23, 2012.

[52] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[53] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1196–1204, 2013.

[54] Kamil D Onal, Hamed Zhang, Richard Berendsen, Maarten de Rijke, Evangelos Kanoulas, and Krisztian Balog. Neural information retrieval: A literature review. *Information Retrieval Journal*, 21:111–182, 2018.

[55] Samet Oymak, Zeyuan Fabian, Mingchen Li, and Mahdi Soltanolkotabi. Generalization guarantees for neural networks via harnessing the low-rank structure of the jacobian. *arXiv preprint arXiv:1906.05392*, 2019.

[56] Liangliang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 26th ACM International Conference on Information and Knowledge Management (CIKM)*, 2017.

[57] Gabriele Paolacci, Jesse Chandler, and Panos G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision Making*, 5(5):411–419, 2010.

[58] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[59] Geoff Pleiss, Tianyi Chen, Gabriel Huang, and Kilian Q Weinberger. Detecting noisy training data with loss curves. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[60] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[61] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations (ICLR)*, 2015.

[62] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 4331–4340, 2018.

[63] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, et al. Theoretical analysis of self-supervised contrastive learning with noisy labels. *arXiv preprint arXiv:1902.09229*, 2019.

[64] Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

[65] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[66] Jun Shu, Zeming Xie, Zeyuan Yi, Qian Zhao, Deyu Meng, and Zongben Xu. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[67] Kihyuk Sohn, David Berthelot, Chun Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, and Han Zhang. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020.

[68] Chenglin Song, Sheng Liu, Yisen Huang, and Dacheng Tao. How does learning rate affect generalization ability of deep networks with label noise? In *International Conference on Machine Learning (ICML)*, 2020.

[69] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. How does early stopping help generalization against label noise? In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

[70] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[71] Alex Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *CVPR Workshops*, 2008.

[72] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[73] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. In *International Conference on Learning Representations (ICLR)*, 2015.

[74] Daiki Tanaka, Daisuke Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5552–5560, 2018.

[75] Luke Taylor and Geoff Nitschke. Improving deep learning with generic data augmentation. *arXiv preprint arXiv:1708.06020*, 2017.

[76] William Toner and Amos Storkey. Noisy early stopping for noisy labels. *arXiv preprint arXiv:2402.09839*, 2024.

[77] Yisen Tu, Xudong Yu, Hengshu Zhao, Shuai Yan, and Le Sun. Learning from noisy labels with decoupled meta label purifier. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24217–24226, 2023.

[78] Yidong Wang, Han Zhang, and et al. Pico: Contrastive label disambiguation for partial label learning. In *NeurIPS*, 2022.

[79] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jilin Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *International Conference on Computer Vision (ICCV)*, 2019.

[80] Farooq Ahmad Wani, Maria Sofia Bucarelli, and Fabrizio Silvestri. Learning with noisy labels through learnable weighting and centroid similarity. *arXiv preprint arXiv:2309.07542*, 2023.

[81] Hongxin Wei, Lei Feng, Xiang Chen, Bo An, and Bo Liu. Combating noisy labels by agreement: A joint training method with co-regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13726–13735, 2020.

[82] Jingkang Wei, Qian Song, Xiyang Ma, Yisen Tong, Hang Su, and Jun Zhu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations (ICLR)*, 2022.

[83] Xinyu Xia, Tongliang Liu, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Extended t: Learning with mixed closed-set and open-set noisy labels. *arXiv preprint arXiv:2003.03680*, 2020.

[84] Han Xiao, Wei Xia, Zhizhong Lu, and Yong Zhang. Noise or signal: The role of image variation in deep network training. In *International Conference on Machine Learning (ICML)*, 2020.

[85] Han Xiao, Hong Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *European Conference on Artificial Intelligence (ECAI)*, 2012.

[86] Ruixuan Xiao, Yiwen Dong, Haobo Wang, Lei Feng, Runze Wu, Gang Chen, and Junbo Zhao. Promix: Combating label noise via maximizing clean sample utility. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[87] Tong Xiao, Tianzhe Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[88] Zhi Xu, Yuan Zhang, Yisen Luo, Chaowei Xiao, Jiefeng Ma, Bo Li, and Dawn Song. Understanding training dynamics of deep neural networks via fourier analysis. In *International Conference on Machine Learning (ICML)*, 2019.

[89] Xiaowei Xue, Jian Wu, and Weilin Huang. Investigating why contrastive learning benefits robustness against noisy labels. In *International Conference on Machine Learning (ICML)*, 2022.

[90] Yucheng Yao, Xiaotian Ma, Shifeng Li, Yisen Wang, and Shijiang Liu. Jo-src: A contrastive approach for combating noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[91] Kwanghee Yi and Jiancheng Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7019–7027, 2019.

[92] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Access*, 6:41342–41357, 2018.

[93] Xingrui Yu, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. How does instance-dependent label noise affect the sample complexity of margin-based active learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[94] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[95] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

[96] Weinan Zhang, Ting Du, and Jun Wang. Deep learning over multi-field categorical data. In *European Conference on Information Retrieval (ECIR)*, 2016.

[97] Xiyang Zhang, Qian Liu, Ming Lin, et al. Contrastive dividemix: Learning with noisy labels via contrastive semi-supervised learning. *arXiv preprint arXiv:2110.05214*, 2021.

[98] Xuhong Zhang, Qian Liu, Min Lin, Pengfei Zhang, Yunfeng Zhang, and Ping Luo. Contrastive dividemix (codim): Learning with noisy labels via contrastive semi-supervised learning. *arXiv preprint arXiv:2110.05214*, 2021.

[99] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[100] Yuyin Zhou, Xianhang Li, Fengze Liu, and Qingyue Wei. Learning to bootstrap robust models for combating label noise (l2b). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[101] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review*, 22(3):177–210, 2004.